

Ю. Ф. Лазарєв

Моделювання динамічних систем у Matlab

Електронний навчальний посібник

Рекомендовано Методичною радою НТУУ "КПІ"

Київ – КПІ – 2011

**УДК 681.3(0.75)
Л17**

Лазарєв Ю. Ф.

Л17 Моделювання динамічних систем у Matlab. Електронний навчальний посібник. – Київ: НТУУ "КПІ", 2011. – 421 с.

Викладені основи теорії моделювання. Детально розглянуті питання чисельного (програмного) моделювання динамічних систем, які описуються звичайними диференціальними рівняннями. Наводяться необхідні початківцю відомості про комп'ютерну систему Matlab. Показані можливості і особливості застосування Matlab для моделювання динамічних систем.

Призначено для курсового і дипломного проектування студентів вищих навчальних закладів. Може бути корисним для науковців і інженерам для підвищення кваліфікації.

Іл. 338, табл. 2

ЗМІСТ

1. МОДЕЛІ І МОДЕЛЮВАННЯ.....	7
1.1. МАТЕМАТИКА Й РЕАЛЬНІСТЬ	9
1.2. ТЕОРІЯ ПОДІБНОСТІ І АНАЛІЗ РОЗМІРНОСТЕЙ	12
1.2.1. Подібність явищ і її ознаки	12
1.2.2. Основні положення теорії подібності	13
1.2.3. Аналіз розмірностей.....	15
1.2.4. Приклади застосування аналізу розмірностей.....	16
1.2.5. Застосування теорії подібності для побудови моделей.....	19
1.3. ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ	22
2. ЕТАПИ РОЗВ'ЯЗУВАННЯ ІНЖЕНЕРНИХ ЗАДАЧ НА ЕОМ.....	23
2.1. ПОСТАНОВКА ЗАДАЧІ	24
2.2. СТВОРЕННЯ МАТЕМАТИЧНОЇ МОДЕЛІ	24
2.3. МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ	27
2.4. ПОБУДОВА ОБЧИСЛЮВАЛЬНОЇ МОДЕЛІ.....	28
2.5. АЛГОРИТМ МЕТОДА.....	29
2.6. РЕАЛІЗАЦІЯ МЕТОДУ ОБЧИСЛЕНЬ	30
2.7. ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ	32
3. ОРГАНІЗАЦІЯ НАБЛИЖЕНИХ ОБЧИСЛЕНЬ	33
3.1. Джерела й види похибок	33
3.2. ЗАПИС НАБЛИЖЕНИХ ЧИСЕЛ. ПРАВИЛО ОКРУГЛЕННЯ.....	35
3.3. ПОХИБКИ РЕЗУЛЬТАТУ ПРИ ДІЯХ ІЗ НАБЛИЖЕНИМИ ЧИСЛАМИ	37
3.3.1. Похибки підсумовування.....	37
3.3.2. Похибки добутку, ділення й обчислення довільної функції.....	39
3.4. ПОШИРЕННЯ ПОХИБОК ОКРУГЛЕННЯ ПРИ ОБЧИСЛЕННЯХ. ЗБЕРІГАННЯ ЧИСЕЛ В ЕОМ	40
3.5. ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ	47
4. ДИНАМІЧНІ СИСТЕМИ.....	48
4.1. ОСНОВНІ ПОНЯТТЯ ТЕОРІЇ ДИНАМІЧНИХ СИСТЕМ	48
4.2. ОСНОВИ ПРОГРАМНОГО МОДЕЛЮВАННЯ ДИНАМІЧНИХ СИСТЕМ	55
4.2.1. Нормальна форма Коші диференціальних рівнянь, фазові змінні.....	55
4.2.2. Чисельне інтегрування диференціальних рівнянь	56
4.2.3. Експериментальне дослідження похибок чисельного інтегрування.....	60
4.3. ЗАДАЧІ І ОСОБЛИВОСТІ МОДЕЛЮВАННЯ ГІРОСКОПІЧНИХ ПРИСТРОЇВ	66
4.4. ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ	68
5. ЗНАЙОМСТВО З MATLAB	69
5.1. КОМАНДНЕ ВІКНО	69
5.2. ОПЕРАЦІЇ З ЧИСЛАМИ	70
5.2.1. Введення дійсних чисел.....	70
5.2.2. Найпростіші арифметичні дії.....	73
5.2.3. Введення комплексних чисел	75
5.2.4. Елементарні математичні функції.....	76
5.2.5. Спеціальні математичні функції.....	77
5.2.6. Елементарні дії з комплексними числами	78
5.2.7. Функції комплексного аргументу.....	79
5.2.8. Знайомство з програмуванням в Matlab.....	80
5.2.9. Запитання для самоперевірки.....	82
5.3. ОПЕРАЦІЇ З ВЕКТОРАМИ	83
5.3.1. Введення векторів і матриць	84
5.3.2. Дії над векторами.....	86
5.3.3. Операції з поліномами.....	88
5.3.4. Процедура plot.....	91
5.3.5. Спеціальні графіки.....	94
5.3.6. Додаткові функції графічного вікна	99
5.3.7. Вставлення графіків до документу	101
5.3.8. Апроксимація та інтерполяція даних.....	101

5.3.9. Векторна фільтрація й спектральний аналіз	104
5.3.10. Запитання для самоперевірки.....	110
5.4. ОПЕРАЦІЇ З МАТРИЦЯМИ.....	110
5.4.1. Формування матриць	110
5.4.2. Витягання й вставляння частин матриць.....	114
5.4.3. Обробка даних вимірів.....	116
5.4.4. Поелементне перетворення матриць.....	121
5.4.5. Матричні дії над матрицями	122
5.4.6. Розв'язування систем лінійних алгебричних рівнянь	124
5.4.7. Матричні функції.....	127
5.4.8. Функції лінійної алгебри	129
5.4.9. Запитання для самоперевірки.....	137
5.5. ЛІТЕРАТУРА	138
6. ПРОГРАМУВАННЯ У МАТЛАВ	139
6.1. ОПЕРАТОРИ КЕРУВАННЯ ОБЧИСЛЮВАЛЬНИМ ПРОЦЕСОМ.....	142
6.1.1. Оператор умовного переходу	142
6.1.2. Оператор переключення.....	143
6.1.3. Оператори циклу.....	144
6.2. СТВОРЕННЯ НАЙПРОСТІШИХ ФАЙЛ-ФУНКЦІЙ (ПРОЦЕДУР)	146
6.2.1. Загальні вимоги до побудови	146
6.2.2. Типове оформлення процедури-функції.....	149
6.2.3. Запитання для самоперевірки.....	149
6.3. СТВОРЕННЯ SCRIPT-ФАЙЛІВ	149
6.3.1. Основні особливості Script-файлів.....	149
6.3.2. Введення й виведення інформації у діалоговому режимі	150
6.3.3. Організація повторювання дії	153
6.3.4. Організація змінювання даних у діалоговому режимі	153
6.3.5. Типова структура й оформлення Script-файлу.....	156
6.4. ГРАФІЧНЕ ОФОРМЛЕННЯ РЕЗУЛЬТАТІВ	157
6.4.1. Загальні вимоги до подання графічної інформації.....	157
6.4.2. Розбиття графічного вікна на підвікна.....	159
6.4.3. Виведення тексту в графічне вікно (підвікно)	160
6.5. ФУНКЦІЇ ФУНКЦІЙ	163
6.5.1. Стандартні функції від функцій Matlab.....	163
6.5.3. Запитання для самоперевірки.....	166
6.6. СТВОРЕННЯ ФУНКЦІЙ ВІД ФУНКЦІЙ	166
6.6.1. Процедура feval	166
6.6.2. Приклади створення процедур від функцій	167
6.6.2.1. Процедура методу Рунге-Кутта 4-го порядку	167
6.6.2.2. Процедура правих частин рівняння маятника	168
6.7. ПРОГРАМА МОДЕЛЮВАННЯ РУХУ МАЯТНИКА	172
6.8. ЛІТЕРАТУРА	182
7. ПАКЕТ ПРОГРАМ SIMULINK.....	183
7.1. ЗАПУСК SIMULINK	183
7.2. БІБЛІОТЕКА SIMULINK	185
7.2.1. Поділ Sinks (Приймачі)	187
7.2.2. Поділ Sources (Джерела).....	196
7.2.3. Поділ Continuous (Неперервні елементи)	209
7.2.4. Поділ Discrete (Дискретні елементи)	213
7.2.5. Поділ Discontinuities (Розривні елементи).....	216
7.2.6. Поділ Signal Routing (Пересилання сигналів).....	218
7.2.7. Поділ Math Operations (Математичні операції).....	220
7.2.8. Поділ Logic & Bit Operations (Логічні та бітові операції).....	224
7.2.9. Поділ User-defined Functions (Функції, що визначаються користувачем)	225
7.2.10. Поділ Ports & Subsystems (Порти та підсистеми).....	226
7.3. ПОБУДОВА БЛОК-СХЕМ	227
7.3.1. Виділення об'єктів.....	227
7.3.2. Операції з блоками	229
7.3.3. Проведення з'єднувальних ліній	232
7.3.4. Мітки сигналів і коментарів.....	235

7.3.5. Створення підсистем.....	238
7.3.6. Зберігання і виведення до друку блок-схеми	239
7.4. ПРИКЛАДИ УТВОРЕННЯ S-МОДЕЛЕЙ	240
7.4.1. Модель поводження фізичного маятника.....	240
7.4.2. Моделювання руху трьох тіл під дією сил гравітації.....	246
7.5. ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ	256
7.6. ЛІТЕРАТУРА.....	256
8. ЗАСОБИ ВЗАЄМОДІЇ МАТЛАВ З SIMULINK.....	257
8.1. Об'єднання S-МОДЕЛЕЙ З ПРОГРАМАМИ МАТЛАВ.....	258
8.1.1. Керування процесом моделювання у Simulink.....	258
8.1.2. Виявлення перетинання нуля	261
8.1.3. Обмін даними між середовищем Matlab і S-моделлю	266
8.1.4. Запуск процесу моделювання S-моделі з середовища Matlab.....	269
8.1.5. Створення S-блоків з використанням програм Matlab. S-функції	270
8.1.6. Приклад утворення і роботи з S-функцією	273
8.1.7. Запуск M-програм із S-моделі.....	279
8.2. КОРИСТУВАЦЬКІ БІБЛІОТЕКИ S-БЛОКІВ.....	288
8.2.1. Утворення бібліотеки.....	288
8.2.2. Утворення вікна настроювання (маски) блоку.....	293
8.3. ПРИКЛАДИ ЗАСТОСУВАННЯ КОРИСТУВАЦЬКОЇ БІБЛІОТЕКИ.....	297
8.3.1. Орієнтування космічного апарату.....	297
8.3.2. Маятник під дією сил сухого тертя.....	305
8.4. ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ	314
8.5. ЛІТЕРАТУРА.....	315
9. МОЖЛИВОСТІ МАТЛАВ ДЛЯ МОДЕЛЮВАННЯ ДИНАМІЧНИХ СИСТЕМ.....	316
9.1. УТВОРЕННЯ І ВИКОРИСТОВУВАННЯ ВЛАСНИХ КЛАСІВ ОБЧИСЛЮВАЛЬНИХ ОБ'ЄКТІВ	316
9.2. ОБРОБКА ЦИФРОВИХ СИГНАЛІВ І ПРОЕКТУВАННЯ ФІЛЬТРІВ.....	320
9.3. АНАЛІЗ І СИНТЕЗ СИСТЕМ АВТОМАТИЧНОГО КЕРУВАННЯ	320
9.4. ВІЗУАЛЬНЕ ПРОГРАМУВАННЯ.....	321
9.5. ЗАСОБИ СУМІСНОГО ВИКОРИСТАННЯ МАТЛАВ І SIMULINK.....	322
9.6. УТВОРЕННЯ І ВИКОРИСТОВУВАННЯ ВЛАСНИХ БІБЛІОТЕК БЛОКІВ КОРИСТУВАЧА.....	328
10. ВИКОРИСТАННЯ БІБЛІОТЕКИ AEROSPACE	329
10.1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА БІБЛІОТЕКИ AEROSPACE	329
10.1.1. Розділ <i>Equations of Motion</i>	330
10.1.2. Розділ <i>Environment</i>	335
10.1.3. Розділ <i>Propulsion</i>	336
10.1.4. Розділи <i>Actuators</i> і <i>GNC</i>	337
10.1.5. Розділ <i>Transformations</i>	340
10.2. МОДЕЛЬ ВІЛЬНОГО КУТОВОГО РУХУ КОСМІЧНОГО АПАРАТУ	342
10.3. МОДЕЛЬ КЕРОВАНОГО КУТОВОГО РУХУ КОСМІЧНОГО АПАРАТА	346
10.4. МОДЕЛЬ РУХУ ШТУЧНОГО СУПУТНИКА ЗЕМЛІ	354
10.5. ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ	359
11. ЗАГАЛЬНА ХАРАКТЕРИСТИКА БІБЛІОТЕКИ SIMPOWERSYSTEMS.....	360
11.1. РОЗДІЛ ELECTRICAL SOURCES	361
11.2. РОЗДІЛ ELEMENTS	366
11.3. РОЗДІЛ CONNECTORS	371
11.4. РОЗДІЛ POWER ELECTRONICS	371
11.5. РОЗДІЛ MACHINES	372
11.6. РОЗДІЛ MEASUREMENTS	376
12. ВИКОРИСТАННЯ БІБЛІОТЕКИ SIMMECHANICS.....	378
12.1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА БІБЛІОТЕКИ SIMMECHANICS	379
12.1.1. Розділ <i>Bodies</i>	381
12.1.2. Розділ <i>Joints</i>	384
12.1.3. Розділ <i>Sensors&Actuators</i>	390
12.1.4. Розділ <i>Constraints & Drivers</i>	395
12.1.5. Розділ <i>Utilities</i>	396

12.2. МОДЕЛЬ ЗРІВНОВАЖЕНОГО ВІЛЬНОГО ГІРОСКОПА	397
12.3. МОДЕЛЬ КРИВОШИПНО-ШАТУННОГО МЕХАНІЗМУ	402
12.4. МОДЕЛЬ РУХУ МАЯТНИКА	407
12.5. ЗАПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ	414
ЛІТЕРАТУРА	415
АБЕТКОВИЙ ПОКАЖЧИК	416

1. МОДЕЛІ І МОДЕЛЮВАННЯ

Моделювання є основою пізнання людиною навколишнього світу. Проводячи експерименти, теоретичні досліджування, навіть обговорювання власних дій, намірів, висновків, ми фактично здійснюємо моделювання. Цілі, задачі, засоби й методи моделювання у цих випадках значно відрізняються один від одного, але загальна спрямованість залишається єдиною – одержання нового знання шляхом випробування (досліджування) деякого замітника реального об'єкта дослідження – моделі.

Взагалі, спрощено, моделювання можна розглядати як певний експеримент, об'єктом якого у першому випадку є матеріальний аналог досліджуваного об'єкта, у другому випадку об'єктом випробувань є знакова (математична) модель, у третьому – відношення до моделі, яка обмірковується, з боку громади.

Поняття моделі можна визначити у такий спосіб:

модель – це природний або штучний реальний об'єкт, який має певну відповідність до другого реального об'єкта (оригінала), поведження якого потрібно вивчити, або до деяких сторін оригінала, які потребують вивчення.

У процесі *дослідження моделі* (який називають *моделюванням*) вона постає у якості відносно самостійного квазі-об'єкта, який дозволяє шляхом його дослідження одержати деяке опосередковане знання про оригінал.

У випадку експериментальних досліджень моделлю є реальний об'єкт, який має ту саму фізичну природу, що й досліджуваний об'єкт. При теоретичних досліджуваннях модель має знакову форму – математичних формул, співвідношень, рівнянь, а задачею моделювання є встановлення нових знань про об'єкти, що описуються цими співвідношеннями. Обговорення встановлює слушність тих припущень і висновків, які були зроблені, шляхом моделювання відношення до них досвідчених співрозмовників.

Пізнання за допомогою моделювання, як це впливає з зазначеного, складається з наступних етапів:

- 1) побудова моделі або обрання її з існуючих;
- 2) дослідження моделі з потрібного боку – власне моделювання;
- 3) перенесення вивчених властивостей моделі на властивості оригінала, тобто виявлення поведження оригіналу за одержаними відомостями про вивчені властивості моделі.

До моделювання вдаються тоді, коли потребують пізнання деяких властивостей об'єкта вивчення, але при цьому сам об'єкт є недосяжним, або його вивчення наштовхується на значні труднощі і незручності. За таких обставин *одним з найважливіших етапів здійснення моделювання є утворення (або обрання) спеціального реального об'єкта дослідження – моделі – з наступними властивостями:*

– він має властивості, що є подібними до відповідних властивостей об'єкта досліджування (оригінала), які потребують досліджування;

– він є більш доступним, більш простим і зручним для досліджування, аніж оригінал, щоб можливо було безпосередньо дослідити бажані властивості.

Як бачимо, щоб утворити (побудувати) модель, з одного боку, потрібно щось знати про об'єкт досліджування (щоб бути впевненим, що в ній збережені досліджувані властивості), хоча, з іншого боку, знання про об'єкт досліджування обов'язково є неповними (інакше створення моделі не матиме сенсу). Тобто модель не потрібно утворювати, коли в об'єкті досліджування не має властивостей, які потрібно дослідити, і неможливо створити, коли про об'єкт досліджування не відомо нічого.

У подальшому обмежимося розглядом науково-технічного моделювання, тобто моделювання технічних (штучно створених) об'єктів на ґрунті наукових знань про поведінку таких об'єктів. Наукові знання базуються на математичном формулюванні головних властивостей об'єкта. Тому опис поведінки об'єкта у вигляді математичних співвідношень, рівнянь (алгебричних, диференціальних або інтегральних) є основою як для побудови моделі, так і для перенесення результатів моделювання на оригінал.

Зупинемося більш докладно на головних видах моделей.

Якщо модель відбудовується на основі реального об'єкта тієї самої фізичної природи, що й у оригінала, її називають *фізичною моделлю*.

Перевагою створення фізичної моделі є те, що, завдяки тому, що модель і оригінал підпорядковуються тим самим закономірностям, для створення моделі можна не знати конкретних математичних співвідношень і рівнянь, що описують поведінку оригіналу і моделі. Відповідність властивостей моделі і оригінала забезпечується автоматично.

Недоліком є необхідність:

- створення (проектування і виробництва) реального технічного об'єкта (моделі), хоча зазвичай більш простого, аніж оригінал;
- проведення експериментальних досліджень (випробувань) цього технічного об'єкта.

Це потребує значних витрат коштів, часу, кваліфікованих людських ресурсів, використання коштовного обладнання.

До фізичного моделювання вдаються, у головному, тоді, коли рівняння, що описують процес або явище, є невідомими або наближеними. Тому будується така модель, про яку можна заздалегідь сказати, що її рівняння такі самі, як й оригінала, причому не знаючи цих рівнянь. Це цілком можливо у випадку, коли модель є зменшеною (або збільшеною) копією оригінала або є тим самим оригіналом, але в інших умовах або з деякими зміненими масштабами.

Інший підхід – створювати матеріальну модель іншої фізичної природи, закономірності якої є аналогічними до закономірностей фізики оригіналу. Відповідна модель називається *аналоговою*. При цьому зазвичай найбільш зручним є обрання моделі електричної природи, бо електроелементи є найбільш доступними, невеликими за розміром, дешевими. За їх допомогою досить просто утворювати вельми складні системи з різноманітними властивостями. Досить просто і дешево забезпечити випробування електричних систем. Це складає го-

ловну перевагу аналогового моделювання. Неважко углядити і недоліки такого моделювання. У цьому випадку потрібно заздалегідь знати математичні співвідношення і рівняння поводження оригіналу, щоб забезпечити їх відворення при побудові аналогової моделі з використанням елементів іншої фізичної природи.

Найбільш дешевим і швидким є *математичне моделювання*, коли природа утвореної моделі є *знаковою*. *Математична* (теоретична) модель заміщує собою оригінал тільки в інформаційному сенсі, не маючи нічого спільного з оригіналом ані в матеріальній його природі, ані в енергетичному відношенні. Спільним є лише взаємозв'язок між реальними зв'язками між фізичними величинами оригінала, з одного боку, і математичними зв'язками між знаками, що відображують ці величини в математичній моделі – з іншого боку.

Суто знакова (теоретична) модель оперує лише знаками, що відповідають фізичним величинам оригіналу. Граничною задачею теоретичного моделювання є відшукування узагальненого розв'язку рівнянь, які подають математичну модель, не надаючи конкретні числові значення знакам в рівняннях, що відповідають незмінним величинам. Якщо вдається відшукати такий узагальнений теоретичний розв'язок, то стає можливим дослідити поводження оригіналу за довільних значень параметрів і, отже, обрати такі значення цих параметрів (якщо їх можна на практиці змінювати у широких межах), які забезпечували би найоптимальніше поводження оригіналу (технічного об'єкта). Тому, якщо вдається відшукати такий розв'язок, теоретичне моделювання є найбільш доцільним.

Але математика не є всесильною. Загальний теоретичний розв'язок вдається відшукати далеко не для усіх видів рівнянь. Наприклад, такий розв'язок мають лише лінійні звичайні диференціальні рівняння з постійними коефіцієнтами. Загальних методів розв'язування нелінійних диференціальних рівнянь і лінійних зі змінними коефіцієнтами наразі не існує. У цих випадках доцільніше відшукувати розв'язки рівнянь чисельними методами, застосовуючи обчислювальну техніку. Програмну модель, призначену для чисельного моделювання можна називати *чисельною математичною моделлю*. Саме про побудову таких моделей і проведення моделювання за ними йдеться далі.

1.1. Математика й реальність

Розповсюджений погляд, що математика – це специфічна мова. Ця думка має певне підґрунтя. Математика має усі ознаки мови. У зв'язку з цим постають деякі практичні питання, пов'язані із застосуванням математики у житті.

Завдяки певним рисам сучасного викладання математики у школі, іноді частина випускників сприймає математику як зібрання (зведення) деякої кількості правил, які мають до дійсності досить мале відношення, а у головному вигадані людьми – математиками. Це уявлення може бути досить стійким і підтримується в учнях завдяки тому, що головне полягання у викладанні математиці здійснюється часто не на задачі з життя, а на виконання математичних вправ, в яких головне – не відкрити для себе щось нове у оточуючому середо-

вищі, а міцно закріпити математичні правила оперування з математичними об'єктами. Це те саме, що при вивченні мови замість опанування змістом нових слів, вивчати лише правила граматичного поєднання слів у речення. Таке уявлення про математику глибоко хибне й шкідливе. Варто нагадати, що саме завдяки досягненням математики, людство спромоглося піднятися на сучасний рівень цивілізації.

Зазначимо, що будь-яка мова складається не лише з правил побудови слів та речень. Найважливішою складовою кожної мови є її змістозна частина, тобто ділянка реальної дійсності, яка описується за допомогою цієї мови. Без такої ділянки немає й самої мови. Без встановлення змістового зв'язку між словами мови й об'єктами дійсності, які вони позначають, немає сенсу і вести мову про мову. Власне мову й призначено задля відображення частини реальної дійсності, зберігання й передавання інформації про неї.

У математиці як мові є також ділянка дійсності, про яку математика розповідає. Наприклад, арифметика розмовляє з нами про деякі однорідні речі (предмети), надаючи можливість висновувати про їхні кількісні відношення і про їхнє змінювання при реальному оперуванні цими речами. Коли ми пишемо $m + n$, то розуміємо, що маємо купу з m однакових предметів і іншу купу з n таких предметів і додаємо предмети з другої купи до першої. При цьому неявно припускається, що *кожна річ із кожної купи існує окремо, незалежно від інших, має деяку стабільність (не змінюється з часом), займає деяку ділянку простору, може переміщуватися у просторі, не змінюючись, може приєднуватися до інших предметів, не змішуючись із ними*. І всі ці особливості не вигадані, вони взяті зі спостережень за реальними речами, наприклад, за стадами тварин тощо. Саме із реальної дійсності узята й сама операція додавання, яка математично узагальнює реальні дії по переміщенню окремих речей з одного місця у друге, де вже розміщено інші аналогічні речі. Саме із практики, завдяки простому перерахуванню, було встановлено, що $2+2=4$. Подібна операція зворотного напрямку (коли з купи речей відбираються окремі речі і переносяться у інше місце) була названа у математиці відніманням. А через те, що практично усі математичні дії походять з операції додавання як головної, то можна висновувати, що *уся математика спирається саме на описані властивості речей і дії з ними*.

Отже, практично усі *властивості математичних об'єктів узяті з реальної дійсності і лише децю узагальнені*. При цьому варто дати собі раду у тому, що математичні дії й оператори мають відношення зовсім не до будь-якої сфери дійсності, а лише до таких її частин, які мають вищезазначені властивості.

Перш за все до таких властивостей відноситься *існування фізичної операції додавання*, яка має такі властивості:

- асоціативності: $a + b + c = (a + b) + c = a + (b + c)$; *результат рахунку не залежить від того, у якому порядку здійснюється додавання; ця властивість має належати реальній операції додавання речей, які переліковуються;*
- комутативності: $a + b = b + a$; *результат додавання не залежить від того, до якої купи додаються речі з інших куп; ця властивість теж*

не є вигаданою, вона має належати реальній операції додавання речей;

- наявність нуля: є місце, а в ньому немає речей; і ця властивість має виконуватися у дійсних операціях із речами, що перераховуються;
- операція додавання має приводити до результату, який кількісно перевищує кожний з доданків.

Якщо хоча б одна із зазначених властивостей на практиці не властива фізичній операції додавання, до цих речей не можна прикладати математичні дії. А таких речей безліч у нашому оточенні.

Перш за все до них відносяться так звані якісні величини. Наприклад, важко уявити собі реальні операції з речами, внаслідок якої можна було б додати одна до одній гладкість, гіркоту, або твердість. Деякі з величин можна деяким чином вимірювати, наприклад, твердість матеріалів, або гладкість поверхонь. Але якщо для них неможливо вказати операції їхнього фізичного додавання, яка б мала усі зазначені властивості, такі кількісні величини називають екстенсивними. До них, наприклад, можна віднести таку фізичну величину, як температура, а також вищезгадані твердість і гладкість.

Кількісні (тобто такі, які можуть бути тим чи іншим способом виміряні) величини, для яких встановлено реальну (фізичну) операцію додавання, називають інтенсивними. До інтенсивних відноситься більшість фізичних величин. Переміщення, маса тіл, електричний струм, напруга, механічна напруга, сила, моменти сил, час – усе це приклади інтенсивних величин. Деякі з цих величин мають власну фізичну операцію додавання, інші – ні, але можуть бути подані як деякі прості функції від тих величин, що мають таку операцію.

Наприклад, операцією додавання для довжини (або переміщення) є така, коли початок другої із двох довжин сполучається з кінцем першої. Результатом при цьому вважається довжина від початку першої довжини до кінця другої. Неважко впевнитися, що за умови розташування довжин вздовж однієї прямої в одному напрямку така операція матиме усі ознаки операції додавання. У випадку просторового переміщення (або довільного розташування довжин у просторі) аналогічна операція є слушною по відношенню до будь-яких трьох ортогональних напрямків. У цілому в результаті одержуємо правило векторного додавання переміщень у просторі.

Для часу операція додавання може виглядати наступним чином: початок другого процесу сполучається з кінцем першого. Результатом є тривалість від початку першого процесу до кінця другого.

Додавання мас збігається з операцією жорсткого з'єднання мас в одну масу.

Додавання електричних зарядів полягає у об'єднанні зарядів при дотикуванні заряджених тіл.

Величини, що є похідними від тих, що мають операцію додавання, також є інтенсивними. Наприклад, інтенсивною величиною є швидкість, яка визначається як результат ділення переміщення на проміжок часу, протягом якого це переміщення здійснюється, а також прискорення матеріальної точки. Анало-

гічно, електричний струм, що визначається як відношення приросту електричного заряду до проміжку часу, за який цей приріст відбувся, також є інтенсивною величиною.

Виходячи з того, що усі арифметичні дії (математичні операції віднімання, множення, ділення, піднесення до степеня, взяття похідної та інтегрування) є похідними від операції додавання, можна зробити висновок, що у повній мірі математичні висновки торкаються лише інтенсивних величин. Лише по відношенню до цих величин можна застосовувати усі здобутки математики як мови.

Математика (принаймні, це стосується диференціального й інтегрального зчислень, теорії диференціальних і інтегральних рівнянь) – це мова про інтенсивні величини, тобто, повторимо, про величини, які, з одного боку, є вимірюваними (кількісними), а, з іншого боку, мають реальну фізичну операцію додавання.

1.2. Теорія подібності і аналіз розмірностей

Побудова моделі (особливо фізичної і аналогової) потребує виконання наступних попередніх дій.

По-перше, необхідно уточнити задачу моделювання, тобто чітко визначити, які саме властивості оригіналу і в яких нових умовах потребують вивчення.

Далі слід виявити які саме фізичні величини і які параметри описують досліджувані властивості, і які фізичні величини та їхні параметри можуть *суттєво* впливати на ці властивості (до них обов'язково слід додати ті величини, вивчення впливу яких на поведінку оригіналу потрібно вивчити). Усі інші величини і параметри, що характеризують оригінал, утворять множину характеристик оригіналу, які не впливають на досліджування і не входять у сферу цілей моделювання. Майбутню модель тепер слід будувати, встановлюючи останні величини з міркування, щоб модель була якомога більш доступною, простою і зручною для дослідження. При цьому потрібно слідкувати, щоб *суттєві* характеристики оригіналу зберігалися у моделі.

Виконання усіх перелічених дій ще не гарантує створення моделі, оскільки невідомо, які значення мають одержати суттєві параметри моделі щоб результати моделювання можна було б перерахувати у параметри поведінки оригіналу без суттєвих похибок. Науково-обрунтовані методи визначення суттєвих параметрів моделі за заданими суттєвими параметрами оригіналу і переносу результатів моделювання на оригінал встановлюються *теорією подібності*.

1.2.1. Подібність явищ і її ознаки

При протіканні будь-якого досліджуваного процесу змінюються деякі величини, що характеризують стан системи з досліджуваного боку. Ці величини називаються *параметрами процесу*. Система, в якій здійснюються процеси,

складається з елементів, які характеризуються власними (зазвичай сталими) параметрами, які називають *параметрами системи*.

При досліджуванні механічних процесів до параметрів процесу відносяться величини сил, переміщень, швидкостей, прискорень, а до параметрів системи – маси тіл, моменти інерції, коефіцієнти тертя, жорсткості, демпфірування в'язкості і т.п. У випадку електричних систем параметрами процесу є величини напруг, струмів, електричних зарядів, а параметрами системи – величини індуктивностей, опорів, ємностей тощо.

Параметри системи або можна вважати незмінними протягом усього досліджуваного процесу, або можна враховувати їхнє змінювання у просторі і часі.

Якщо параметри системи змінюються при змінюванні одного чи кількох досліджуваних параметрів процесу (координат), то вони називаються нелінійними і, відповідно, система називається нелінійною,

Явище складається з сукупності процесів, що описуються рівняннями, які пов'язують параметри процесу з параметрами системи, записаними в обраній системі відліку. Система головних рівнянь, що визначають певне явище, математично описує механізм цілого класу фізичних явищ. Щоб з нескінченної множини явищ цього класу виділити поодинокі конкретне явище, тобто щоб одержати частковий розв'язок даної системи диференціальних рівнянь, необхідно задати відповідні додаткові умови, так звані умови однозначності. Умови однозначності – це ті умови, якими визначаються індивідуальні відмінності явищ певного класу. Вони встановлюються незалежно від механізму процесу, але суттєво впливають на перебіг явищ.

До складу умов однозначності відносяться:

- 1) фізичні *параметри середовища*, що є суттєвими для плину процесу;
- 2) *початкові (часові) умови* (тільки для нестационарних процесів), тобто поля усіх змінних у початковий момент часу;
- 3) *граничні умови*, тобто умови взаємодії з оточуючим середовищем (зокрема, поля змінних на межах системи).

Сукупність системи основних рівнянь і умов однозначності принципово є достатньою для одержання часткового розв'язку, який визначає досліджуване конкретне явище. Однак точний аналітичний розв'язок фактично може бути одержаний тільки у виняткових випадках. Набагато більш продуктивним є експериментальне досліджування поодиноких явищ з наступним по можливості більш широким узагальненням одержаних часткових результатів. Для цього потрібно мати науково обгрунтовані методи такого узагальнення.

Вченням про науково-обгрунтовані методи узагальнення результатів поодиноких експериментів і є теорія подібності.

1.2.2. Основні положення теорії подібності

Головне завдання теорії подібності – виділення серед кожного класу явищ таких груп, у межах яких можливо узагальнення результатів поодиноких

експериментів. Виконання цього завдання показує, що науково-обґрунтоване узагальнення результатів поодиноких експериментів можливо лише у межах групи подібних явищ.

1. *Подібними* називаються фізичні явища, в яких є подібними, тобто *відповідно пропорційними, усі характерні величини*

$$\frac{R_{oi}}{R_{mi}} = k_R = \text{const}.$$

Тут позначено: R_{oi} – деяка характерна скалярна або векторна фізична величина i -ої точки оригіналу; R_{mi} – відповідна характерна фізична величина відповідної i -тої точки моделі.

При цьому для подібності векторів необхідною є також подібність їхніх компонент по осях систем відліку (тобто паралельність векторів):

$$\frac{R_{oiX}}{R_{miX}} = \frac{R_{oiY}}{R_{miY}} = \frac{R_{oiZ}}{R_{miZ}} = k_R = \text{const}.$$

2. *Необхідними і достатніми умовами подібності фізичних явищ є подібність умов однозначності при тотожності (інваріантності) систем основних диференціальних рівнянь* (теорема М. В. Кирпичова і А. А. Гухмана).

Отже, необхідними і достатніми умовами подібності є:

- а) фізична подібність – подібність усіх фізичних параметрів середовища;
- б) подібність полів усіх змінних у початковий момент часу процесу (тільки для нестационарних процесів);
- в) подібність умов на межах системи;
- г) інваріантність системи основних диференціальних рівнянь по відношенню до подібного перетворення змінних, тобто сумісне виконання рівнянь

$$F(u_1, u_2, \dots, u_n) = 0 \quad \text{та} \quad F(c_1 u_1, c_2 u_2, \dots, c_n u_n) = 0.$$

де u_1, u_2, \dots, u_n – змінні; c_1, c_2, \dots, c_n – множники подібного перетворення.

3. *Необхідною же і достатньою ознакою інваріантності системи основних диференціальних рівнянь по відношенню до подібного перетворення є інваріантність (рівність для відповідних точок полів і відповідних моментів часу) деяких безрозмірних (не маючих фізичної розмірності) степеневих комплексів з величин, що характеризують поведінку оригіналу і моделі. Ці безрозмірні степеневі комплекси прислужуються у якості кількісних ознак подібності фізичних явищ і тому називаються критеріями подібності.*
4. Для забезпечення подібності фізичних явищ (окрім подібності умов однозначності) достатньо рівності тільки *критеріїв, побудованих з величин, що входять в умови однозначності* (зазвичай, – параметрів системи і параметрів зовнішніх дій). Тому такі критерії називають *визна-*

чальними. Рівність решти критеріїв подібності є наслідком подібності фізичних явищ. Тому решта критеріїв називаються *визначуваними*.

5. Уточнене формулювання головної теореми теорії подібності: *необхідними і достатніми умовами подібності фізичних явищ є рівність визначальних критеріїв подібності при наявності подібності умов однозначності*.

1.2.3. Аналіз розмірностей

Як випливає з теорії подібності, для науково-обґрунтованої побудови моделі потрібно попередньо відшукати критерії подібності, тобто безрозмірні степеневі комплекси з фізичних величин, що характеризують поведінку досліджуваної системи і моделі. Основним методом відшукування критеріїв подібності є аналіз розмірностей. Він дозволяє одержати критерії подібності навіть у тому випадку, коли рівняння, що описують поведінку системи, невідомі, відомі лише усі параметри, які однозначно визначають досліджуваний процес, і фізичні розмірності цих параметрів. Таке знання дається зазвичай досвідом і експериментом.

Відшукування критеріїв подібності методом аналізу розмірностей зводиться до наступної послідовності дій:

- 1) обираються необхідні одиниці вимірювання величин, що є суттєвими для опису досліджуваного процесу; наприклад, для механічних систем такими одиницями можуть бути три – T – одиниця виміру часу, L – одиниця виміру переміщень і M – одиниця виміру маси; для електричних систем можна обрати такі основні одиниці: T – одиниця виміру часу, Q – одиниця виміру електричного заряду і V – одиниця виміру електричної напруги;
- 2) розмірності усіх параметрів системи, включаючи змінні і умови однозначності, виражаються через обрані одиниці виміру;
- 3) з числа постійних параметрів системи і зовнішніх сил обираються кілька з незалежними розмірностями (тобто з такими розмірностями, щоб жодну з них не можна було б подати у вигляді степеневого комплексу, складеного з розмірностей решти); очевидно, кількість таких *базових параметрів* буде дорівнювати або меншим за кількість основних одиниць виміру (3 – для механічних систем, 3 – для електричних систем);
- 4) відшукуються безрозмірні степеневі комплекси (критерії подібності) у вигляді

$$\frac{R_i}{A^{\alpha_i} B^{\beta_i} C^{\gamma_i}}$$

де R_i – деякий i -й параметр системи, який не входить до числа базових; A , B і C – базові параметри; α_i , β_i , γ_i – показники степенів, в які

потрібно піднести відповідно розмірності параметрів A , B і C , щоб у знаменнику одержати розмірність параметра R_i ;

- 5) далі, прирівнюючи показники при однакових одиницях виміру у чисельнику і знаменнику степеневому комплексу, одержують значення показників степенів $\alpha_i, \beta_i, \gamma_i$, за яких розглядуваний степеневий комплекс стає безрозмірною величиною.

1.2.4. Приклади застосування аналізу розмірностей.

Розглянемо коливальну механічну систему, яка складається з масивного тіла масою m , яке може переміщуватися вдовж горизонтальної осі X і з'єднане з нерухомою основою за допомогою пружини з жорсткістю c і демпфера з коефіцієнтом в'язкого тертя f . На тіло діє гармонічно змінювана у часі сила, амплітуда змінювання якої дорівнює F_m , частота змінювання ω , а початкова фаза $-\varepsilon$. Потрібно відшукати критерії подібності такої системи, враховуючи перехідний режим.

До числа параметрів системи у цьому випадку відносяться: m – маса тіла, c – жорсткість пружини і f – коефіцієнт тертя.

Умови однозначності складаються з амплітуди сили F_m , частоти її змінювання ω , початкової фази ε і початкових умов: x_0 – початкового відхилення тіла від положення рівноваги і \dot{x}_0 – початкової швидкості.

До параметрів процесу можна віднести: поточний час t протікання процесу, поточне значення x відхилення тіла від положення рівноваги, поточне значення \dot{x} швидкості тіла, поточне значення \ddot{x} прискорення тіла.

Оберемо у якості основних одиниць виміру T одиницю виміру часу, L – одиницю виміру переміщень і M – одиницю виміру маси.

Виразимо розмірності усіх вищезгаданих величин через ці одиниці. Матимемо:

$$\{m\} = M; \quad \{t\} = T; \quad \{x\} = \{x_0\} = L; \quad \{\dot{x}\} = \{\dot{x}_0\} = LT^{-1}; \quad \{\ddot{x}\} = LT^{-2};$$

$$\{F_m\} = \{m\ddot{x}\} = \{m\}\{\ddot{x}\} = MLT^{-2}; \quad \{c\} = \frac{\{F_m\}}{\{x\}} = MT^{-2}; \quad \{f\} = \frac{\{F_m\}}{\{\dot{x}\}} = MT^{-1}.$$

Оберемо як базові три параметри:

m – масу тіла;

c – жорсткість пружини;

F_m – амплітуду змінювання зовнішньої сили.

Неважно впевнитися, що розмірності цих величин є взаємозалежними.

Тоді безрозмірні степеневі комплекси слід відшукувати у формі:

$$\bar{R}_i = \frac{R_i}{m^{\alpha_i} c^{\beta_i} F_m^{\gamma_i}}.$$

Запишемо розмірність цього степеневого комплексу:

$$\left\{ \frac{R_i}{m^{\alpha_i} c^{\beta_i} F_m^{\gamma_i}} \right\} = \frac{\{R_i\}}{\{m\}^{\alpha_i} \{c\}^{\beta_i} \{F_m\}^{\gamma_i}} = \frac{\{R_i\}}{M^{\alpha_i} (MT^{-2})^{\beta_i} (MLT^{-2})^{\gamma_i}} =$$

$$= \frac{M^{r_M} T^{r_T} L^{r_L}}{M^{\alpha_i} (MT^{-2})^{\beta_i} (MLT^{-2})^{\gamma_i}} = \frac{M^{r_M} T^{r_T} L^{r_L}}{M^{(\alpha_i + \beta_i + \gamma_i)} T^{-2(\beta_i + \gamma_i)} L^{\gamma_i}}.$$

Тут позначено r_M, r_T, r_L – показники степеня у розмірності параметра R_i , у які підносяться відповідно одиниці маси, часу і переміщення.

Для забезпечення безрозмірності степеневого комплексу потрібно розв'язати систему лінійних алгебричних рівнянь

$$\begin{cases} \alpha_i + \beta_i + \gamma_i = r_M \\ -2(\beta_i + \gamma_i) = r_T \\ \gamma_i = r_L \end{cases}$$

і визначити у такий спосіб шукані показники $\alpha_i, \beta_i, \gamma_i$.

Неважко впевнитися, що застосування цієї процедури до базових параметрів приводить до того, що відповідні степеневі комплекси дорівнюватимуть одиниці:

$$\bar{m} = 1; \quad \bar{c} = 1; \quad \bar{F}_m = 1.$$

Здійснивши вищевказані дії по відношенню до решти параметрів, одержимо наступні критерії подібності:

$$\bar{t} = t \sqrt{\frac{c}{m}}; \quad \bar{x} = x \frac{c}{F_m}; \quad \bar{\dot{x}} = \dot{x} \frac{\sqrt{cm}}{F_m}; \quad \bar{\ddot{x}} = \ddot{x} \frac{m}{F_m}; \quad \bar{x}_0 = x_0 \frac{c}{F_m};$$

$$\bar{\dot{x}}_0 = \dot{x}_0 \frac{\sqrt{cm}}{F_m}; \quad \bar{\omega} = \omega \sqrt{\frac{m}{c}}; \quad \bar{f} = \frac{f}{\sqrt{mc}}.$$

Якщо рівняння руху досліджуваної системи має вигляд

$$m\ddot{x} + f\dot{x} + cx = F_m \sin(\omega t + \varepsilon)$$

і якщо ввести позначення

$$\omega_0 = \sqrt{\frac{c}{m}}; \quad \nu = \frac{\omega}{\omega_0}; \quad \zeta = \frac{f}{2\sqrt{mc}}; \quad \tau = \omega_0 t; \quad \bar{x}' = \frac{d\bar{x}}{d\tau}; \quad \bar{x}'' = \frac{d^2\bar{x}}{d\tau^2},$$

то це рівняння в безрозмірному вигляді набуде форми

$$\bar{x}'' + 2\zeta\bar{x}' + \bar{x} = \sin(\nu\tau + \varepsilon). \quad (a)$$

Як бачимо, рівняння у безрозмірному вигляді значно простіше. Його визначають лише три числові параметри – ζ, ν, ε , замість шести у початковому рівнянні.

Здійснимо аналогічні процедури по відношенню до фізичного маятника, момент інерції якого відносно його осі підвісу дорівнює J , опорний маятниковий момент – mgl (m – маса маятника, g – прискорення сили тяжіння, l – зміщення центру мас відносно осі обертання маятника), R – коефіцієнт в'язкого тертя у підшипниках осі обертання маятника. На маятник навколо осі маятника

діє момент сил, який змінюється з часом за гармонічним законом з частотою ω , амплітудою M_m і початковою фазою ε .

У цьому випадку параметрами системи є величини J, R, mgl , параметрами процесу – величини поточного кута φ відхилення маятника від вертикалі, кутової швидкості $\dot{\varphi}$ і кутового прискорення $\ddot{\varphi}$. До умов однозначності слід віднести $M_m, \omega, \varepsilon, \varphi_0$ і $\dot{\varphi}_0$.

Оберемо ті самі основні одиниці виміру: T – часу, L – переміщення і M – маси. Тепер можна виразити розмірності усіх вищевказаних параметрів у такий спосіб:

φ, ε – безрозмірні величини за визначенням;

$$\{J\} = ML^2; \quad \{mgl\} = ML^2T^{-2}; \quad \{R\} = ML^2T^{-1}; \quad \{t\} = T;$$

$$\{\dot{\varphi}\} = \{\dot{\varphi}_0\} = T^{-1}; \quad \{\ddot{\varphi}\} = T^{-2}; \quad \{M_m\} = ML^2T^{-2}; \quad \{\omega\} = T^{-1}.$$

Тут вже неможливо обрати три базові параметри, а лише два. Оберемо базовими параметрами момент інерції J і опорний момент mgl . Тоді відповідні критерії подібності дорівнюватимуть:

$$\bar{J} = 1; \quad \bar{mgl} = 1; \quad \tau = \bar{t} = t \sqrt{\frac{mgl}{J}}; \quad \bar{R} = \frac{R}{\sqrt{mgl \cdot J}}; \quad \bar{\omega} = \omega \sqrt{\frac{J}{mgl}};$$

$$\bar{\varphi} = \varphi' = \varphi \sqrt{\frac{J}{mgl}}; \quad \bar{\varphi}_0 = \varphi'_0 = \varphi_0 \sqrt{\frac{J}{mgl}}; \quad \bar{\ddot{\varphi}} = \ddot{\varphi}'' = \ddot{\varphi} \frac{J}{mgl}; \quad \bar{M}_m = \frac{M_m}{mgl}.$$

Введемо позначення:

$$\omega_0 = \sqrt{\frac{mgl}{J}}; \quad \tau = \omega_0 t; \quad \varphi' = \frac{d\varphi}{d\tau}; \quad \varphi'' = \frac{d^2\varphi}{d\tau^2}; \quad v = \frac{\omega}{\omega_0}; \quad \zeta = \frac{\bar{R}}{2} = \frac{R}{2\sqrt{mgl \cdot J}}.$$

Тоді, якщо рівняння руху фізичного маятника має вигляд

$$J \cdot \ddot{\varphi} + R \cdot \dot{\varphi} + mgl \cdot \varphi = M_m \sin(\omega \cdot t + \varepsilon),$$

то відповідне рівняння у безрозмірній формі набуде вигляду

$$\varphi'' + 2\zeta\varphi' + \varphi = \bar{M}_m \sin(v\tau + \varepsilon). \quad (6)$$

Розглянемо електричний коливальний контур, який складається з послідовно з'єднаних соленоїда з індуктивністю L , резистора з опором R , конденсатора з ємністю C і джерела електрорушійної сили (е.р.с.), який виробляє гармонічно змінювану е.р.с. з амплітудою E_m , частотою ω і початковою фазою ε .

Параметрами процесу у цій системі є поточне значення заряду q на конденсаторі, електричний струм $i = \frac{dq}{dt}$ у контурі і напруги на індуктивності, резисторі і конденсаторі.

У цьому випадку оберемо такі головні одиниці виміру: T – одиниця виміру часу, Q – одиниця виміру електричного заряду і V – одиниця виміру електричної напруги. Тоді розмірності визначальних параметрів можна подати у такий спосіб:

$$\{t\} = T; \quad \{\omega\} = T^{-1}; \quad \{E_m\} = V; \quad \{q\} = \{q_0\} = Q; \quad \{i\} = \{i_0\} = QT^{-1};$$

$$\left\{ \frac{di}{dt} \right\} = QT^{-2}; \quad \{L\} = VQ^{-1}T^2; \quad \{R\} = VQ^{-1}T; \quad \{C\} = V^{-1}Q.$$

Оберемо у якості базових параметрів індуктивність L , ємність C і амплітуду е.р.с. E_m . Проводячи відшукування критеріїв подібності, дійдемо:

$$\bar{L} = 1; \quad \bar{C} = 1; \quad \bar{E}_m = 1; \quad \bar{t} = \frac{t}{\sqrt{L \cdot C}}; \quad \bar{\omega} = \omega \sqrt{L \cdot C};$$

$$\bar{q} = \frac{q}{CE_m}; \quad \bar{i} = \frac{i}{E_m} \sqrt{\frac{L}{C}}; \quad \bar{q}_0 = \frac{q_0}{CE_m}; \quad \bar{i}_0 = \frac{i_0}{E_m} \sqrt{\frac{L}{C}}; \quad \bar{R} = R \sqrt{\frac{C}{L}}.$$

Використовуючи позначення:

$$\omega_0 = \frac{1}{\sqrt{LC}}; \quad \tau = \omega_0 t; \quad \bar{q}' = \frac{d\bar{q}}{d\tau}; \quad \bar{q}'' = \frac{d^2\bar{q}}{d\tau^2}; \quad \nu = \frac{\omega}{\omega_0}; \quad \zeta = \frac{\bar{R}}{2} = \frac{R}{2} \sqrt{\frac{C}{L}},$$

можна рівняння електричного коливального контуру

$$L \frac{di}{dt} + R \cdot i + \frac{1}{C} q = E_m \sin(\omega \cdot t + \varepsilon)$$

подати у безрозмірному вигляді у такий спосіб:

$$\bar{q}'' + 2\zeta \cdot \bar{q}' + \bar{q} = \sin(\nu \cdot \tau + \varepsilon). \quad (\text{в})$$

Порівнюючи рівняння (а, б, в), можна зробити деякі попередні висновки:

- 1) рівняння руху деяких матеріальних систем, навіть різної фізичної природи, у безрозмірній формі можуть набувати практично ідентичного вигляду;
- 2) завдяки цьому виникають можливості:

– розповсюджувати результати поодинокого експерименту на поведження цілого класу подібних явищ або тієї самої фізичної природи, або навіть зовсім іншої фізичної природи;

– проводити дослідження поведження оригіналу на основі експериментів з моделями або тієї самої фізичної природи, або іншої фізичної природи.

1.2.5. Застосування теорії подібності для побудови моделей.

Подання параметрів досліджуваного процесу у вигляді безрозмірних критеріїв подібності надає широкі можливості у побудові моделей цього процесу. Наприклад, досліджування поведження механічної коливальної ланки, що описується рівнянням (а) можна здійснити:

- 1) на основі моделі того самого типу (маси з пружно-демпфіруючим зв'язком з основою), але з зовсім іншими (більш доступними для дослідника) характеристиками маси, жорсткості пружини і амплітуди зовнішньої сили; до такого моделювання вдаються, коли потрібно вивчити поведження об'єкта або надто великих, або надто малих розмірів, виготовлення і експериментування з якими потребує великих втрат, чи взагалі неможливе;

2) на основі електричної моделі, що описується рівнянням (в); у цьому випадку замість виготовлення і проведення експериментів з механічною моделлю, що є досить складною справою, достатньо зібрати електричний контур з досить великого асортименту індуктивностей, резисторів і конденсаторів, підключити його до джерела змінної напруги і, проводячи експеримент, вимірювати струм і напруги за допомогою також досить великого і досяжного набору електричних вимірювачів (амперметрів і вольтметрів).

Розглянемо більш докладно побудову моделі, моделювання і розповсюдження результатів моделювання на поводження оригіналу у цих двох випадках.

Отже, нехай потрібно дослідити поводження механічної коливальної системи (оригіналу) з такими заданими значеннями її характеристик – $m_O, f_O, c_O, F_{mO}, \omega_O, \varepsilon_O, x_{0O}, \dot{x}_{0O}$, де індекс O позначає, що відповідний параметр відноситься до оригіналу.

Припустимо, що потрібно провести ці дослідження на механічному аналозі цієї системи. З того, що у якості базових параметрів при відшуканні критеріїв подібності обрані величини маси, жорсткості пружини і амплітуди діючої зовнішньої сили, впливає, що вказані величини у моделі можуть бути обрані дослідником *довільними*, виходячи з умов доступності, простоти експериментування, дешевизни тощо. Позначимо обрані значення цих величин через m_m, c_m, F_{mm} , де індекс "м" позначає, що відповідні параметри характеризують модель.

Щоб забезпечити подібність явищ в моделі і оригіналі, достатньо обрати решту параметрів моделі з умови рівності критеріїв подібності, що відповідають умовам однозначності. У розглядуваному випадку цих критеріїв чотири – $\zeta, v, \bar{x}_0, \bar{\dot{x}}_0$. Отже, має бути забезпечено виконання наступних співвідношень між параметрами оригіналу і моделі:

$$\zeta_O = \frac{f_O}{2\sqrt{m_O c_O}} = \zeta_m = \frac{f_m}{2\sqrt{m_m c_m}}; \quad \bar{\omega}_O = \omega_O \sqrt{\frac{m_O}{c_O}} = \bar{\omega}_m = \omega_m \sqrt{\frac{m_m}{c_m}};$$

$$\bar{x}_{0O} = x_{0O} \frac{c_O}{F_{mO}} = \bar{x}_{0m} = x_{0m} \frac{c_m}{F_{mm}}; \quad \bar{\dot{x}}_{0O} = \dot{x}_{0O} \frac{\sqrt{c_O m_O}}{F_{mO}} = \bar{\dot{x}}_{0m} = \dot{x}_{0m} \frac{\sqrt{c_m m_m}}{F_{mm}}.$$

Це означає, що коефіцієнт тертя, частота коливань діючої сили і початкові значення відхилення і швидкості тепер не можуть бути довільними, а мають дорівнювати значенням, підрахованим з попередніх співвідношень на основі заданих значень параметрів оригіналу і обраних значень параметрів моделі:

$$f_m = f_O \sqrt{\frac{m_m}{m_O}} \sqrt{\frac{c_m}{c_O}}; \quad \omega_m = \omega_O \sqrt{\frac{m_O}{m_m}} \sqrt{\frac{c_m}{c_O}}; \quad x_{0m} = x_{0O} \frac{F_{mm} c_O}{F_{mO} c_m}; \quad \dot{x}_{0m} = \dot{x}_{0O} \sqrt{\frac{m_O}{m_m}} \sqrt{\frac{c_O}{c_m}} \frac{F_{mm}}{F_{mO}}.$$

Після встановлення розрахункових значень у моделі, можна проводити експериментальне її дослідження. Вимірюючи поточне переміщення, час і швидкість тіла у моделі і фіксуючи результати вимірювань, одержимо потрібні залежності $x_m(t_m), \dot{x}_m(t_m)$.

Далі потрібно розповсюдити ці залежності на шукані аналогічні залежності в оригіналі. Це здійснюється також на основі рівності відповідних критеріїв подібності

$$x_O(t) = x_M(t_M) \frac{c_M F_{mO}}{c_O F_{mM}}; \quad \dot{x}_O(t) = \dot{x}_M(t_M) \sqrt{\frac{m_M}{m_O}} \sqrt{\frac{c_M}{c_O}} \frac{F_{mO}}{F_{mM}},$$

причому зв'язок між часом t оригіналу і часом t_M моделі (модельним часом) визначається співвідношенням

$$t = t_M \sqrt{\frac{c_M}{c_O}} \sqrt{\frac{m_O}{m_M}}.$$

Цікавою є така обставина. Модельний час можна зробити значно більш швидкоплинним у порівнянні з часом оригіналу, за рахунок, наприклад, обрання таких параметрів моделі, щоб виконувалася нерівність

$$\sqrt{\frac{m_O}{m_M}} \sqrt{\frac{c_M}{c_O}} \gg 1.$$

Таке моделювання забезпечує значне скорочення часу дослідження. Цим часто користуються на практиці, наприклад, при моделюванні старіння будівельних конструкцій, аналізі поведінки насипів протягом значного часу (десятки років) тощо. Таке моделювання називають іноді "лупою часу".

Тепер перейдемо до аналогового моделювання тієї самої механічної системи на електричній моделі.

З того, що базовими параметрами електричного аналога є індуктивність, ємність конденсатора і амплітуда змінювання е.р.с., ці параметри дослідник може обрати довільно. Позначимо обрані значення через L_M , C_M і E_{mM} .

Як і у попередньому випадку для визначення решти параметрів електричного аналога у відповідності до теорії подібності слід забезпечити рівність визначальних критеріїв подібності:

$$\zeta_O = \frac{f_O}{2\sqrt{m_O c_O}} = \zeta_M = \frac{R_M}{2} \sqrt{\frac{C_M}{L_M}}; \quad \bar{\omega}_O = \omega_O \sqrt{\frac{m_O}{c_O}} = \bar{\omega}_M = \omega_M \sqrt{L_M \cdot C_M};$$

$$\bar{x}_{0O} = x_{0O} \frac{c_O}{F_{mO}} = \bar{q}_{0M} = \frac{q_{0M}}{C_M E_{mM}}; \quad \bar{\dot{x}}_{0O} = \dot{x}_{0O} \frac{\sqrt{c_O m_O}}{F_{mO}} = \bar{i}_{0M} = \frac{i_{0M}}{E_{mM}} \sqrt{\frac{L_M}{C_M}}.$$

З них випливає, що величини індуктивності L_M і ємності C_M моделі можуть обрати довільними, а величина опору резистора моделі має визначатися з

співвідношення $R_M = \frac{f_O}{\sqrt{m_O c_O}} \sqrt{\frac{L_M}{C_M}}$, величина частоти змінювання е.р.с. – з рів-

ності $\omega_M = \frac{\omega_O}{\sqrt{L_M \cdot C_M}} \sqrt{\frac{m_O}{c_O}}$. При експерименті на моделі слід забезпечити почат-

кову величину заряду на конденсаторі $q_{0M} = x_{0O} \frac{c_O}{F_{mO}} C_M E_{mM}$ і початкове зна-

чення струму у контурі $i_{0,m} = \dot{x}_{0O} \frac{\sqrt{c_O m_O}}{F_{mO}} E_{mm} \sqrt{\frac{C_m}{L_m}}$. Одержані внаслідок моделювання залежності величини струму в контурі від модельного часу можна перевести у залежність швидкості тіла за формулою

$$\dot{x}_O(t) = i_m(t_m) \frac{F_{mO}}{\sqrt{c_O m_O}} \frac{1}{E_{mm}} \sqrt{\frac{L_m}{C_m}}.$$

При цьому модельний час пов'язаний з часом оригіналу у такий спосіб:

$$t = \frac{t_m}{\sqrt{L_m \cdot C_m}} \sqrt{\frac{m_O}{c_O}}.$$

Примітки.

1. Звичайно, у розглянутих прикладах матеріальне моделювання не має сенсу, бо наведені рівняння досить просто розв'язуються теоретично. Але якщо рівняння, що описують поведінку оригіналу, є нелінійними і надто складними, моделювання може стати необхідним. Наведені приклади призначені лише для ілюстрації механізму застосування апарату теорії подібності для побудови моделей і розповсюдження результатів моделювання на оригінал.

2. Проведені операції з забезпечення подібності оригіналу і моделі і перерахунку результатів моделювання ґрунтуються на застосуванні аналізу розмірностей і не спираються на знання диференціальних рівнянь руху. Тому забезпечення подібності можливо навіть тоді, коли рівняння руху взагалі є невідомими. Достатньо лише чітко визначити сукупність фізичних параметрів, які суттєво впливають на досліджуване явище, і встановити їхню фізичну розмірність.

1.3. Запитання для самоперевірки

1. Що таке модель?
6. Що розуміють під моделюванням?
3. Що називають параметрами процесу? параметрами системи? умовами однозначності?
4. Що таке критерії подібності?
5. Що називають базовими параметрами в аналізі розмірностей і чому дорівнюють їх безрозмірні аналоги у безрозмірних рівняннях?
6. Які умови слід витримати, щоб модель і оригінал були подібними?

2. ЕТАПИ РОЗВ'ЯЗУВАННЯ ІНЖЕНЕРНИХ ЗАДАЧ НА ЕОМ

Одним з головних завдань інженера є проектування нового технічного об'єкта.

Проектування поділяється на стадії науково–дослідницьких робіт (НДР), дослідно-конструкторських робіт (ДКР), ескізного проекту, технічного проекту, робочого проекту, випробувань дослідного зразка. На етапі ескізного проектування здійснюється детальне опрацювання можливості побудови проектованої системи, його результатом є ескізний проект. На етапі технічного проектування виконується укрупнені подання всіх конструкторських і технологічних рішень; результатом цього етапу є технічний проект. На етапі робочого проектування проводиться детальне опрацювання всіх блоків, вузлів і деталей проектованої системи, а також технологічних процесів виробництва деталей і їх збирання у вузли і блоки. Заключний етап – виготовлення дослідного зразка, за результатами випробувань якого вносять необхідні зміни в проектну документацію.

Фактично жоден з етапів проектування не обходиться без того чи іншого виду моделювання поведінки тих чи інших частин створюваної системи і системи у цілому. Зазвичай, на початкових стадіях проектування обмежуються переважно знаковим моделюванням (перш за все – математичним), або аналоговим. На наступних стадіях застосовують і фізичне моделювання до окремих деталей, частин системи. Проектування завершується головним фізичним моделюванням – натурним випробуванням дослідного зразка системи.

Результатом розв'язування інженерних (прикладних) задач будь-якого рівня є, як правило, чисельні оцінки (параметрів пристроїв, процесів, технічних і економічних характеристик, тощо), які є наслідком розрахунків, що здійснюються з наближеними первинними даними. *Більшість прикладних задач зводяться до математичних задач, які розв'язуються різноманітними обчислювальними методами.*

Послідовність розв'язування таких задач можна подати у вигляді таких етапів, наведених на рис. 2.1:

- 1) постановка задачі;
- 2) створення математичної моделі (формулювання задачі); перевірка моделі на адекватність;
- 3) побудова розрахункової (обчислювальної) моделі, яка відповідає прийнятій математичній моделі;
- 4) проведення розрахунків за обраною обчислювальною моделлю при заданих (відомих) значеннях первісних даних;
- 5) аналіз одержаних результатів.

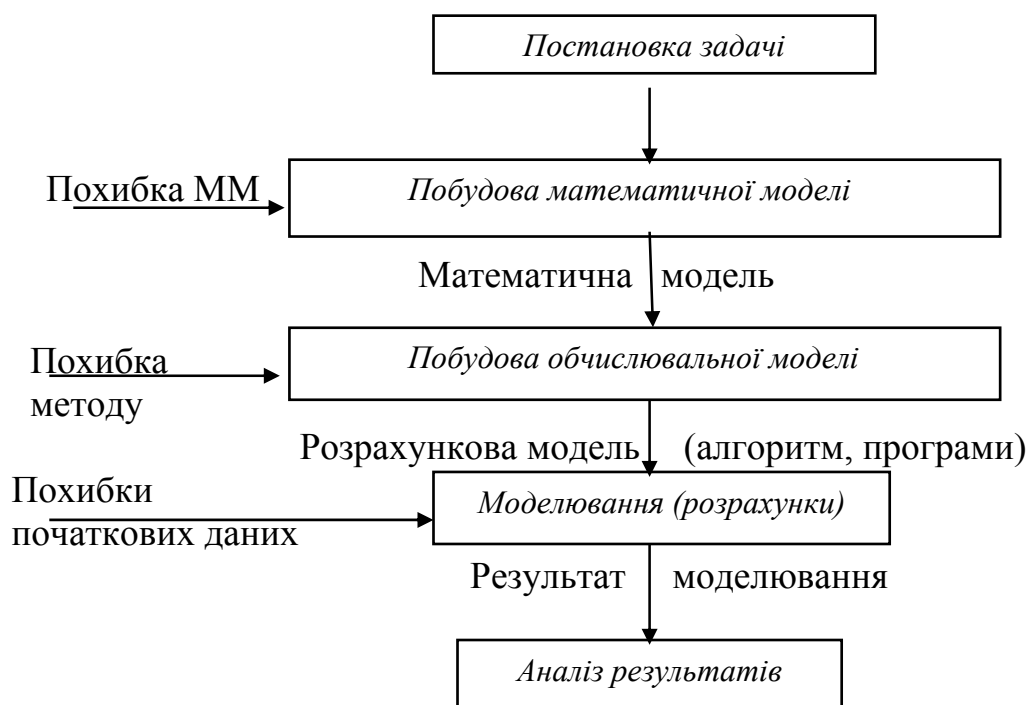


Рис. 2.1. Схема розв'язування інженерної задачі

Розглянемо докладніше кожний з цих етапів.

2.1. Постановка задачі

Постановка задачі має передумовою словесне, змістовне формулювання задачі, умов, за яких вона ставиться, та вимог до її розв'язування. Слова "змістовне формулювання" слід розуміти так, що задача має бути сформульована у термінах опису реального об'єкта (технічного пристрою або процесу), поведіння якого підлягає вивченню.

Як приклади розглядатимемо такі найпростіші інженерні задачі.

Задача 1. Визначити характеристики власного руху фізичного маятника за умови малих його коливань.

Задача 2. Визначити змінювання швидкості тіла при його падінні, враховуючи опір оточуючого середовища.

Задача 3. Відшукати моменти інерції ротора гіроскопа.

Задача 4. Визначити характеристики власного руху гіроскопа у кардановому підвісі, а також характеристики його вимушеного руху під дією моментів зовнішніх сил, що діють по осях карданового підвісу і змінюються з часом за гармонічним законом.

2.2. Створення математичної моделі

Математична модель – це математичний опис співвідношень постановки задачі. Такий опис можливий лише на ґрунті попередньо одержаних знань про поведінку об'єкта, що вивчається, і про способи правильного й ефективного опису цього поведіння у математичних термінах. В одних випадках утворення математичної моделі не складає труднощів (наприклад, модель є відомою за-

здадуться за результатами раніше проведених досліджень), а в інших потрібно неодноразове уточнення постановки задачі, виділення головних визначальних чинників, відкидання чинників, які незначно впливають на результат і т.д..

Так, для задачі 1 математична модель може бути утворена, якщо врахувати наступні теоретичні відомості.

1. До характеристик власного руху коливальної ланки, яким є фізичний маятник, відносять:

- 1) частоту власних коливань;
- 2) коефіцієнт загасання цих коливань.

2. При малих відхиленнях від вертикалі рух маятника з достатньою точністю описується лінійним диференціальним рівнянням другого порядку зі сталими коефіцієнтами:

$$J \cdot \ddot{\varphi} + R \cdot \dot{\varphi} + mgl \cdot \varphi = 0, \quad (2.1)$$

де φ – кут відхилення маятника від вертикалі; J – момент інерції маятника відносно його осі обертання; R – коефіцієнт демпфірування; m – маса маятника; g – прискорення вільного падіння; l – зміщення центра мас маятника відносно осі його обертання;

$\dot{\varphi} = \frac{d\varphi}{dt}$ – кутова швидкість повороту маятника навколо його осі обертання; $\ddot{\varphi} = \frac{d\dot{\varphi}}{dt}$ – кутове прискорення маятника.

3. Власний рух маятника описується співвідношенням

$$\varphi(t) = A \cdot e^{-h \cdot t} \cdot \sin(\omega_o \cdot t + \varepsilon), \quad (2.2)$$

де A – початкове значення амплітуди власних коливань і ε – початкова фаза власних коливань визначаються початковими умовами руху маятника, а ω_o – частота власних коливань та h – коефіцієнт загасання власних коливань; це параметри, які визначаються лише параметрами самого маятника і не залежать від інших чинників. Фактично ω_o і h є шуканими величинами.

4. Величини ω_o і h є відповідно уявною і дійсною частинами пари комплексно спряжених коренів характеристичного рівняння

$$J \cdot p^2 + R \cdot p + mgl = 0, \quad (2.3)$$

яке випливає з диференціального рівняння (2.1), тобто корінь рівняння (2.3) має вигляд:

$$p_{1,2} = -h \pm \omega_o. \quad (2.4)$$

У підсумку розв'язування задачі 1 математично зводиться до *відшукування комплексних коренів квадратного рівняння (2.3) і виділенню їхніх дійсної й уявної частин* за заданими первинними даними - значеннями параметрів J , R та mgl .

В задачі 2 треба припустити, що тіло є матеріальною точкою маси m , з'ясувати, під дією яких сил відбувається падіння тіла, визначити чинники, що впливають на силу опору, встановити залежність сили опору від цих факторів. Якщо вважати, що на тіло діють сила тяжіння $F_1 = mg$ та сила опору, що є про-

порційною до швидкості v падіння, тобто $F_2 = -k \cdot v$, то, на основі законів механіки одержимо рівняння $m \cdot a = m \cdot g - k \cdot v$, або

$$\frac{dv}{dt} = g - \frac{k}{m}v. \quad (2.5)$$

Це диференціальне рівняння із врахуванням початкової умови $v(t_0) = v_0$ і є математичною моделлю задачі.

У задачі 3 насамперед слід з'ясувати форму ротора, його розміри, розподіл мас, потім виділити у тілі ротора ряд частин, відшукування моментів інерції яких робиться досить просто (циліндри, кільця, конуси тощо). Тоді задача зводиться до обчислень моментів інерції окремих елементарних тіл і їхньому підсумовуванню. Формули обчислення моментів інерції окремих частин ротора і їх підсумовування і складуть математичну модель цієї задачі.

Постановка задачі 4 має містити опис власних параметрів системи "гіроскоп у кардановому підвісі", опис параметрів зовнішніх моментів сил, опис рівнянь руху. Наприклад, рівняння руху гіроскопа для цієї задачі можуть бути взяті у наступному вигляді

$$\begin{cases} A \cdot \ddot{\alpha} + H_o \cos \beta_o \cdot \dot{\beta} = N_m \cdot \sin(\omega \cdot t + \varepsilon_1) \\ B \cdot \ddot{\beta} - H_o \cos \beta_o \cdot \dot{\alpha} = L_m \cdot \sin(\omega \cdot t + \varepsilon_2) \end{cases}. \quad (2.6)$$

Тут α і β – кути повороту гіроскопа навколо осей підвісу; A та B – його моменти інерції, H_o – власний кінетичний момент гіроскопа, β_o – початкове значення кута β ; $\dot{\alpha} = \frac{d\alpha}{dt}$; $\dot{\beta} = \frac{d\beta}{dt}$; $\ddot{\alpha} = \frac{d\dot{\alpha}}{dt}$; $\ddot{\beta} = \frac{d\dot{\beta}}{dt}$; N_m , L_m – амплітуди змінювання моментів зовнішніх сил; ω – частота (колова) цього змінювання; ε_1 , ε_2 – початкові фази коливань цих моментів.

За математичну модель у цьому випадку може правити сукупність розв'язків рівнянь (2.6), наведена нижче:

$$\begin{aligned} \alpha &= \alpha_o + C_1(1 - \cos \lambda t) / \lambda + (C_2 \sin \lambda t) / \lambda + F_{1s}(1 - \cos \omega t) / \omega + \\ &+ (F_{1c} \sin \omega t) / \omega; \\ \beta &= \beta_o + [C_2(1 - \cos \lambda t) - C_2 \sin \lambda t] \sqrt{\frac{A}{B}} / \lambda + F_{2s}(1 - \cos \omega t) / \omega + \\ &+ (F_{2c} \sin \omega t) / \omega, \end{aligned} \quad (2.7)$$

де α_o і β_o – початкові значення кутів визначаються α і β ; $\lambda = \frac{H_o \cos \beta_o}{\sqrt{AB}}$ – частота власних (нутаційних) коливань гіроскопа; F_{1s} , F_{1c} , F_{2s} , F_{2c} , C_1 , C_2 визначаються сукупністю співвідношень:

$$\begin{aligned} F_{1s} &= -\lambda \cdot (l_s + \sqrt{\frac{B}{A}}v \cdot n_c) / (1 - v^2); & F_{1c} &= \lambda \cdot (-l_c + \sqrt{\frac{B}{A}}v \cdot n_s) / (1 - v^2); \\ F_{2s} &= \lambda \cdot (n_s - \sqrt{\frac{A}{B}}v \cdot l_c) / (1 - v^2); & F_{2c} &= \lambda \cdot (n_c + \sqrt{\frac{A}{B}}v \cdot l_s) / (1 - v^2); \end{aligned}$$

$$C_1 = \lambda \cdot \left(\sqrt{\frac{B}{A}} n_c + \nu \cdot l_s \right) / (1 - \nu^2) - \sqrt{\frac{B}{A}} \cdot \dot{\beta}_o;$$

$$C_2 = \lambda \cdot \left(l_c - \sqrt{\frac{B}{A}} \nu \cdot n_s \right) / (1 - \nu^2) + \dot{\alpha}_o;$$

$$n_s = \frac{N_s \sqrt{AB}}{H_o^2 \cdot \cos^2 \beta_o}; \quad n_c = \frac{N_c \sqrt{AB}}{H_o^2 \cdot \cos^2 \beta_o}; \quad l_s = \frac{L_s \sqrt{AB}}{H_o^2 \cdot \cos^2 \beta_o}; \quad l_c = \frac{L_c \sqrt{AB}}{H_o^2 \cdot \cos^2 \beta_o};$$

$$N_c = N_m \cdot \sin \varepsilon_1; \quad N_s = N_m \cdot \cos \varepsilon_1; \quad L_c = L_m \cdot \sin \varepsilon_2; \quad L_s = L_m \cdot \cos \varepsilon_2;$$

$\nu = \frac{\omega}{\lambda}$ - відносна частота коливань моментів сил; $\dot{\alpha}_o$ і $\dot{\beta}_o$ - початкові значення

кутових швидкостей $\dot{\alpha}$ і $\dot{\beta}$. Рух гіроскопа за цими співвідношеннями може бути визначений у довільний момент часу.

Але як математичну модель можна також розглядати і первісну систему диференціальних рівнянь (2.6) за вказаних початкових умов.

Складання математичної моделі у прикладній задачі є найбільш складним і відповідальним етапом розв'язування і потребує, окрім істотних знань у спеціальній області, також й математичних й теоретичних знань.

Вже на цьому етапі розв'язування прикладної задачі *доводиться нехтувати багатьма реальними процесами*, як такими, що незначно впливають на досліджувані процеси, *абстрагуватися від впливу багатьох чинників*. Інакше кажучи, навіть коректно утворена математична модель завжди неповно, лише наближено, відображає реальні процеси. Але при цьому вона набуває риси більшої ясності, прозорості, більш доступна вичерпному дослідженню (з того боку, який підлягає вивченню).

2.3. Математичне моделювання

Модель утворюється задля подальшого її дослідження з метою одержати нові знання про відповідний реальний об'єкт. Таке дослідження вже готової моделі називають *моделюванням*. Дослідження *математичної* моделі називатимемо *математичним* моделюванням.

Математична задача є абстрагованою від конкретної сутності задачі. Для її розв'язування створюються спеціальні теоретичні або обчислювальні методи, причому *до тої самої математичної моделі можуть зводитися зовсім різні прикладні задачі*.

Так, задача 1 звелася до розв'язування квадратного рівняння, яке може відображувати характеристичне рівняння не тільки фізичного, але й математичного маятника, маси, яка з'єднана пружиною з корпусом (лінійного акселерометра), гіроскопічного тахометру і т. п.

Диференціальне рівняння (2.5) у задачі 2 може бути моделлю і для багатьох інших задач (вивчення змінювання швидкості тіла у в'язкому середовищі, змінювання електричного струму у найпростішому електричному ланцюзі, змінювання швидкості репродукції бактерій тощо).

Задля розв'язування задачі 3 потрібно обчислити низку визначених інтегралів. До обчислення визначених інтегралів приходять і при відшукуванні площ складних фігур, об'єму тіла або дуги плоскої кривої, розрахунках роботи змінної сили й у багатьох інших фізичних задачах.

Математична модель (2.7) задачі 4 може описувати не тільки поведіння гіроскопу, але й будь-якої іншої системи, якщо диференціальні рівняння руху останньої подібні до рівнянь (2.6).

2.4. Побудова обчислювальної моделі

Побудова обчислювальної моделі може здійснюватися різними методами, які можна поділити на *точні* й *наближені*. *Точні методи* – це такі, які після скінченної кількості дій (обчислень) приводять до *точного результату* за умови, що обчислення здійснюються без похибок. Наближеними називають такі методи, які за тих самих умов дозволяють одержати результат лише з деякою похибкою.

При використанні точних методів етап досліджування математичної моделі поділяється на такі підетапи: 1) відшукування точного розв'язку математичної моделі; 2) підставлення вихідних даних у знайдений точний розв'язок і реалізація передбачених ним обчислень.

Наприклад, для розв'язування задачі 1 краще використати точний метод, тобто формулу

$$p_{1,2} = -\frac{R}{2J} \pm j \cdot \sqrt{\frac{mgl}{J} - \left(\frac{R}{2J}\right)^2} \quad (2.8)$$

(припускається, що $mgl/J > [R/(2J)]^2$), але можна застосовувати й наближені способи відшукування коренів квадратного рівняння.

Диференціальне рівняння (2.5) задачі 2 краще розв'язувати, розділяючи змінні, тобто приводячи його до вигляду

$$\frac{m}{mg - kv} dv = dt. \quad (2.9)$$

Однак, його можна розглядати і як лінійне диференціальне рівняння зі сталими коефіцієнтами, або розв'язувати (інтегрувати) наближеними чисельними методами.

При розв'язуванні задачі 3 слід використовувати методи наближеного обчислення визначених інтегралів.

Задачу 4 також можна розв'язувати двома шляхами. Розглядаючи систему диференціальних рівнянь (2.6) як вихідну математичну модель, можна, з одного боку, знайти точний її розв'язок (2.7), а потім здійснити підставлення значень вихідних даних і дійти явних залежностей $\alpha(t)$ і $\beta(t)$, а отже, й $\alpha(\beta)$. З іншого боку, до системи (2.6) можна безпосередньо застосувати методи чисельного інтегрування диференціальних рівнянь (наближені методи).

Досліджування математичної моделі наближеними методами поділяється на такі етапи:

- 1) обрання обчислювального методу (зазвичай наближених чисельних методів буває декілька);
- 2) вивчення або складання алгоритму методу;
- 3) реалізація алгоритму за допомогою обчислювальних засобів.

При виборі чисельного методу суттєвими є *обсяг обчислень, швидкість збіжності обчислень* (як швидко здобувається результат) та інші чинники. Зокрема, обрання методу залежить і від вхідних даних.

Крім того, на вибір методу впливають засоби його реалізації (ручний розрахунок, наявність обчислювальної машини, наявність готової програми тощо). Так, якщо буде використані швидкодіюча ЕОМ і готова програма, то обсяг обчислень не має засмучувати виконавця і бути визначальним фактором при обранні методу. При ручному ж розрахунку слід віддати перевагу методу, який, можливо, потребує деяких певних попередніх досліджень і перетворень математичної моделі, але завдяки цьому потребує й значно меншу кількість обчислень.

2.5. Алгоритм методу

Алгоритмом методу називається система правил, яка задає точно визначену послідовність операцій, яка приводить до шуканого результату (точного або наближеного).

Алгоритм – одне із ґрунтовних понять математики. Хід розв'язування обчислювальної (і взагалі будь-якої) задачі має бути поданий через алгоритм.

Алгоритм можна записати словесно-формульно або у вигляді *схеми*. Так, словесно-формульний опис алгоритму розв'язування задачі 1 за формулою (8) має наступний вигляд:

1. Обчислити $h = \frac{R}{2J}$.

2. Обчислити $D = h^2 - \frac{mgl}{J}$.

3. Якщо $D < 0$, перейти до п. 7.

4. Обчислити $p_1 = -h + \sqrt{D}$ і $p_2 = -h - \sqrt{D}$.

5. Подати на пристрій виведення інформацію: "Рівняння має два дійсні корені:" і роздрукувати значення шуканих коренів p_1 і p_2 .

6. Перейти до п. 8.

7. Вивести на пристрій виведення інформацію:

"Коефіцієнт загасання дорівнює " і вивести значення h

"Частота власних коливань дорівнює" і роздрукувати значення $\sqrt{-D}$.

8. Кінець обчислень.

При виконанні алгоритму перехід від однієї дії до іншої здійснюється строго у порядку їх запису. Якщо ж потрібно перервати природний хід дій за деякої умови, слід указувати на це (див. п. 3 наведеного алгоритму).

На рис. 2.2. подані зображувальні елементи блок-схеми алгоритму обчислень.

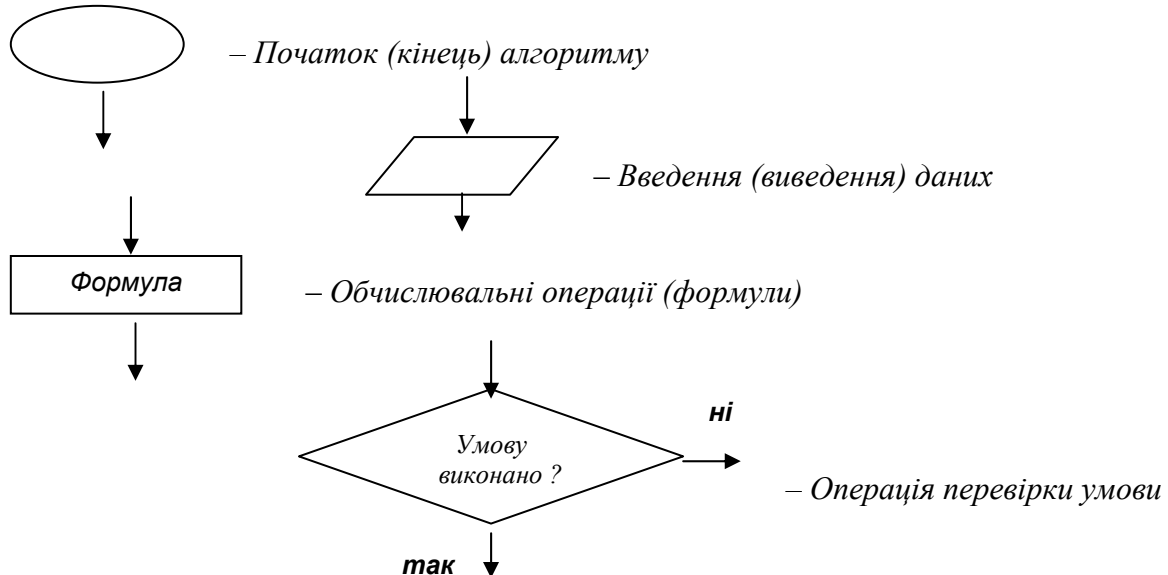


Рис. 2.2. Елементи блок-схеми алгоритму

Фігури з'єднуються лініями зі стрілками, які вказують на операцію, до виконання якої слід перейти.

Структурною схемою алгоритму називають графічне зображення послідовності дій обчислювального процесу. У схемі кожна дія розміщується у певному геометричному символі (фігурі). Послідовність дій указується на схемі напрямком стрілок на лініях, якими з'єднують ці символи. Зазвичай прийнято початок і кінець обчислень зображувати овалами, введення даних і виведення результатів – у вигляді паралелограма. Обчислювальні операції розміщуються у прямокутниках, а операція перевірки деякої умови зображується у вигляді ромбу. У середині кожної фігури розміщується стислий формульний опис відповідної операції. Символи операцій перевірки умови мають два виходи: "так" і "ні".

Стрілка на лінії, що виходить із виходу "так" вказує на операцію, до виконання якої потрібно перейти, якщо умову, яка перевіряється, виконано. Стрілка з написом "ні" вказує на операцію, до виконання якої слід перейти у випадку, коли умову не виконано.

Для прикладу на рис 2.3 зображено схему алгоритму відшукування коренів квадратного рівняння.

2.6. Реалізація методу обчислень

Обчислення по алгоритмах відбувається за допомогою різних обчислювальних засобів.

При ручних (безпосередніх) розрахунках зазвичай використовуються найпростіші обчислювальні засоби: логарифмічна лінійка, таблиці, механічні, електричні, електронні клавішні обчислювальні машини. Проміжні результати дій алгоритму треба записувати у спеціальний розрахунковий бланк. Наявність програмувальних мікрокалькуляторів дозволяє реалізовувати обчислення автоматично, під керуванням програми.

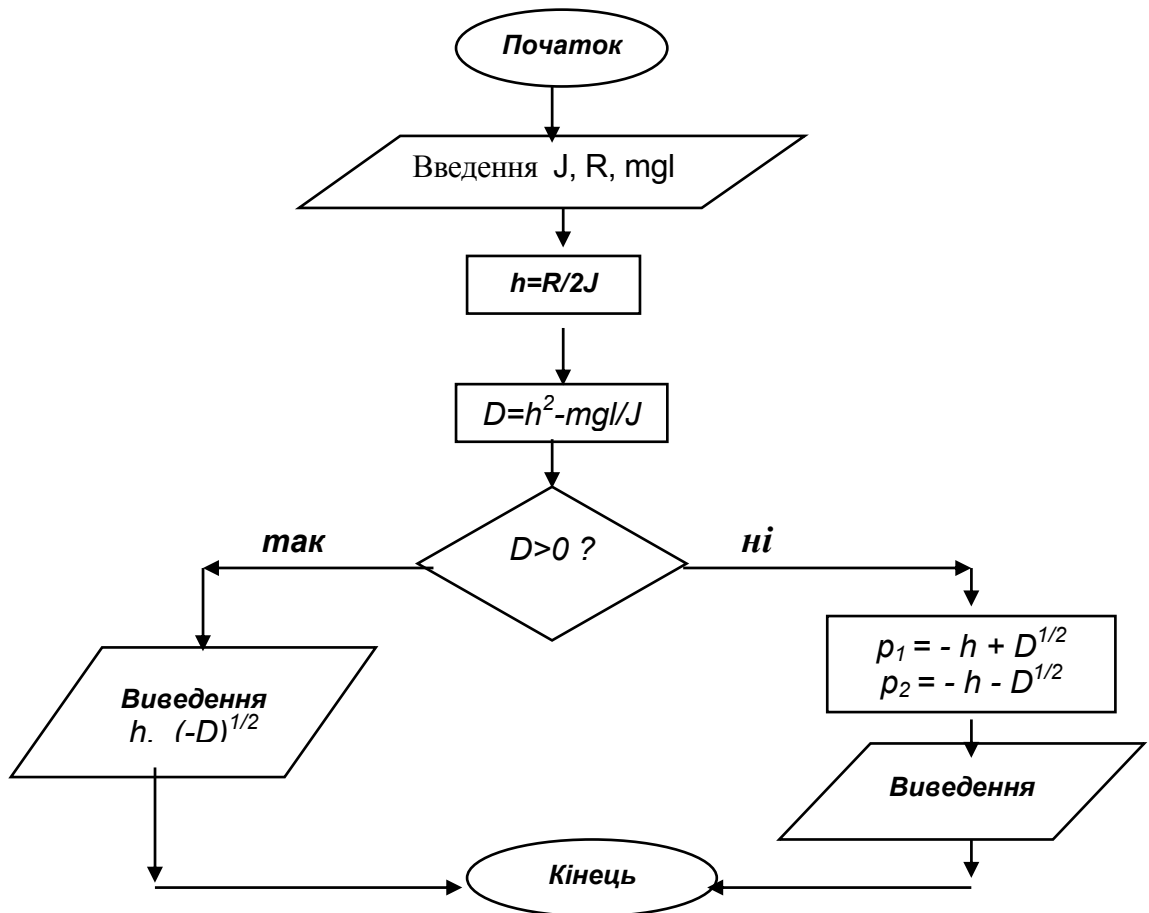


Рис. 2.3. Схема алгоритму відшукування коренів квадратного рівняння

Суттєвим є контроль обчислень, який проводять за так званим *контрольним прикладом* (тестом). Результат контрольного прикладу має бути заздалегідь відомим, тобто він або є очевидним, або його відшукують яким-небудь іншим способом. При ручному рахунку контроль рекомендується проводити поетапно. При розрахунках на ЕОМ за складеною програмою контрольний приклад заздалегідь прораховують вручну, а потім звіряють поетапно результати розрахунків із здійснюваними машиною.

2.7. Запитання для самоперевірки

1. Що таке "модель"? "моделювання"?
2. Які об'єктивні й суб'єктивні чинники можуть впливати на створювану модель?
3. Які види моделей трапляються в інженерній практиці?
4. Що таке "математична модель"? "обчислювальна модель"?
5. Як можна охарактеризувати постановку інженерної задачі?
6. Які етапи проходить у загальному випадку розв'язування інженерної задачі?
7. Що таке "алгоритм"? На які види поділяються алгоритми?
8. Що таке "блок-схема алгоритму"? Які позначення прийняті при побудові блок-схеми алгоритму?
9. Якими є загальні правила побудови блок-схеми алгоритму?

3. ОРГАНІЗАЦІЯ НАБЛИЖЕНИХ ОБЧИСЛЕНЬ

Як вже зазначалося, неминучим етапом розв'язування інженерних задач є проведення наближених розрахунків із наближеними вихідними даними. Найбільш розгорнутим варіантом таких розрахунків є проведення обчислень за програмною моделлю, яка реалізує складний чисельний обчислювальний алгоритм.

Тому до основних умінь, необхідних інженеру в його професійної діяльності, слід віднести *вміння грамотно (раціонально) організувати обчислення* (у тому числі – у вигляді програми для ЕОМ). Під цим слід розуміти наступне:

- знати можливі джерела похибок;
- вміти правильно записувати наближені дані й результати (у тому числі проміжні);
- вміти оцінювати похибку результату за заданими похибками компонент;
- вміти обирати найбільш раціональний *порядок обчислень*;
- вміти обирати алгоритм обчислення, найстійкіший до похибок обчислень;
- вміти контролювати хід і результати обчислень із метою виключення грубих похибок.

Не маючи достатніх умінь і навичок практичних обчислень можна одержати результат, який не матиме нічого спільного із дійсним розв'язком задачі.

3.1. Джерела й види похибок

На кожному етапі розв'язування прикладної задачі виникають власні джерела похибок.

Математична модель – це вже наближене подання реального об'єкта. Вихідні дані, що використовуються у розрахунках і виходять з експерименту, можна визначити лише наближено. Деякі точні числа, такі як π , $\sqrt{3}$, $6/7$ і т. п., при обчисленнях на ЕОМ вимушено замінюють десятковими дробами, залишаючи лише певну кількість знаків після десяткової коми. Обчислювальні методи у більшості також є наближеними. Навіть при використанні найпростішої формули результат, як правило, одержують наближений.

Основні джерела виникнення похибок наближеного розв'язування прикладних задач такі.

1. *Похибки математичної моделі.* Їх пов'язано з використаними припущеннями, які дозволяють спростити математичну модель задачі. Вони не контролюються у процесі чисельного розв'язування задачі і можуть бути зменшені лише за рахунок більш точного математичного опису фізичної задачі.

2. Похибки первісних даних. Значення параметрів, що входять у математичний опис задачі, вимірюються експериментально з деякою похибкою.

Похибки математичної моделі і вихідних даних у цілому утворюють так звані неусувні похибки. Назву обумовлено тим, що ці види похибок не можна усунути шляхом організації обчислень. Зменшення їх лежить лише на шляху перебудови математичної моделі і точнішого виміру вихідних даних.

3. Дії над наближеними числами приводять до поширення похибок на кінцевий результат обчислень. У залежності від того, як (раціонально чи ні) організований обчислювальний процес, похибки одержаного результату можуть бути меншими за похибки первісних даних, а можуть і, навпаки, набагато перевищувати похибки вихідних даних.
4. Похибки наближеного методу, або похибки усікання. При чисельному розв'язуванні задачі точний оператор, в якому кількість чисел або операцій перевищує допустимі межі, замінюється наближеним, який потребує скінченної кількості операцій. Наприклад, замінюють інтеграл сумою, функцію – поліномом (багаточленом) або будують нескінченний процес і обривають його після скінченної кількості операцій.
5. Обчислювальні похибки, що виникають в результаті вимушеного округлення чисел, наприклад, внаслідок скінченної кількості розрядів у запису числа в оперативній пам'яті ЕОМ.

Якщо розв'язок деякої задачі неперервно залежить від вхідних даних, тобто малому змінюванню вхідних даних відповідає мале змінювання розв'язку, то задача називається стійкою за вхідними даними, або грубою. У стійкому обчислювальному алгоритмі похибки округлення не накопичуються.

Точність наближеного числа характеризується поняттями абсолютної й відносної похибки.

Абсолютною похибкою наближеного числа a називають абсолютне значення різниці між ним і точним його значенням:

$$\Delta = |A - a|,$$

де A – точне значення, a – наближене значення. Абсолютна похибка має суто теоретичний інтерес, оскільки точне значення A невідоме. Тому на практиці частіше використовують граничну абсолютну похибку Δ_a наближеного числа a , рівну по можливості найменшому числу, що є більшим за абсолютну похибку

$$\Delta = |A - a| \leq \Delta_a.$$

Значення a і Δ_a дозволяють вказати інтервал, в якому розташоване точне значення A :

$$a - \Delta_a \leq A \leq a + \Delta_a.$$

Частіше використовується компактніший запис

$$A = a \pm \Delta_a.$$

Очевидно, таке визначення абсолютної похибки не є однозначним. Так, якщо $A = \pi$, а як наближене значення узяти $a = 3,14$, то, враховуючи, що $3,140 < \pi < 3,142$, можна записати:

$$|\pi - a| < 0,002; \quad |\pi - a| < 0,01; \quad |\pi - a| < 0,1.$$

Кожне з чисел 0,002; 0,01; 0,1 буде граничною абсолютною похибкою числа a . Але чим ближче між собою числа $|A - a|$ і Δ_a , тим точніше абсолютна похибка оцінює фактичну похибку.

Основною характеристикою точності наближеного числа є його відносна похибка

$$\delta = \frac{\Delta}{|A|}.$$

Оскільки число A невідоме, то, як правило, вважають

$$\delta = \frac{\Delta}{|a|}.$$

Аналогічно з нерівності $\delta \leq \delta_a$ визначають граничну відносну похибку числа a , вважаючи

$$\delta_a = \frac{\Delta_a}{|a|}.$$

Величина δ характеризує якість наближення. Це безрозмірна величина, зазвичай її виражають у процентах. Так, відносна похибка числа π , прийнятого за наближене значення числа π , при $\Delta(3,14) = 0,002$ дорівнює

$$\delta(3,14) = \frac{0,002}{3,14} = 0,00064, \quad (0,064\%).$$

3.2. Запис наближених чисел. Правило округлення

Записувати наближене число у вигляді $a \pm \Delta_a$ незручно. Тому в обчислювальній практиці часто вдаються до різних прийомів, які дозволяють тільки за записом наближеного числа a висновувати про його похибку.

Нехай наближене число a подане у вигляді скінченного десяткового дробу:

$$a = \alpha_m \cdot 10^m + \alpha_{m-1} \cdot 10^{m-1} + \dots + \alpha_{m-n+1} \cdot 10^{m-n+1}; \quad (\alpha_m \neq 0),$$

де α_i – цифри числа a ($\alpha_i = 0, 1, \dots, 9$).

Значущими цифрами числа називають усі цифри у запису числа, починаючи з першої ненульової зліва. Наприклад, у чисел $a = 0,0503$, $b = 0,00630500$ значущими є підкреслені цифри.

Значущу цифру називають вірною, якщо абсолютна похибка числа не перевищує половини одиниці розряду, який відповідає цій цифрі. У зворотному випадку цифра вважається сумнівною.

Наприклад, число 647,326 при $\Delta(a) = 0,03$ ($< 0,5 \cdot 10^{-1}$) має чотири вірні цифри 6, 4, 7, 3 і дві сумнівні 2, 6.

За правилом, запропонованому О. М. Криловим, наближене число потрібно записувати так, щоб усі значущі цифри у запису числа були вірними, а перша цифра з відкинутої частини відносилася до сумнівних. Так, число 0,884, відоме з похибкою 0,004, має бути записано так: 0,88. Раніше згадане число 647,326 при описаних умовах слід записати у вигляді 647,3.

У математичних таблицях значень функцій приводяться тільки вірні цифри. Відносна похибка табличних значень, наприклад, у тризначних таблицях не перевищує $0,5 \cdot 10^{-3}$, у семизначних – $0,5 \cdot 10^{-7}$.

Точність наближеного числа залежить не від кількості значущих цифр, а від кількості вірних цифр.

При проведенні розрахунків остаточний наближений результат зазвичай округлюють до його вірних цифр, залишаючи одну сумнівну, а у проміжних результатах зберігають одну, дві, а часом і три сумнівні цифри.

Приклад. Задані числа при вказаних абсолютних похибках округлити до вірних цифр. Визначити абсолютну похибку результату.

- | | |
|---|--|
| 1. $a_1 = 2,6219$, $\Delta_{a_1} = 0,024$; | 6. $a_2 = 47,35$, $\Delta_{a_2} = 1,3$; |
| 2. $a_3 = 6,9971$, $\Delta_{a_3} = 0,0009$; | 4. $a_4 = 0,648$, $\Delta_{a_4} = 0,04$. |

РОЗВ'ЯЗУВАННЯ

1. Оскільки $0,024 < 0,05$, то число a_1 потрібно округлити до 0,1. Одержимо число $a_1 \approx 2,6$ із двома вірними цифрами. Визначимо абсолютну похибку результату $\Delta = 0,024 + |2,6219 - 2,6| = 0,0459 \approx 0,05$.

2. Через те що $1,3 < 0,5 \cdot 10^1$, то число a_2 слід округлити до десятків. Одержимо $a_2 \approx 5 \cdot 10$ з однією вірною цифрою. Варто відзначити, що не можна писати ані 50, ані $5,0 \cdot 10$, бо у обох випадках у відповідності із прийнятою системою запису це буде означати, що похибка записаного числа менша за 0,5, що суперечить умові. Абсолютна похибка результату дорівнює $\Delta(5 \cdot 10) = 1,3 + |50 - 47,35| = 1,3 + 2,65 = 3,95 \approx 4$.

3. Оскільки $0,0009 < 0,005$, то округлення числа a_3 здійснюємо до 0,01. Одержимо $a_3 \approx 7,00$ із трьома вірними цифрами. Абсолютна похибка результату буде наступною $\Delta(a_3) = 0,0009 + |7,00 - 6,9971| \approx 0,004$. У запису 7,00 нулі свідчать про три його вірні знаки, і цей запис відрізняється від 7 або 7,0.

4. Оскільки $0,04 < 0,05$, то число a_4 округлюємо до 0,1. Одержуємо $a_4 \approx 0,6$ з однією вірною цифрою. Абсолютна похибка результату дорівнюватиме $\Delta(0,6) = 0,04 + 0,048 = 0,088 > 0,05$, тобто цифра 6 вже є сумівною. Тому при округленні рекомендується залишати одну-дві сумнівні цифри

$$a_4 = 0, (65); \quad \Delta(0,65) = 0,042 < 0,05.$$

Зазначимо, що термін "вірні цифри" не слід розуміти буквально. Так, у числі 7,00, яке замінює 6,9971 у попередньому прикладі, жодна з цифр не збігається з цифрами числа, хоча усі цифри цього числа є "вірними" в описаному вище сенсі $|6,9971 - 7,00| < 0,005$. Однак вірні знаки наближеного числа часто збігаються з відповідними цифрами точного числа.

Відносна похибка наближеного числа безпосередньо залежить від кількості його вірних знаків. Наприклад, якщо наближене число a має три вірних знаки, то його відносна похибка δ_a перебуває у межах від 0,05 до 0,5% (залежить від першої значущої цифри числа a). При збільшенні кількості вірних знаків на 1 відносна похибка зменшується у 10 разів.

На практиці зазвичай вважають, що число a є наближенням числа A з n вірними десятковими знаками, якщо $\Delta_a \leq 10^{-n}$. При такому визначенні в числі $a = 647,35$ при $\Delta_a = 0,095 < 0,1$ будуть вірними цифри 6, 4, 7, 3 і його слід записувати у вигляді 647,3. Згідно з колишнім визначенням, оскільки $\Delta(a) = 0,095 < 0,5$, то у цьому числі будуть вірними лише три цифри 6, 4 і 7 і його слід записувати як 647.

3.3. Похибки результату при діях із наближеними числами

При організації процесу складних числових розрахунків, зокрема, при складанні обчислювальних програм на ЕОМ, слід вміти оцінювати можливі похибки результату обчислень внаслідок поширення похибок вихідних даних і застосовувати способи їх зменшення.

Для оцінки похибок результатів потрібно знати похибки вихідних чисел і правила обчислення похибки результату. Розглянемо ці правила.

3.3.1. Похибки підсумовування

Неважно впевнитися у слушності наступних тверджень.

Абсолютна похибка суми й різниці дорівнює сумі абсолютних похибок доданків

$$a = b \pm c \Rightarrow \Delta_a = \Delta_b + \Delta_c.$$

Відносна похибка суми двох величин однакового знаку перебуває в інтервалі між найменшою й найбільшою відносними похибками доданків

$$\delta_a = \frac{\Delta_a}{|a|} = \frac{\Delta_b + \Delta_c}{|b + c|}.$$

За умови $|b|=|c|$ матимемо
$$\delta_a = \frac{\Delta_b + \Delta_c}{2|b|} = \frac{\delta_b + \delta_c}{2}.$$

Якщо ж $|b| \gg |c|$, через що останнім доданком у знаменнику можна знехтувати, то

$$\delta_a = \frac{\Delta_b + \Delta_c}{|b|} = \delta_b + \frac{\Delta_c}{|b|} = \delta_b + \delta_c \frac{|c|}{|b|} \approx \delta_b.$$

Подамо вираз для відносної похибки у такий спосіб

$$\delta_a = \delta_b \frac{|b|}{|b+c|} + \delta_c \frac{|c|}{|b+c|}.$$

З його розгляду випливає наступне.

Додавання величин протилежного знаку (або віднімання величин однакового знаку) практично завжди приводить до збільшення відносної похибки результату у порівнянні з найбільшою з відносних похибок доданків.

Особливо небезпечним є віднімання дуже близьких величин. У цьому випадку відносна похибка результату може сягати неприпустимих величин.

Приклад. Потрібно обчислити площу кругового тонкого кільця із внутрішнім радіусом $r = 1,750$ і товщиною $h = 5,0 \cdot 10^{-3}$ за формулами $S = \pi[(r+h)^2 - r^2]$ або $S = \pi \cdot h \cdot (2r+h)$. Відшукати похибки.

РОЗВ'ЯЗУВАННЯ

Перш за все потрібно зазначити, що, як випливає з угоди про форму запису наближеного числа, абсолютні похибки вихідних даних є такими:

$$\Delta_r = 5 \cdot 10^{-4}; \quad \Delta_h = 5 \cdot 10^{-5},$$

а відносні похибки

$$\delta_r = \frac{5 \cdot 10^{-4}}{1,75} \approx 2,9 \cdot 10^{-4}; \quad \delta_h = \frac{5 \cdot 10^{-5}}{5 \cdot 10^{-3}} \approx 10^{-2}.$$

Розрахунки згідно першої формули приводять до таких похибок:

$$\Delta_{r+h} = \Delta_r + \Delta_h = 5 \cdot 10^{-4} + 5 \cdot 10^{-5} = 5,5 \cdot 10^{-4};$$

$$\delta_{r+h} = \frac{\Delta_{r+h}}{r+h} = \frac{5,5 \cdot 10^{-4}}{1,755} = 3,13 \cdot 10^{-4}; \quad \delta_{(r+h)^2} = 2\delta_{r+h} = 6,26 \cdot 10^{-4};$$

$$\Delta_{(r+h)^2} = \delta_{(r+h)^2} \cdot (r+h)^2 = 6,26 \cdot 10^{-4} \cdot (1,755)^2 = 19,3 \cdot 10^{-4};$$

$$\delta_{r^2} = 2 \cdot \delta_r = 5,8 \cdot 10^{-4}; \quad \Delta_{r^2} = \delta_{r^2} \cdot r^2 = 5,8 \cdot 10^{-4} \cdot (1,75)^2 = 17,8 \cdot 10^{-4};$$

$$\Delta_{(r+h)^2 - r^2} = \Delta_{(r+h)^2} + \Delta_{r^2} = 19,3 \cdot 10^{-4} + 17,8 \cdot 10^{-4} \approx 3,7 \cdot 10^{-3};$$

$$\delta_{(r+h)^2 - r^2} = \frac{\Delta_{(r+h)^2 - r^2}}{(r+h)^2 - r^2} = \frac{3,7 \cdot 10^{-3}}{17,5 \cdot 10^{-3}} = 0,21.$$

Якщо ж скористатися другою формулою, то одержимо такі похибки:

$$\Delta_{2r+h} = \Delta_{2r} + \Delta_h = 2\Delta_r + \Delta_h = 10 \cdot 10^{-4} + 5 \cdot 10^{-5} = 10,5 \cdot 10^{-4};$$

$$\delta_{2r+h} = \frac{\Delta_{2r+h}}{2r+h} = \frac{10,5 \cdot 10^{-4}}{3,505} = 3 \cdot 10^{-4};$$

$$\delta_{(2r+h)h} = \delta_{2r+h} + \delta_h = 3 \cdot 10^{-4} + 10^{-2} \approx 1 \cdot 10^{-2};$$

$$\Delta_{(2r+h)h} = \delta_{(2r+h)h} \cdot (2r+h) \cdot h = 1 \cdot 10^{-2} \cdot (3,505) \cdot 0,005 = 1,75 \cdot 10^{-4};$$

Як бачимо

$$\frac{\Delta_{(2r+h)h}}{\Delta_{(r+h)^2 - r^2}} = \frac{1,75 \cdot 10^{-4}}{3,7 \cdot 10^{-3}} = 4,75 \cdot 10^{-2} \approx 0,05,$$

тобто підрахунок за другою формулою дає змогу одержати результат у 20 разів точніший, ніж розрахунок за першою формулою, де відбувається віднімання близьких за значенням величин.

Наведений приклад засвідчує, що *похибка результату залежить від порядку проведення обчислень* і це потрібно враховувати при розрахунках. Алгебрично наведені формули тотожні, але для проведення обчислень кращою є друга. У першій формулі при відніманні близьких величин $(r+h)^2$ і r^2 різко збільшується відносна похибка.

3.3.2. Похибки добутку, ділення й обчислення довільної функції

Доведемо, що *відносна похибка добутку дорівнює сумі відносних похибок співмножників*.

Дійсно:

$$a = b \cdot c \Rightarrow (a + \Delta_a) = (b + \Delta_b)(c + \Delta_c) = b \cdot c + \Delta_b \cdot c + \Delta_c \cdot b + \Delta_b \cdot \Delta_c,$$

звідки випливає

$$\Delta_a = \Delta_b \cdot c + \Delta_c \cdot b + \Delta_b \cdot \Delta_c \Rightarrow \delta_a = \delta_b + \delta_c + \delta_b \cdot \delta_c.$$

Через те що $\delta_b \ll 1$ і $\delta_c \ll 1$, то $\delta_b \delta_c \ll \delta_b$, а тому

$$\boxed{\delta_a \approx \delta_b + \delta_c}; \quad \Delta_a = \delta_a \cdot |a| = (\delta_b + \delta_c) \cdot |b \cdot c|,$$

що й треба було довести.

Відносна похибка обчислення величини, зворотної до даної, дорівнює відносній похибці вихідної величини.

Щоб довести це, врахуємо, що, якщо $d = \frac{1}{c} = c^{-1}$, то

$$d + \Delta d = \frac{1}{c - \Delta c} \approx \frac{1}{c} \left(1 + \frac{\Delta c}{c}\right) = \frac{1}{c} (1 + \delta_c).$$

З цього випливає $d = \frac{\delta_c}{c}$, а значить $\delta_d = \frac{\Delta d}{d} = \frac{\delta_c/c}{1/c} = \delta_c$, що й треба було

довести.

Враховуючи це, можна дійти висновку, що *відносна похибка частки дорівнює сумі відносних похибок діленого та дільника*:

$$a = \frac{b}{c} \Rightarrow \delta_a = \delta_b + \delta_c \Rightarrow \Delta_a = (\delta_b + \delta_c) b / c.$$

Аналогічно можна встановити, що *відносна похибка піднесення до степеня n наближеного числа (n – натуральне ціле) дорівнює добуткові відносної похибки основи на абсолютну величину показника степеня*

$$a = b^n \Rightarrow \delta_a = |n| \cdot \delta_b \Rightarrow \Delta_a = |n| \cdot \Delta_b \cdot b^{n-1}.$$

Абсолютна похибка обчислення функції дорівнює добутку абсолютної похибки аргументу на абсолютну величину похідної від функції:

$$a = f(b) \Rightarrow \Delta_a = \Delta_b \cdot \left| \frac{df(b)}{db} \right| \Rightarrow \delta_a = \delta_b \cdot \left| \frac{df(b)}{db} \cdot \frac{b}{f(b)} \right|.$$

Приклад 1. Похибка обчислення лінійної функції $y = k \cdot x$.

$$\text{Маємо } \frac{dy}{dx} = k \Rightarrow \Delta_y = \Delta_x \cdot k \Rightarrow \delta_y = \frac{\Delta_y}{y} = \frac{\Delta_x \cdot k}{k \cdot x} = \frac{\Delta_x}{x} = \delta_x.$$

Приклад 6. Похибки обчислення синуса $y = \sin(x)$.

$$\text{У цьому випадку } \frac{dy}{dx} = \cos(x), \text{ а тому } \Delta_y = \Delta_x \cdot |\cos(x)|, \text{ а } \delta_y = \delta_x \cdot \frac{|x|}{|\operatorname{tg}x|}.$$

Приклад 3. Похибки обчислення косинуса $y = \cos(x)$.

$$\text{У цьому випадку } \frac{dy}{dx} = -\sin(x), \text{ а тому } \Delta_y = \Delta_x \cdot |\sin(x)|, \text{ а } \delta_y = \delta_x \cdot |x \cdot \operatorname{tg}x|.$$

Аналіз останніх прикладів дозволяє висновувати, що

- 1) похибка обчислень суттєво залежить від значення аргументу;
- 2) при малих значеннях аргументу обчислення косинуса здійснюється зі значно меншою відносною похибкою, ніж похибка завдання аргументу;
- 3) обчислення косинуса при значеннях аргументу, близьких до $\pm \pi/2$ приводить до вельми значних обчислювальних похибок; відносна похибка визначення косинуса у цьому випадку у багато разів перевищує відносну похибку завдання кута;
- 4) відносна ж похибка визначення синуса у діапазоні $[-\pi/2, \pi/2]$ завжди менша за відносну похибку аргументу.

3.4. Поширення похибок округлення при обчисленнях. Зберігання чисел в ЕОМ

До цього були розглянуті приклади розрахунків похибок результатів обчислень, обумовлених впливом похибок вихідних даних. Тепер зосередимо увагу на похибках результату обчислень за рахунок похибок округлення.

При виконанні арифметичних дій на будь-якому обчислювальному пристрої (логарифмічній лінійці, калькуляторі або ЕОМ) неминуче округлення проміжних результатів до певної кількості розрядів, а арифметичні операції з округленням мають інші властивості, аніж точні операції. Так, точні операції є комутативними, асоціативними і дистрибутивними. Ті ж операції, реалізовані на обчислювальному пристрої, вже не є такими.

Машинна арифметика має власні характерні особливості. Правильно враховуючи їх, можна досягти високої ефективності у розв'язуванні задач на ЕОМ. Неуважність до цих особливостей нерідко приводить до помилкових результатів.

Нехай обчислення відбуваються на ЕОМ, в якій кожне число подається п'ятьма значущими цифрами. Складемо два числа

$$9,2654 + 7,1625 = 16,4279.$$

Результат містить 6 значущих цифр і не вміщується у розрядну сітку машини. Його буде округлено до 16,428, і при цьому виникне (крім похибки внаслідок похибок вихідних даних) похибка округлення.

Оскільки в оперативній пам'яті число завжди зберігається з фіксованою кількістю розрядів, то округлення виникає весь час при запису проміжних результатів у пам'ять машини. Округлення полягає у корегуванні останнього (молодшого) розряду. Але й буває, що ніякого корегування не відбувається, а відкидається частина числа, яка не вміщується. Похибка округлення при цьому більша, але значне скорочення машинного часу й довжини робочої програми у цілому є економічно вигідними.

Число в ЕОМ подається у такому нормалізованому вигляді

$$x = \pm q^p \cdot \left(\sum_{k=1}^s \alpha_k \cdot q^{-k} \right), \quad (3.1)$$

де q – основа системи позиційного зчислення; p – *порядок числа*; s – кількість значущих розрядів (додатне ціле); α_k – цифри числа у заданій системі зчислення, причому вважається, що α_1 не дорівнює нулеві. Отже для десяткової системи зчислення $q = 10$, порядок числа – p , кількість значущих десяткових розрядів – s , цифри $\alpha_k = 0, 1, \dots, 9$. Наприклад, для числа $0,00123406 \cdot 10^{-5} = 0,123406 \cdot 10^{-7}$ десятковий порядок дорівнює $p = -7$, кількість значущих розрядів $s = 6$, а цифри

$$\alpha_1 = 1; \quad \alpha_2 = 2; \quad \alpha_3 = 3; \quad \alpha_4 = 4; \quad \alpha_5 = 0; \quad \alpha_6 = 6.$$

У нормалізованій формі (3.1)) вираз у дужках являє собою так звану *мантису числа*. Мантиса у нормалізованому десятковому поданні завжди менша за одиницю і більша або дорівнює 0,1. У цілому будь-яке число однозначно описується трьома його характеристиками:

- 1) знаком числа;
- 2) значенням мантиси (вираз у дужках у (3.1));
- 3) цілим числом p – порядком числа.

ЕОМ оперують в основному двійковою системою зчислення, тому для них $q = 2$. Обсяг оперативної пам'яті ЕОМ, який віддається на запис окремого числа, вимірюється у байтах (8 двійкових розрядів – бітів) і залежить від мови програмування, на який записано програму, і типу числа, під яким його оголошено у програмі.

Наприклад, у мові **Fortran** під запис цілого числа типу **INTEGER** відводять 2 байти, тобто 16 двійкових розряди. Один із цих розрядів займає запис знака числа (додатне число чи від'ємне). У решті розрядів записується абсолютне значення цього числа. Неважко зрозуміти, що у такий обсяг можна записати число, за абсолютним значенням не більше за 32516.

Довільне число типу **REAL** записується у Фортрані у 4 байти оперативної пам'яті. З 32 бітів цього обсягу один біт займає запис знака числа, 7 розрядів – двійковий запис десяткового порядку числа (з них 1 розряд – запис знака порядку) і 24 розряди займає запис мантиси числа. З цього випливає, що абсолют-

не значення порядку десяткового числа у цьому випадку не перевищує 36. Тобто тип REAL дозволяє оперувати з числами за абсолютним значенням від 10^{-32} до 10^{+32} . При цьому мантиса числа, маючи для запису 24 розряди, зберігає $k = n \cdot \lg(2) = 24 \cdot 0.3 \approx 7$ десяткових розрядів числа. Тобто розглядуваний тип зберігає сім вірних десяткових цифр у кожному числі.

Розглянемо типи даних, передбачені мовою **Pascal**.

Тут для подання цілих чисел використовують 5 типів даних:

- byte – займає 1 байт пам'яті; за його допомогою можуть зберігатися цілі додатні числа від 0 до 255;
- shortint (коротке ціле) – займає теж 1 байт пам'яті; цей тип зберігає цілі числа від -128 до 127;
- integer (ціле) – для даних цього типу відводять 2 байти пам'яті; за його допомогою записуються цілі числа від -32768 до +32767;
- word – займає теж 2 байти; ним записуються лише додатні цілі від 0 до 65535;
- longint (довге ціле) – обіймає 4 байти пам'яті; зберігає числа від –6.147.483.648 до +6.147.483.647.

Дійсні дані у **Pascal** подані наступними типами:

- single – на запис числа цього типу відводять 4 байти пам'яті, з них 24 біти займає запис мантиси, а 7 бітів – запис порядку числа; ним можуть бути подані дійсні числа з абсолютним значенням від $1,5 \cdot 10^{-45}$ до $3,4 \cdot 10^{+38}$ і з 7 вірними десятковими розрядами;
- real – займає 6 байтів (48 бітів); із них 7 бітів – запис порядку, а 40 бітів – мантиса числа; записуються числа з абсолютним значенням від $2,9 \cdot 10^{-39}$ до $1,7 \cdot 10^{+38}$ з 12 вірними десятковими розрядами;
- double – запис має обсяг у 8 байтів (64 бітів), запис порядку займає 10 бітів, запис мантиси – 54 бітів; ним зберігаються числа з абсолютним значенням від $5 \cdot 10^{-324}$ до $1,7 \cdot 10^{+308}$ і з 16 вірними десятковими розрядами;
- extended – тип, під який відводять 10 байтів оперативної пам'яті (14 бітів – під запис порядку і 65 – під запис мантиси); при цьому забезпечується збереження чисел від $1,9 \cdot 10^{-4951}$ до $1,1 \cdot 10^{+4932}$ з 19 вірними десятковими розрядами.

У середовищі комп'ютерної системи **Mat LAB** усі числові дані мають єдиний тип **double**, який по всіх показниках збігається з відповідним типом мови **Pascal**.

Максимальна відносна похибка округлення при записі числа у пам'ять ЕОМ залежить не від його величини, а лише від кількості s десяткових розрядів у запису мантиси у його поданні на ЕОМ:

$$\delta \approx 1 \cdot 10^{-s+1}.$$

Наприклад, для типу REAL у Фортрані $\delta = 1 \cdot 10^{-6}$, для того ж типу мови Паскаль $\delta = 1 \cdot 10^{-11}$. Дані типу **double** у мові **Pascal** і системі **MatLAB** мають елементарну похибку округлення

$$\delta = 1 \cdot 10^{-15}.$$

Розглянемо процес поширення похибок округлення при обчисленнях і впливу їх на похибку визначення результату обчислення.

Припустимо, потрібно скласти на ЕОМ дві величини

$$y = x_1 + x_2.$$

Цей процес розкладається на чотири етапи:

1) запис числа x_1 у пам'ять ЕОМ; при цьому, навіть коли вихідне значення x_1 відоме точно, при його запису може виникнути похибка округлення δ (це може бути, якщо точне значення величини містить більшу кількість десяткових розрядів, ніж s – кількість розрядів для запису числа в ЕОМ): $\delta_{x_1} = \delta$; $\Delta_{x_1} = \delta \cdot x_1$;

2) запис числа x_2 у пам'ять; це приводить до появи аналогічної похибки $\delta_{x_2} = \delta$; $\Delta_{x_2} = \delta \cdot x_2$;

3) підсумовування величин x_1 і x_2 ; при цьому перші дві похибки впливають на результат сумування $\delta_{x_1+x_2} = \frac{\Delta_{x_1} + \Delta_{x_2}}{x_1 + x_2} = \frac{\delta \cdot (x_1 + x_2)}{x_1 + x_2} = \delta$;

4) запис результату у пам'ять; при цьому може виникнути похибка округлення δ , якщо кількість розрядів в результаті перевищує кількість розрядів у запису числа у пам'ять; у цілому результуюча відносна похибка результату може сягати величини $\delta_y = \delta_{x_1+x_2} + \delta = 2\delta$, а абсолютну похибку сумування можна оцінити за формулою $\Delta_y = 2\delta \cdot (x_1 + x_2)$.

Якщо тепер перейти до сумування (послідовного) трьох величин

$$y = (x_1 + x_2) + x_3,$$

то, повторюючи попередній аналіз, дістанемо

■ похибка результату першого підсумовування $(x_1 + x_2)$ знайдена перед цим: $\Delta_y = 2\delta \cdot (x_1 + x_2)$;

■ похибка запису числа x_3 : $\Delta_{x_3} = \delta \cdot x_3$;

■ похибка сумування (проходження похибок вихідних даних):

$$\Delta_{x_1+x_2+x_3} = \Delta_{x_1+x_2} + \Delta_{x_3} = 2\delta \cdot (x_1 + x_2) + \delta \cdot x_3;$$

$$\delta_{x_1+x_2+x_3} = \frac{\Delta_{x_1+x_2+x_3}}{x_1 + x_2 + x_3} = \delta \frac{2(x_1 + x_2) + x_3}{x_1 + x_2 + x_3};$$

■ похибка запису результату у пам'ять ЕОМ – δ .

У цілому одержуємо

$$\delta_y = \delta_{x_1+x_2+x_3} + \delta = \delta \cdot \left[\frac{2(x_1 + x_2) + x_3}{x_1 + x_2 + x_3} + 1 \right] = \delta \cdot \frac{3(x_1 + x_2) + 2x_3}{x_1 + x_2 + x_3};$$

$$\Delta_y = \delta \cdot [3(x_1 + x_2) + 2x_3].$$

Поширення цього результату на підсумовування n чисел

$$y = x_1 + x_2 + \dots + x_n$$

приводить до наступних висновків

$$\Delta_y = \delta \cdot [n(x_1 + x_2) + (n-1) \cdot x_3 + \dots + 3x_{n-1} + 2x_n];$$

$$\delta_y = \delta \cdot \frac{n(x_1 + x_2) + (n-1) \cdot x_3 + \dots + 3x_{n-1} + 2x_n}{x_1 + x_2 + x_3 + \dots + x_{n-1} + x_n}.$$

Одержаний результат є слушним у випадку, коли усі доданки додатні. Якщо вони усі від'ємні, можна користуватися одержаними результатами, розуміючи в них під x_i їхні абсолютні значення.

Становище різко змінюється при відніманні чисел (підсумовуванні чисел із протилежними знаками).

Отже, нехай $y = x_1 - x_2$; ($x_1 > x_2 > 0$). Повторюючи попередні міркування, одержимо

$$\delta_y = \delta + \frac{\Delta_{x_1} + \Delta_{x_2}}{|x_1 - x_2|} = \delta \cdot \left[1 + \frac{|x_1 + x_2|}{|x_1 - x_2|}\right] = 2\delta \cdot \frac{x_1}{x_1 - x_2}.$$

Як бачимо, похибка при відніманні чисел більше за похибку при їхньому додаванні у $\frac{x_1}{x_1 - x_2} > 1$ разів. Якщо віднімаються близькі величини, ця похибка може бути вельми значною.

Приклад 1. Нехай $x_1 = 3,14569$, $x_2 = 3,14551$. Оцінимо похибку віднімання цих чисел.

Вважаючи, що відносна похибка округлення $\delta = 1 \cdot 10^{-6}$, із формули (9) одержимо

$$\delta_y = 2\delta \cdot \frac{x_1}{x_1 - x_2} = 2 \cdot 10^{-6} \frac{3,14569}{0,00018} = 0,035.$$

Ця похибка більша за похибку підсумовування тих самих чисел у 17 500 разів.

Приклад 6. Порівняємо похибки одержання того самого результату за двома формулами

$$y_1 = \frac{x_1 - x_2}{x_3}; \quad y_2 = \frac{x_1}{x_3} - \frac{x_2}{x_3}.$$

Похибки округлення при розрахунку згідно першої формули:

1) занесення x_1 у пам'ять: $\delta_{x_1} = \delta$; $\Delta_{x_1} = \delta \cdot x_1$;

2) занесення x_2 у пам'ять: $\delta_{x_2} = \delta$; $\Delta_{x_2} = \delta \cdot x_2$;

3) віднімання x_1 і x_2 :

$$\Delta_{x_1-x_2} = \Delta_{x_1} + \Delta_{x_2} = \delta \cdot (x_1 + x_2); \quad \delta_{x_1-x_2} = \delta \cdot \frac{x_1 + x_2}{|x_1 - x_2|};$$

4) занесення різниці у пам'ять $\delta_y = \delta_{x_1-x_2} + \delta = \delta \cdot \left(\frac{x_1 + x_2}{x_1 - x_2} + 1\right) = \delta \cdot \frac{2x_1}{x_1 - x_2}$;

5) занесення x_3 у пам'ять: $\delta_{x_3} = \delta$;

$$6) \text{ ділення } \delta_{(x_1-x_2)/x_3} = \delta_y + \delta_{x_3} = \delta \cdot \frac{3x_1 - x_2}{x_1 - x_2};$$

$$7) \text{ занесення результату у пам'ять } \delta_{y_1} = \delta_{(x_1-x_2)/x_3} + \delta = \delta_y + \delta_{x_3} = \delta \cdot \frac{4x_1 - 2x_2}{x_1 - x_2}.$$

Оцінка похибок розрахунку за другою формулою:

$$1) \text{ занесення } x_1 \text{ у пам'ять: } \delta_{x_1} = \delta; \quad \Delta_{x_1} = \delta \cdot x_1;$$

$$2) \text{ занесення } x_3 \text{ у пам'ять: } \delta_{x_3} = \delta;$$

$$3) \text{ ділення } x_1 \text{ на } x_3: \quad \delta_{x_1/x_3} = \delta_{x_1} + \delta_{x_3} = 2\delta;$$

$$4) \text{ занесення результату у пам'ять } \delta_1 = \delta_{x_1/x_3} + \delta = 3\delta; \quad \Delta_1 = 3\delta \cdot x_1 / x_3;$$

$$5) \text{ аналогічно похибка від другого ділення } x_2 \text{ на } x_3: \quad \delta_2 = 3\delta; \quad \Delta_2 = 3\delta \cdot x_2 / x_3;$$

$$6) \text{ віднімання: } \Delta_{x_1/x_3 - x_2/x_3} = \Delta_1 + \Delta_2 = 3\delta \cdot (x_1 + x_2) / x_3;$$

$$\delta_{x_1/x_3 - x_2/x_3} = \Delta_1 + \Delta_2 = 3\delta \cdot (x_1 + x_2) / (x_1 - x_2);$$

$$7) \text{ занесення результату у пам'ять } \delta_{y_2} = \delta_{x_1/x_3 - x_2/x_3} + \delta = \delta \cdot \frac{4x_1 + 2x_2}{x_1 - x_2}.$$

Тепер можна порівняти похибки розрахунків за цими двома формулами, поділивши похибку за другою формулою на похибку за першою:

$$\frac{\delta_{y_2}}{\delta_{y_1}} = \frac{4x_1 + 2x_2}{4x_1 - 2x_2}.$$

Якщо, наприклад, $x_1 \approx x_2$, то розрахунок за другою формулою приведе до похибки у 3 рази більшій, ніж розрахунок за першою формулою.

Приклад 3. Розглянемо відносну похибку результату обчислень, обумовлену округленням, для двох варіантів обчислення площі тонкого кільця

$$y_1 = (r + h)^2 - r^2; \quad y_2 = (2r + h) \cdot h.$$

Нехай $r = 1,750$; $h = 5,0 \cdot 10^{-3}$. Відносні похибки вихідних даних покладемо рівними нулеві. Елементарну відносну похибку при округленні приймемо рівною $\delta = 10^{-6}$.

РОЗВ'ЯЗУВАННЯ.

Розрахунки похибок за першою формулою:

$$1) \text{ при додаванні } r + h \text{ і запису результату у пам'ять відносна похибка становитиме } \delta_{r+h} = \delta;$$

$$2) \text{ при піднесенні у квадрат попередня відносна похибка подвоїться, а при запису результату додасться ще } \delta: \quad \delta_{(r+h)^2} = 2\delta + \delta = 3\delta;$$

$$\Delta_{(r+h)^2} = 3\delta \cdot (r + h)^2;$$

$$3) \text{ при записі результату множення } r^2 \text{ у пам'ять виникає похибка округлення}$$

$$\delta_{r^2} = \delta; \quad \Rightarrow \quad \Delta_{r^2} = \delta \cdot r^2;$$

4) при обчисленні різниці абсолютні похибки додаються

$$\Delta_{y1} = \Delta_{(r+h)^2} + \Delta_{r^2} = 3\delta \cdot (r+h)^2 + \delta \cdot r^2 \approx 2\delta r(2r+3h); \Rightarrow$$

$$\Rightarrow \delta_{y1} \approx \delta \frac{2r(2r+3h)}{h(2r+h)} \approx \delta \frac{2r}{h} = 10^{-6} \cdot \frac{3,5}{5 \cdot 10^{-3}} = 7 \cdot 10^{-4}.$$

Розрахунки похибок за другою формулою:

1) відносна похибка після занесення результату додавання $2r+h$ у пам'ять

$$\delta_{2r+h} = \delta;$$

2) відносна похибка після множення результату на h і занесення результату у пам'ять

$$\delta_{y2} = \delta + \delta = 2\delta = 2 \cdot 10^{-6}.$$

Відношення одержаних похибок дорівнює

$$\frac{\delta_{y1}}{\delta_{y2}} = \frac{7 \cdot 10^{-4}}{2 \cdot 10^{-6}} = 350.$$

Отже, похибка результату при обчисленні різниці двох близьких величин у 350 разів більша за похибку за формулою, яка виключає таке віднімання.

Резюмуючи, слід відзначити таку важливу особливість похибок округлення, яка відрізняє її від інших видів похибок: *похибки внаслідок округлення проміжних результатів можуть накопичуватися*, внаслідок чого підсумкова похибка збільшується із зростанням кількості здійснених операцій. З цього випливає, що основним засобом зменшення підсумкової похибки округлення є зменшення кількості обчислювальних операцій.

ВИСНОВКИ. Задля зменшення похибок результату обчислень внаслідок округлення проміжних результатів слід уживати наступних заходів:

- 1) потрібно зводити до мінімуму кількість арифметичних дій; для цього, зокрема, максимально використовувати дужки, використовувати попереднє обчислення з проміжним позначенням повторюваних виразів;
- 2) додавання чисел слід здійснювати у порядку зростання їхніх абсолютних величин;
- 3) по можливості потрібно уникати віднімання близьких величин, попередньо перетворюючи обчислювальні вирази;
- 4) якщо при обчисленнях зустрічаються різниці близьких величин, слід спочатку обчислити ці різниці, а лише потім здійснити решту операцій.

3.5. Запитання для самоперевірки

1. У чому полягає вміння раціонально організувати обчислювальний процес?
2. Які існують джерела виникнення похибок при обчисленнях?
3. Що таке "неусувні" похибки? З яких саме похибок складаються вони?
4. Що таке похибки "усікання"? похибки округлення? похибки поширення?
5. Дайте визначення поняттю абсолютної похибки, граничної абсолютної похибки наближеного числа. Що таке відносна похибка? Чим визначається відносна похибка при округленні числа?
6. Як подаються числові дані в оперативній пам'яті ЕОМ? Від чого це залежить?
7. Назвіть основні правила, що дозволяють зменшити вплив на похибку результату обчислень похибок вихідних даних.
8. Назвіть основні правила, що дозволяють зменшити вплив на похибку результату обчислень похибок округлення.
9. Порівняйте можливості внутрішнього подання числових даних у різних мовах програмування.

4. ДИНАМІЧНІ СИСТЕМИ

4.1. Основні поняття теорії динамічних систем

Динамічною системою зазвичай називають реальний об'єкт, поведінка якого з задовільною для досліджування точністю може бути описано системою звичайних диференціальних рівнянь, аргументом яких є час.

Як бачимо, визначення динамічної системи – суто математичне. Воно стосується будь-яких фізичних (механічних, електричних, теплових тощо) і навіть біологічних процесів, поведінка яких можна описати за допомогою диференціальних рівнянь.

Нагадаємо деякі визначення.

1. **Диференціальним рівнянням** прийнято називати рівняння, яке пов'язує значення деякої невідомої функції певних аргументів у деякій точці зі значенням її похідних різних порядків по цих аргументах у тій самій точці. Диференціальне рівняння містить у своєму запису невідому функцію, її похідні та незалежні змінні (аргументи). Система диференціальних рівнянь складається з кількох диференціальних рівнянь, в які входять кілька (за кількістю рівнянь) невідомих функцій та їхні похідні.

2. **Розв'язком (інтегралом) диференціального рівняння** називається *функція аргументу*, при підставленні якої у рівняння воно стає тотожністю. Процес розв'язування диференціального рівняння називають інтегруванням. Розв'язок системи диференціальних рівнянь являє собою сукупність функцій (за кількістю рівнянь), одночасне підставлення яких у рівняння обертає їх усі у тотожності.

3. **Порядок, або степінь диференціального рівняння** – це найбільший порядок похідних, що входять в нього. Порядок системи диференціальних рівнянь являє собою суму найбільших порядків похідних усіх шуканих змінних, що входять у ці рівняння.

4. **Диференціальні рівняння** поділяють на **звичайні (ЗДР)**, в які входять лише функції (і їхні похідні) від **одного** аргументу, і **рівняння з частинними похідними (РЧП)**, в яких функції залежать від **кількох незалежних змінних** (аргументів).

5. Серед звичайних диференціальних рівнянь математика виділяє так звані **лінійні диференціальні рівняння**, усі члени якого є лінійними функціями або самих шуканих змінних (за кількістю рівнянь), або їх похідних (того чи іншого порядку) за аргументом. Особливо повно розроблена теорія відшукування розв'язків **лінійних диференціальних рівнянь з постійними коефіцієнтами**, в яких усі кутові коефіцієнти відповідних лінійних залежностей є постійними величинами (тобто не залежать від аргументу).

6. У подальшому мова йтиме лише про **звичайні диференціальні рівняння**. Саме такі рівняння описують поведінка динамічних систем. Єдиним аргументом цих диференціальних рівнянь зазвичай є час. Це ми й будемо мати на увазі у подальшому.

Коли говорять про диференціальні рівняння як засіб опису поведінки реальних систем слід узяти до уваги наступне.

Будь-яке подання реального процесу чи явища у виді сукупності диференціальних рівнянь, навіть якщо воно побудовано на основі твердо встановлених законів (механіки, електрики тощо), завжди є наближеним до реального процесу, і тому може розглядатися лише як його теоретична модель. Це пов'язано з тим, що наукові закони формулюються для ідеалізованих об'єктів, якими реальні об'єкти не є. Як будь-яка модель, система диференціальних рівнянь відбиває реальну дійсність лише з деяким наближенням, яке може бути задовільним для досягнення поставленої мети дослідження, або незадовільним. В останньому випадку слід замінити теоретичну модель на більш точну, більш досконалу. Тому в теорії динамічних систем склалася наступна термінологія щодо реальних систем і їх теоретичного опису.

Реальні системи, поведінка яких із задовільною для потреб дослідження точністю може бути описано за допомогою системи лінійних диференціальних рівнянь зі сталими коефіцієнтами, називають *лінійними стаціонарними системами* (ЛСС) (англійською – Line Time-invariant Systems (LTI) – лінійними системами, інваріантними у часі).

Якщо ж задовільного опису реальних процесів у системі можна досягти лише за системою лінійних диференціальних рівнянь, в яких хоча б один коефіцієнт у членах рівняння, пропорційних невідомим змінним чи їхнім похідним, не є постійним (тобто є заданою явною функцією часу), то такі реальні системи носять назву *лінійних нестаціонарних систем* (ЛНС).

В реальних технічних системах з розвитком техніки все частіше зустрічаються випадки, коли поведінка системи неможливо описати, застосовуючи лише лінійні диференціальні рівняння, бо в реальній системі виникає низка особливостей руху, що не притаманні лінійній стаціонарній системі.

Тому в інженерній практиці великого значення набуває теоретичне дослідження саме *нелінійних систем* (НС), які описуються диференціальними рівняннями, в яких трапляються члени (сили), які нелінійно залежать від узагальнених координат і (або) узагальнених швидкостей.

Наприклад, рівняння руху гіроскопа у кардановому підвісі мають вигляд:

$$\left\{ \begin{array}{l} (J_1 + J_2 \cos^2 \beta) \ddot{\alpha} - 2J_2 \dot{\alpha} \dot{\beta} \sin \beta \cos \beta + H \dot{\beta} \cos \beta = N - R \sin \beta \\ J_3 \ddot{\beta} + J_2 \dot{\alpha}^2 \sin \beta \cos \beta - H \dot{\alpha} \cos \beta = L \\ \frac{dH}{dt} = R + M_{cm} \end{array} \right.$$

Як бачимо, одна з трьох вихідних величин α , β і H цієї системи диференціальних рівнянь – кут β повороту рамки підвісу – входить до окремих членів системи диференціальних рівнянь як аргумент тригонометричних функцій. Крім того, деякі члени містять добутки цих тригонометричних функцій, добутки їх і похідних від вихідної величини і добутки похідних. У правій частині міститься добуток вхідної величини R (момент відносно головної осі гіроскопа

сил, що діють на ротор) і функції від вихідної ($\sin \beta$). Джерелом нелінійностей можуть бути також моменти сил N , L , R і M_{cm} , які в реальних умовах, будучи моментами сил взаємодії між елементами карданова підвісу, можуть залежати або від самих кутів α , β і γ відносного повороту рамок (коли сили взаємодії мають пружний характер), або від кутових швидкостей $\dot{\alpha}$, $\dot{\beta}$ і $\dot{\gamma}$ (якщо це сили тертя), або й від тих та інших одночасно. Ці залежності також можуть бути нелінійними (наприклад, сухе чи турбулентне тертя).

Зазвичай усі нелінійності розподіляють на два великих класи, підходи до дослідження яких суттєво відрізняються. До першого класу відносять так звані *гладкі нелінійності*, які відображуються математично неперервними і диференційовними залежностями від узагальнених координат і швидкостей. До другого класу належать *суттєві нелінійності*, у яких їх математична залежність від узагальнених координат чи швидкостей або має розриви першого роду, або похідна цих залежностей по аргументах (q чи \dot{q}) не є неперервною.

Нелінійні системи з гладкими нелінійностями вивчати значно простіше, оскільки при цьому можна застосовувати численні аналітичні методи й, зокрема, подання нелінійних залежностей у виді степеневого ряду. Тому *більшість методів теоретичного дослідження нелінійних систем* розраховані саме на системи з *гладкими нелінійностями*. Зокрема, до них належить *теорія дослідження стійкості за Ляпуновим*.

У теорії динамічних систем склалася термінологія, запозичена по більшій мірі з теоретичної механіки. Так, усі окремі члени диференціальних рівнянь називаються *силами*. Члени диференціальних рівнянь, які не залежать ані від узагальнених координат, ані їх похідних (вони можуть залежати лише від часу), прийнято називати *"зовнішніми силами"*.

Позначатимемо у подальшому $q(t)$ узагальнені координати (тобто ті незалежні один від одного величини, завдання яких у сукупності повністю визначає поточне положення системи у просторі). Кількість узагальнених координат визначається кількістю ступенів вільності системи.

Лінійними системами називаються такі динамічні системи, усі члени диференціальних рівнянь яких (окрім вільних членів (зовнішніх сил)) є лінійними функціями узагальнених координат і їх похідних. Лінійна динамічна система з s степенями вільності описується у загальному випадку такою системою лінійних диференціальних рівнянь:

$$\begin{cases} a_{11}\ddot{q}_1 + a_{12}\ddot{q}_2 + \dots + a_{1s}\ddot{q}_s + b_{11}\dot{q}_1 + \dots + b_{1s}\dot{q}_s + c_{11}q_1 + \dots + c_{1s}q_s = e_1(t) \\ a_{21}\ddot{q}_1 + a_{22}\ddot{q}_2 + \dots + a_{2s}\ddot{q}_s + b_{21}\dot{q}_1 + \dots + b_{2s}\dot{q}_s + c_{21}q_1 + \dots + c_{2s}q_s = e_2(t) \\ \dots \dots = \dots \\ a_{s1}\ddot{q}_1 + a_{s2}\ddot{q}_2 + \dots + a_{ss}\ddot{q}_s + b_{s1}\dot{q}_1 + \dots + b_{ss}\dot{q}_s + c_{s1}q_1 + \dots + c_{ss}q_s = e_s(t) \end{cases} \quad (4.1)$$

Тут члени $e_k(t)$, які залежать лише від аргументу (часу) і не залежать від шуканих змінних q_k та їх похідних, презентують зовнішні сили.

Якщо використати позначення матриць:

$$\begin{aligned}
 A &= \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1s} \\ a_{21} & a_{22} & \dots & a_{2s} \\ \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{ss} \end{bmatrix}; & B &= \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1s} \\ b_{21} & b_{22} & \dots & b_{2s} \\ \dots & \dots & \dots & \dots \\ b_{s1} & b_{s2} & \dots & b_{ss} \end{bmatrix}; \\
 C &= \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1s} \\ c_{21} & c_{22} & \dots & c_{2s} \\ \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{ss} \end{bmatrix}; & q &= \begin{bmatrix} q_1 \\ q_2 \\ \dots \\ q_s \end{bmatrix}; & E(t) &= \begin{bmatrix} e_1(t) \\ e_2(t) \\ \dots \\ e_s(t) \end{bmatrix}, & (4.2)
 \end{aligned}$$

то систему лінійних диференціальних рівнянь системи, подану через узагальнені координати, можна записати у такому матричному вигляді:

$$A\ddot{q} + B\dot{q} + Cq = E(t). \quad (4.3)$$

Якщо реальну динамічну систему вдається задовільно для потреб теоретичного вивчення з заданою метою описати сукупністю рівнянь виду (4.1), то її називають *лінійною динамічною системою* (ЛС). Якщо при цьому усі матриці A , B і C системи (4.3) не змінюються з часом (тобто усі коефіцієнти в системі рівнянь (4.1) є сталими), то таку динамічну систему називають *лінійною стаціонарною системою* (ЛСС). У випадку ж, коли хоча б один з коефіцієнтів a_{ij} , b_{ij} або c_{ij} змінюється з часом, динамічна система зветься *нестационарною лінійною* (НЛС), або *параметрично збуджуваною*.

У багатьох випадках стійкість руху визначається структурою сил, під якими розуміються, як зазвичай, окремі складові диференціальні рівнянь, що описують рух.

Якщо усі нелінійні залежності, що входять у систему диференціальних рівнянь, є гладкими, тобто неперервними і необмежено диференційовними, то розкладаючи їх у степеневий ряд по узагальнених координатах і швидкостях, можна таку систему подати у виді

$$\begin{cases} a_{11}\ddot{q}_1 + \dots + a_{1s}\ddot{q}_s + b_{11}\dot{q}_1 + \dots + b_{1s}\dot{q}_s + c_{11}q_1 + \dots + c_{1s}q_s = f_1(q, \dot{q}) + e_1(t) \\ a_{21}\ddot{q}_1 + \dots + a_{2s}\ddot{q}_s + b_{21}\dot{q}_1 + \dots + b_{2s}\dot{q}_s + c_{21}q_1 + \dots + c_{2s}q_s = f_2(q, \dot{q}) + e_2(t) \\ \dots \dots = \dots \\ a_{s1}\ddot{q}_1 + \dots + a_{ss}\ddot{q}_s + b_{s1}\dot{q}_1 + \dots + b_{ss}\dot{q}_s + c_{s1}q_1 + \dots + c_{ss}q_s = f_s(q, \dot{q}) + e_s(t) \end{cases} \quad (4.4)$$

або, у матричній формі

$$A\ddot{q} + B\dot{q} + Cq = F(q, \dot{q}) + E(t). \quad (4.5)$$

де позначено

$$F(q, \dot{q}) = \begin{bmatrix} f_1(q, \dot{q}) \\ f_2(q, \dot{q}) \\ \dots \\ f_s(q, \dot{q}) \end{bmatrix}$$

вектор стовпець з так званих "нелінійних" сил, які являють собою у загальному випадку нескінченні степеневі ряди відносно змінних $q_1, q_2, \dots, q_s, \dot{q}_1, \dot{q}_2, \dots, \dot{q}_s$ з членами, не нижче другого степеня.

Вважатимемо, що матриця A є симетричною і квадратична форма

$$T = \frac{1}{2} \dot{q}^t \cdot A \cdot \dot{q} = \frac{1}{2} (a_{11} \dot{q}_1^2 + a_{22} \dot{q}_2^2 + \dots + a_{ss} \dot{q}_s^2 + 2a_{12} \dot{q}_1 \dot{q}_2 + \dots + 2a_{ik} \dot{q}_i \dot{q}_k) = \sum_{i=1}^s \sum_{k=1}^s a_{ik} \dot{q}_i \dot{q}_k; \quad (a_{ik} = a_{ki}) \quad (4.6)$$

є певнододатною.

Рівнянню (4.5) можна зіставити деяку матеріальну систему, у якій змінні q_1, q_2, \dots, q_s є координатами, їх похідні за часом $\dot{q}_1, \dot{q}_2, \dots, \dot{q}_s$ – швидкостями, а квадратична форма (4.6) у випадку механічної системи – кінетичною енергією. Часто ця квадратична форма й насправді є кінетичною енергією реальної системи, але у багатьох випадках форма (4.6) утворюється в результаті перетворень рівнянь руху.

Складові матриць-стовпців у лівій і правій частинах рівняння (4.5) можна трактувати як сили, причому у деяких випадках вони є реальними силами, а в інших – лише деякими членами рівнянь.

Надалі для простоти квадратичну форму (4.6) називатимемо кінетичною енергією (незалежно від того, розглядається механічна, електрична, електромеханічна чи інша реальна система), змінні q_k – координатами системи, їх похідні за часом \dot{q}_k – швидкостями, матриці-стовпці $A\ddot{q}, B\dot{q}, Cq, F$ і їхні елементи – силами. при цьому сили $A\ddot{q}$ називатимемо *силами інерції*, а сили F – *нелінійними силами*. Розкладемо матриці B і C на симетричні B_d і C_p і косиметричні G і P складові (4.13):

$$B = B^t = \frac{B_1 + B_1^t}{2}; \quad G = -G^t = \frac{B_1 - B_1^t}{2}; \quad C = C^t = \frac{C_1 + C_1^t}{2}; \quad P = -P^t = \frac{C_1 - C_1^t}{2}.$$

Тепер рівняння (4.5) матиме вигляд:

$$A\ddot{q} + B_d \dot{q} + G\dot{q} + C_p q + Pq = F + E(t). \quad (4.7)$$

Сили $C_p q$, які є пропорційними координатам, з симетричною матрицею коефіцієнтів $C_p = [c_{ik}]$, називають *потенціальними* або *консервативними*. Як правило, вони є просто лінійними частинами реальних потенціальних сил тяжіння, пружності тощо (нелінійні частини потенціальних сил входять до матриці F).

Квадратичну форму

$$\Pi = \frac{1}{2} q^t \cdot C \cdot q \quad (4.8)$$

назвемо *потенціальною енергією* системи. Насправді вона є у більшості випадків лише частиною реальної потенціальної енергії.

Складемо за допомогою симетричної матриці $B_d = [b_{ik}]$ квадратичну форму

$$\Phi = \frac{1}{2} \dot{q}^t \cdot B_d \cdot \dot{q}. \quad (4.9)$$

Якщо ця функція є невід'ємною, то вона називається *функцією розсіювання енергії* або *дисипативною функцією Релея*. Відповідні сили $B_d \dot{q}$ у цьому випадку зветься *дисипативними силами*. Якщо квадратична форма Φ є не просто додатною, а певнододатною, то *дисипація* називається *повною*, у противному разі – *неповною*. Нарешті, якщо функція Φ може набувати від'ємні значення, то серед сил, що входять до матриці $B_d \dot{q}$, існують "*прискорювальні*" сили. Зазвичай дисипативні сили виникають природним шляхом при русі тіл у середовищі, яке чинить опір, в електричних колах при наявності омичного опору тощо. Прискорювальні сили, як правило, утворюються за допомогою спеціальних пристроїв.

Сили $G\dot{q}$, які лінійно залежать від швидкостей і коефіцієнти при яких утворюють кососиметричну матрицю $G = [g_{ik}]$, називаються *гіроскопічними силами*. Найчастіше ці сили зустрічаються у системах, що містять гіроскопи, але можуть існувати і в системах без гіроскопів.

Сили Pq , які лінійно залежать від координат і коефіцієнти при яких утворюють кососиметричну матрицю $P = [p_{ik}]$, не мають точно встановленої назви. В літературі їх називають *циркуляційними*, *неконсервативними*, силами *радіальної корекції*, *псевдогіроскопічними*. У подальшому називатимемо їх неконсервативними, пам'ятаючи, що термін цей не зовсім точний, бо всі сили, окрім потенціальних, не є консервативними.

Приклад. Розглянемо гіроскоп у кардановому підвісі, зображений на рис. 4.1.

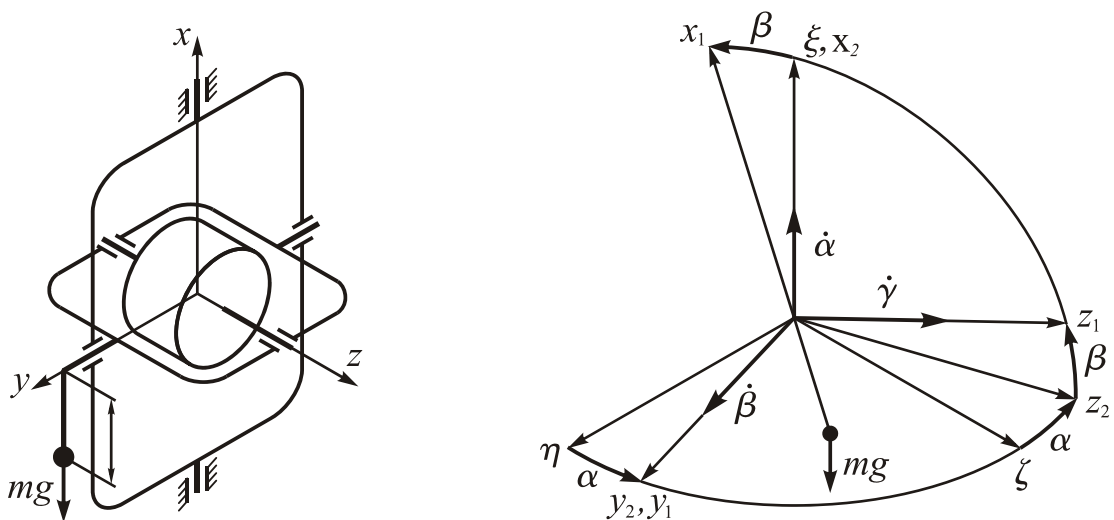


Рис. 4.1. ГКП зі зміщеним центром мас

Його рівняння були наведені раніше і мають вид:

$$\left\{ \begin{array}{l} (J_1 + J_2 \cos^2 \beta) \ddot{\alpha} - 2J_2 \dot{\alpha} \dot{\beta} \sin \beta \cos \beta + H \dot{\beta} \cos \beta = N - R \sin \beta \\ J_3 \ddot{\beta} + J_2 \dot{\alpha}^2 \sin \beta \cos \beta - H \dot{\alpha} \cos \beta = L \\ \frac{dH}{dt} = R \end{array} \right.$$

Припустимо:

1) моменти сил, що діють на ротор гіроскопа навколо головної осі врівноважені:
 $R \equiv 0$;

2) вздовж зовнішньої і внутрішньої осей карданова підвісу діють моменти сил в'язкого тертя

$$N_1 = -f_2 \dot{\alpha}; \quad L_1 = -f_1 \dot{\beta};$$

3) вздовж зовнішньої осі підвісу діє момент сил міжрамкової корекції, пропорційний куту відхилення внутрішньої рамки (для цього, очевидно, цей кут має вимірюватися датчиком кута, а електричний сигнал з цього датчика має подаватися до входу датчика моментів на зовнішній осі підвісу, який й утворить момент сил вздовж цієї осі):

$$N_2 = k\beta;$$

4) центр мас гіроскопа зміщений вздовж осі, перпендикулярній як внутрішній осі підвісу, так й головній осі гіроскопа; в результаті відносно внутрішньої осі підвісу утворюється момент сил тяжіння

$$L_2 = -mgl \sin \beta;$$

5) кути α і β є малими

$$x = \alpha; \quad y = \beta.$$

Лінеаризуючи початкові рівняння, одержимо

$$\begin{cases} (J_1 + J_2) \ddot{x} + f_2 \dot{x} + H_0 \dot{y} - ky = F_1(y, \dot{x}, \dot{y}) \\ J_3 \ddot{y} + f_1 \dot{y} - H_0 \dot{x} + mgl \cdot y = F_2(y, \dot{x}, \dot{y}) \end{cases}$$

Тут $F_1(y, \dot{x}, \dot{y})$ і $F_2(y, \dot{x}, \dot{y})$ – нелінійні сили.

Позначаючи

$$X = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix},$$

подамо цю систему диференціальних рівнянь у матричній формі:

$$A\ddot{X} + B\dot{X} + CX = F(X),$$

де $A = \begin{bmatrix} J_1 + J_2 & 0 \\ 0 & J_3 \end{bmatrix}$ – симетрична матриця коефіцієнтів інерції;

$$B = \begin{bmatrix} f_2 & H_0 \\ -H_0 & f_1 \end{bmatrix}; \quad C = \begin{bmatrix} 0 & -k \\ 0 & mgl \end{bmatrix}.$$

Виділимо сили демпфірування, гіроскопічні, потенціальні і неконсервативні:

– матриця демпфірувальних сил

$$B_d = \frac{1}{2}(B + B^t) = \begin{bmatrix} f_2 & 0 \\ 0 & f_1 \end{bmatrix};$$

– матриця гіроскопічних сил

$$G = \frac{1}{2}(B - B^t) = \begin{bmatrix} 0 & H_0 \\ -H_0 & 0 \end{bmatrix};$$

– матриця консервативних сил

$$C_p = \frac{1}{2}(C + C^t) = \begin{bmatrix} 0 & -\frac{k}{2} \\ -\frac{k}{2} & mgl \end{bmatrix};$$

– матриця сил радіальної корекції

$$P = \frac{1}{2}(C - C^t) = \begin{bmatrix} 0 & -\frac{k}{2} \\ \frac{k}{2} & 0 \end{bmatrix}.$$

Функція Релея у цьому випадку має вигляд $\Phi = \frac{1}{2}(f_2 \cdot \dot{x}^2 + f_1 \cdot \dot{y}^2)$. Вона є певно-додатною, і тому нею відображуються сили повної дисипації

4.2. Основи програмного моделювання динамічних систем

4.2.1. Нормальна форма Коші диференціальних рівнянь, фазові змінні

Більшість методів теоретичного дослідження систем диференціальних рівнянь спирається на подання цієї системи у виді сукупності диференціальних рівнянь першого порядку, розв'язаних відносно похідних, тобто такого виду:

$$\begin{cases} \frac{dy_1}{dt} = Z_1(y_1, y_2, \dots, y_n, t) \\ \frac{dy_2}{dt} = Z_2(y_1, y_2, \dots, y_n, t), \\ \dots \\ \frac{dy_n}{dt} = Z_n(y_1, y_2, \dots, y_n, t) \end{cases} \quad (4.10)$$

де n – порядок системи диференціальних рівнянь, а $Z_k(y_1, y_2, \dots, y_n, t)$, де $k = 1, 2, \dots, n$, є довільними (у тому числі – нелінійними) функціями вказаних аргументів. Форму (4.10) системи диференціальних рівнянь називають нормальною формою Коші.

До такого виду може бути зведена будь-яка система диференціальних рівнянь. Але таке приведення зв'язано з введенням додаткових (по відношенню до узагальнених координат) змінних. Тому ця операція не є однозначною. І но-

рмальних форм Коші однієї системи диференціальних рівнянь може бути безліч.

Змінні y_1, y_2, \dots, y_n , які дозволяють подати задану систему диференціальних рівнянь у формі Коші, називають *змінними стану* системи, або *фазовими змінними* системи. Їх сукупність утворює *стан*, або *фазу системи*. Характерною особливістю стану (фази) системи є те, що його завдання у початкову мить повністю визначає усе подальше поведження системи (тобто значення цього стану у подальші моменти часу).

Умовний математичний простір, в якому координатами є фазові змінні (змінні стану), зазвичай називають *фазовим простором* (простором стану). Конкретна фаза (стан) системи відображується певною точкою у цьому просторі, яка отримала назву *зображувальної точки*. При русі системи зображувальна точка змінює своє положення у фазовому просторі. Геометрично при своєму русі зображувальна точка (яка повністю характеризує стан системи у поточний момент часу) описує у фазовому просторі траєкторію, яку називають *фазовою траєкторією*. Вид цієї фазової траєкторії і напрямок руху вдовж неї зображувальної точки дає можливість скласти уявлення про рух системи в околі незбуреного руху (якому, як відомо, відповідає початок координат у фазовому просторі).

У відповідності до (4.10) значення функцій Z_k , що розташовані у правих частинах рівнянь, визначають поточні значення проєкцій $v_k = \frac{dy_k}{dt}$ *фазової швидкості* v на осі координат фазового простору.

4.2.2. Чисельне інтегрування диференціальних рівнянь

З того, що поведження динамічних систем описується диференціальними рівняннями, випливає, що основним методом програмного моделювання таких систем є чисельне інтегрування відповідних диференціальних рівнянь.

Чисельне інтегрування диференціальних рівнянь зводиться до приведення вихідної системи диференціальних рівнянь до нормальної форми Коші і організації циклічного покрокового процесу, всередині якого обчислюються значення фазових змінних на наступному кроці за відомими їх значеннями на попередньому кроці. Це здійснюється за допомогою спеціальної програми, яка зазвичай зветься розв'язувачем.

Основні розв'язувачі реалізують чисельне інтегрування так званими методами Рунге-Кутта. Кожен з методів Рунге-Кутта є однокроковим, тобто здійснює обчислення значень вектора y_{m+1} фазових змінних на наступному кроці, користуючись відомим значенням y_m цього вектора лише на попередньому кроці. Загальна формула обчислення нового значення є такою:

$$y_{m+1} = y_m + h F(t_m; y_m),$$

де h – значення кроку інтегрування; t_m – значення часу на попередньому кроці; $F(t_m; y_m)$ – деяка функція-вектор правих частин системи диференціальних рівнянь, яка інтегрується (різна для різних методів Рунге-Кутта). Формули, що визначають значення цієї функції наведені у таблиці 4.1.

Таблиця 4.1. Методи Рунге-Кутта

$$y_{m+1} = y_m + h F(t_m; y_m)$$

k – порядок метода	Формула метода	Допоміжні величини	Назва метода
1	$F=k_1$	$k_1=Z(t_m; y_m)$	Ейлера
2	$F=(k_1+k_2)/2$	$k_1=Z(t_m; y_m);$ $k_2=Z(t_m+h; y_m+hk_1)$	модифікований Ейлера
2	$F=Z(t_m+h/2; y_m+hk_1/2)$	$k_1=Z(t_m; y_m)$	
3	$F=(k_1+4k_2+k_3)/6$	$k_1=Z(t_m; y_m);$ $k_2=Z(t_m+h/2; y_m+hk_1/2);$ $k_3=Z(t_m+h; y_m+h(2k_2-k_1))$	Хойне
3	$F=(k_1+3k_3)/4$	$k_1=Z(t_m; y_m);$ $k_2=Z(t_m+h/3; y_m+hk_1/3);$ $k_3=Z(t_m+2h/3; y_m+2hk_2/3)$	
4	$F=(k_1+2k_2+2k_3+k_4)/6$	$k_1=Z(t_m; y_m);$ $k_2=Z(t_m+h/2; y_m+hk_1/2);$ $k_3=Z(t_m+h/2; y_m+hk_2/2);$ $k_4=Z(t_m+h; y_m+hk_3)$	Рунге-Кутта
4	$F=(k_1+3k_2+3k_3+k_4)/8$	$k_1=Z(t_m; y_m);$ $k_2=Z(t_m+h/3; y_m+hk_1/3);$ $k_3=Z(t_m+2h/3; y_m+h(k_2-k_1/3));$ $k_4=Z(t_m+h; y_m+h(k_1-k_2+k_3))$	

Порядок метода чисельного інтегрування – це показник k степеня у степеневої залежності глобальної похибки Δ_i (i – номер метода) усікання цього методу від кроку h інтегрування

$$\Delta_i = a_i \cdot h^k,$$

де a_i – деякий коефіцієнт. Він показує як зменшується похибка чисельного інтегруванням зі зменшенням кроку інтегрування. Якщо, наприклад, зменшити крок інтегрування у 10 разів, то це приведе до зменшення похибки методу інтегрування

- для методу першого порядку – у 10 разів;
- для методу другого порядку – у 100 разів;
- для методу третього порядку – у 1000 разів;
- для методу четвертого порядку – у 10 000 разів.

Характерною рисою методів чисельного інтегрування є їх чисельна нестійкість при перевищенні величини кроку інтегрування деякої порогової величини H . Це означає, що якщо встановити крок інтегрування вищим за це поро-

гове значення ($h \geq H$), то результат, одержаний чисельним інтегруванням, не буде мати нічого загального з істинним розв'язком диференціального рівняння, швидко віддаляючись від нього у процесі інтегрування.

Величина цього порогового значення кроку залежить не від методу інтегрування, а від власних властивостей системи диференціальних рівнянь, яка інтегрується. Для ЛСС порогова величина кроку інтегрування дорівнює значенню мінімальної сталої часу цієї системи (тобто величині, яка є зворотною максимуму за модулем кореню характеристичного рівняння системи).

Для гарантованої працездатності метода чисельного інтегрування крок інтегрування має бути принаймні у десять разів менше порогового значення.

Найбільш розвинутою комп'ютерною системою з точки зору організації процесів чисельного інтегрування диференціальних рівнянь є **Matlab**. Особливо зручно використовувати можливості пакета прикладних програм **Simulink**, який входить до складу цієї системи. Застосування пакета **Simulink** дозволяє автоматизувати процеси обрання величини кроку інтегрування і організації циклу чисельного інтегрування, значно розширити кількість застосовуваних програм-розв'язувачів, у тому числі – для так званих "жорстких" систем диференціальних рівнянь.

Приведення системи первісних диференціальних рівнянь математичної моделі до нормальної форми Коші є обов'язковим етапом складання програмної моделі для чисельного інтегрування цієї системи. Усі розв'язувачі, тобто програми, що здійснюють чисельне інтегрування диференціальних рівнянь спираються на попереднє подання цих рівнянь у формі Коші, бо на кожному кроці інтегрування вони звертаються до процедури обчислення правих частин, тобто функцій $Z_k(y_1, y_2, \dots, y_n, t)$.

Тому перед складанням програми, яка здійснює чисельне інтегрування диференціальних рівнянь, потрібно виконати наступні дії:

- 1) привести первісні рівняння до нормальної форми Коші (4.10);
- 2) скласти програму (процедуру), яка б обчислювала значення функцій правих частин Z_k приведених рівнянь по заданих значеннях вектора y_1, y_2, \dots, y_n змінних стану і часу t .

У випадку використання системи **Matlab** цю процедуру слід зберегти на диску як М-файл з певним ім'ям, наприклад, PravDR.m.

У середовищі **Matlab** задля здійснення чисельного інтегрування передбачені дві процедури **ode23** і **ode45** з автоматичним обчисленням кроку інтегрування на кожному кроці (див. п. 6.4.1, глава 2).

У пакеті **Simulink** системи **Matlab** для цієї мети у розпорядженні користувача є тринадцять процедур чисельного інтегрування (розв'язувачів) – шість з фіксованим розміром кроку інтегрування і сім – з автоматичним обчисленням цього кроку (див. п. 8.1.1).

Якщо математична модель системи вже відома, тобто складена відповідна система з s диференціальних рівнянь загального порядку n відносно s невідомо-

мих шуканих змінних x_k ($k = 1, 2, \dots, s$), то найбільш простим способом одержання рівнянь у формі Коші є такий:

1) позначити усі s шукані змінні як перші s змінні стану:

$$y_1 = x_1; \quad y_2 = x_2; \quad \dots \quad y_s = x_s;$$

2) рештою $(n - s)$ змінними стану позначити усі похідні від первинних шуканих змінних x_k , за виключенням похідної найбільш високого порядку, яка зустрічається у первинних рівняннях.

Як приклад, розглянемо застосування цього способу до рівнянь руху гіроскопа у кардановому підвісі

$$\begin{cases} (J_1 + J_2 \cos^2 \beta) \ddot{\alpha} - 2J_2 \dot{\alpha} \dot{\beta} \sin \beta \cos \beta + H \dot{\beta} \cos \beta = N - R \sin \beta \\ J_3 \ddot{\beta} + J_2 \dot{\alpha}^2 \sin \beta \cos \beta - H \dot{\alpha} \cos \beta = L \\ \frac{dH}{dt} = R + M_{cm} \end{cases},$$

в яких моменти сил N , L , R і M_{cm} вважатимемо заданими функціями часу.

У цьому випадку є три шукані змінні – α , β і H і три відповідних рівняння ($s = 3$). Загальний порядок системи диференціальних рівнянь $n = 2 + 2 + 1 = 5$. Отже, у відповідності до зазначеного, позначимо

$$y_1 = \alpha; \quad y_2 = \beta; \quad y_3 = H; \quad y_4 = \dot{\alpha} = \frac{d\alpha}{dt}; \quad y_5 = \dot{\beta} = \frac{d\beta}{dt}.$$

Останні два позначення у сукупності з першими двома дадуть додаткові два рівняння

$$\frac{dy_1}{dt} = y_4; \quad \frac{dy_2}{dt} = y_5. \quad (4.19)$$

Решта три рівняння випливають з первинних трьох рівнянь, якщо їх розв'язати відносно старших похідних і зробити зазначену заміну змінних:

$$\begin{aligned} \frac{dy_4}{dt} &= \frac{N(t) - R(t) \sin y_2 + 2J_2 y_4 y_5 \sin y_2 \cos y_2 - y_3 y_5 \cos y_2}{J_1 + J_2 \cos^2 y_2}, \\ \frac{dy_5}{dt} &= \frac{L(t) - J_2 y_4^2 \sin y_2 \cos y_2 + y_3 y_4 \cos y_2}{J_3}, \end{aligned} \quad (4.20)$$

$$\frac{dy_3}{dt} = R(t) + M_{cm}(t).$$

Сукупність рівнянь (4.19) і (4.20) є рівняннями у формі Коші.

Порівнюючи їх з (4.10), одержимо такі вирази функцій у правих частинах рівнянь у формі Коші:

$$Z_1 = y_4;$$

$$Z_2 = y_5;$$

$$Z_3 = R(t) + M_{cm}(t);$$

$$Z_4 = \frac{N(t) - R(t) \sin y_2 + 2J_2 y_4 y_5 \sin y_2 \cos y_2 - y_3 y_5 \cos y_2}{J_1 + J_2 \cos^2 y_2};$$

$$Z_5 = \frac{L(t) - J_2 y_4^2 \sin y_2 \cos y_2 + y_3 y_4 \cos y_2}{J_3}.$$

Саме обчислення значень цих функцій має здійснюватися у процедурі PravDR.m на кожному кроці інтегрування по відомих поточних значеннях модельного часу t і змінних стану.

Привести до форми Коші можна й не складаючи попередньо і детально системи диференціальних рівнянь. Для вирішення цієї задачі прислуговуються самі закони механіки, електрики чи електромеханіки, які зазвичай мають диференціальну форму. Особливо це стосується законів механіки.

4.2.3. Експериментальне дослідження похибок чисельного інтегрування

Щоб наочно впевнитися у зазначених особливостях чисельного інтегрування диференціальних рівнянь проведемо дослідження похибок чисельного інтегрування деякого, не надто складного диференційного рівняння (але й не зовсім примітивного) за допомогою спеціальної програми. Щоб одержати можливість обчислення похибки інтегрування, потрібно порівнювати результати чисельного інтегрування з точним розв'язком диференціального рівняння. Для цього годиться лінійне диференційне рівняння зі сталими коефіцієнтами, яким описується поведінка довільної коливальної ланки, наприклад, рух твердого тіла, зв'язаного з основою пружиною і демпфером, при дії на нього сили, яка змінюється у часі за гармонічним законом зі сталою складовою. У безрозмірній формі таке рівняння може звести до вигляду

$$x'' + 2\zeta x' + x = a_0 + a_m \sin v\tau, \quad (4.21)$$

де позначкою "штрих" позначена похідна за безрозмірним часом; v – відносна частота зовнішньої дії (відношення частоти зовнішньої дії до частоти власних незгасаючих коливань).

Точний розв'язок рівняння (4.21) має вигляд:

$$x(\tau) = e^{-\zeta\tau} \left[\frac{x'_0 + \zeta(x_0 - A_0 - A_C) - A_S v}{\sqrt{1-\zeta^2}} \cdot \sin(\sqrt{1-\zeta^2} \cdot \tau) + (x_0 - A_0 - A_C) \cdot \cos(\sqrt{1-\zeta^2} \cdot \tau) \right] + A_0 + A_S \sin v\tau + A_C \cos v\tau, \quad (4.22)$$

де

$$A_0 = a_0; \quad A_S = \frac{1-v^2}{(1-v^2)^2 + 4\zeta^2 v^2} a_m; \quad A_C = -\frac{2\zeta v}{(1-v^2)^2 + 4\zeta^2 v^2} a_m. \quad (4.23)$$

Введемо нові змінні

$$y_1 = x; \quad y_2 = x'. \quad (4.24)$$

Тоді у нормальній формі Коші рівняння (4.21) набуде вигляду:

$$\begin{cases} \frac{dy_1}{d\tau} = y_2 \\ \frac{dy_2}{d\tau} = -2\zeta y_2 - y_1 + a_0 + a_m \sin \nu\tau \end{cases} \quad (4.25)$$

Кінцевою метою утворення програми є дослідження за її допомогою похибок інтегрування диференційного рівняння одним (за вибором) з однокрокових методів чисельного інтегрування.

Це можна зробити суто програмно. Для цього потрібно, по-перше, створити процедуру обчислення поточних значень правих частин рівняння (4.25), по-друге, – розробити процедуру відповідного однокрокового методу, а, по-третє, – розробити головну програму, в якій би організовувався цикл покрокового звертання до процедури метода інтегрування і формувалися масиви даних про проінтегрований процес, обчислювалася масив значень точного розв’язку (4.22) за тих самих значень і його похибки.

Далі (рис. 4.2...4.4) наведені результати роботи утвореного комплексу програм для наступних значень параметрів рівняння: $\zeta = 0,1$; $a_0 = 0$; $a_m = 1$; $\nu = 0,2$.

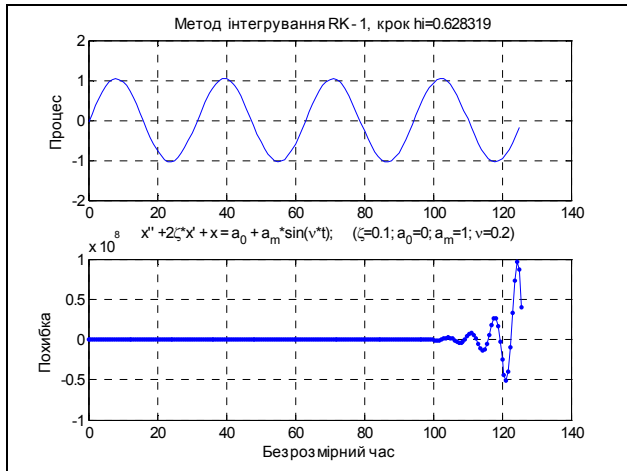
Надалі T_0 буде позначати єдину сталу часу системи. У розглядуваному випадку коливальної ланки, що описується рівнянням (4.21), яка є періодом власних незагасаючих коливань і дорівнює $T_0 = 2\pi \approx 6,28$. Крок інтегрування встановлюватимемо як частку цієї сталої часу системи. На рис. 4.2 встановлено $h = 0,1 \cdot T_0$, на рис. 4.3 – $h = 0,01 \cdot T_0$, а на рис. 4.4 – $h = 0,001 \cdot T_0$.

Початкові умови обрані такими, щоб у проінтегрованому процесі була відсутня перехідна складова, тобто у коливальній ланці одразу встановлювалися усталені вимушені коливання з частотою ν (отже – з періодом $T = 5 \cdot T_0 = 10\pi \approx 31,4$).

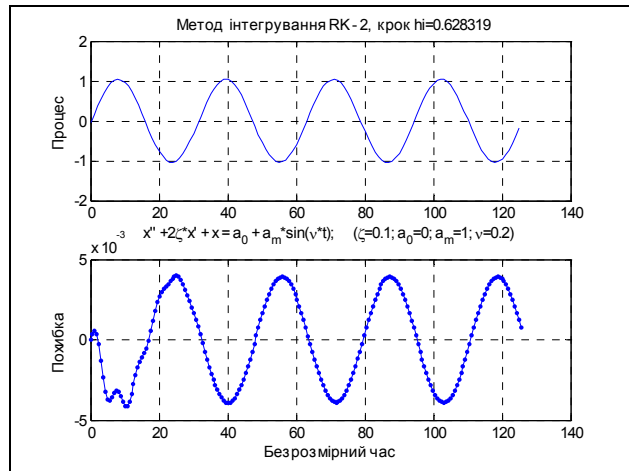
На кожному рисунку наведені результати чисельного інтегрування методами Рунге-Кутта (RK) першого, другого, третього, четвертого і п'ятого порядку.

Розглядаючи наведені рисунки, можна дійти таких висновків.

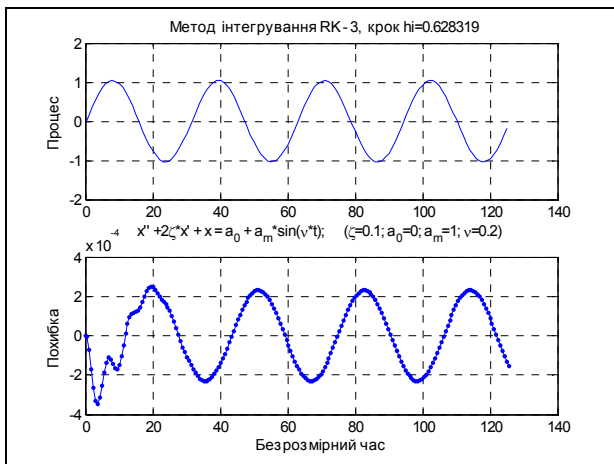
1. Похибки чисельного інтегрування мають у загальному випадку дві складові, – усталену складову, яка змінюється з частотою ν вимушених коливань і перехідну загасаючу складову, яка коливається з частотою $\omega_0 = 1$, – навіть у тому випадку, коли у самому інтегрованому процесі відсутня перехідна складова.
2. Після закінчення перехідного процесу похибка інтегрування здійснює усталені коливання з частотою змушуючої сили і незмінною амплітудою. Тому у якості узагальненої характеристики похибки можна використовувати саме амплітуду усталеної складової похибки.



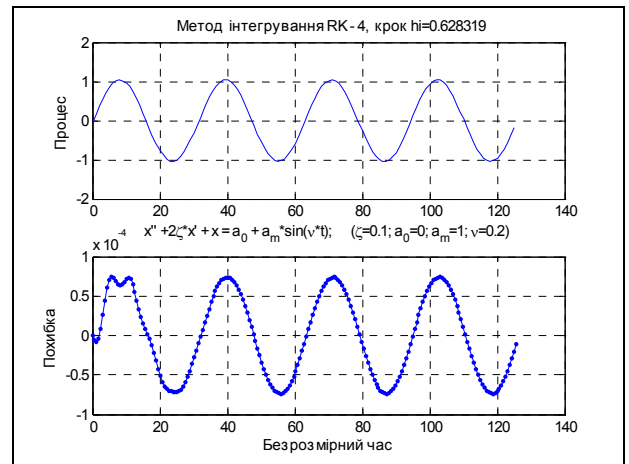
а) метод 1-го порядку



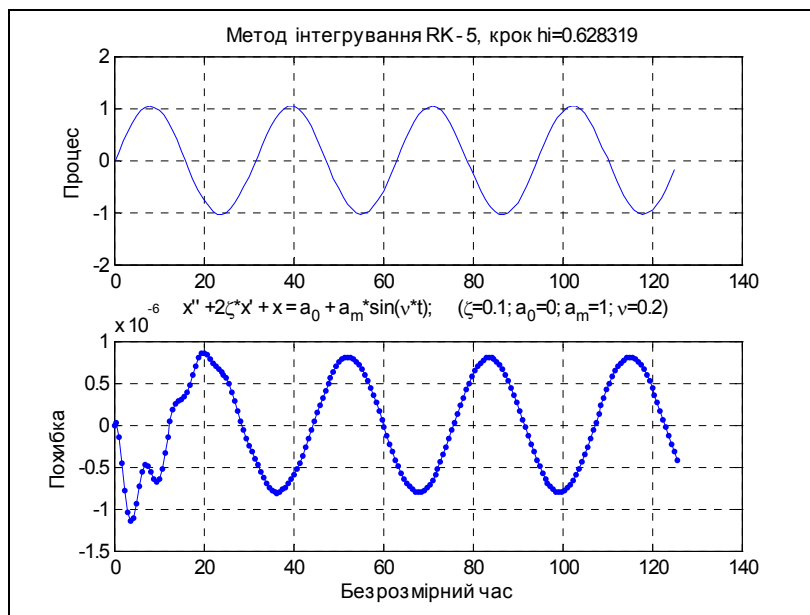
б) метод 2-го порядку



в) метод 3-го порядку

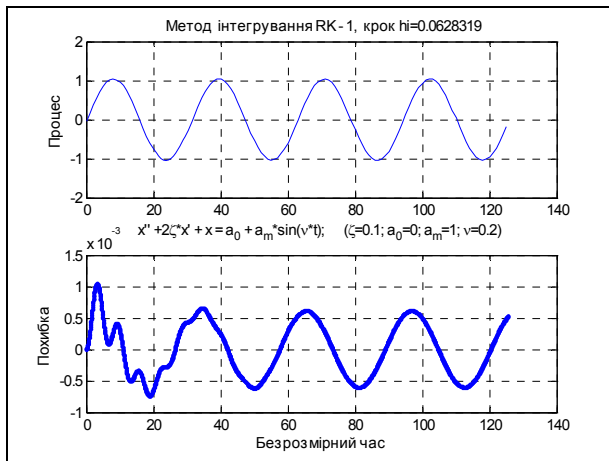


г) метод 4-го порядку

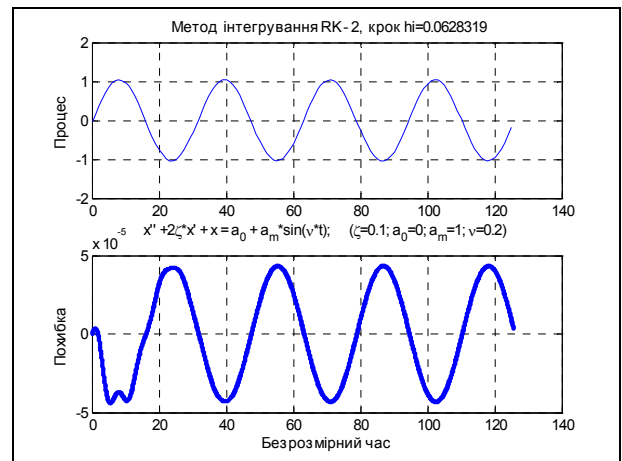


д) метод 5-го порядку

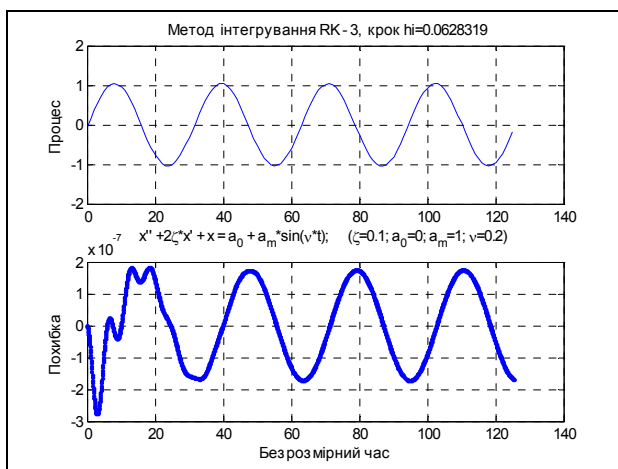
Рис. 4.2. Результати чисельного інтегрування при кроці інтегрування $h = 0,1 \cdot T_0$



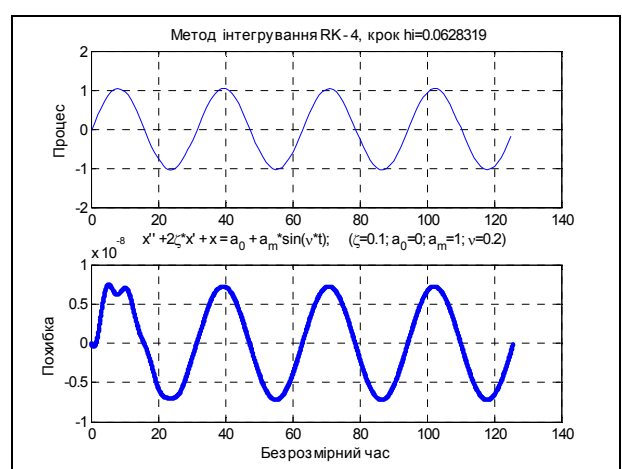
а) метод 1-го порядку



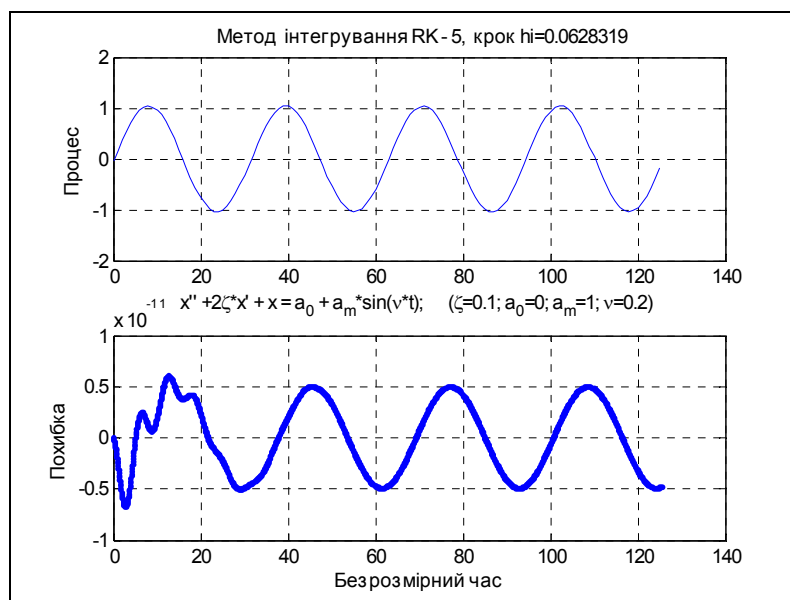
б) метод 2-го порядку



в) метод 3-го порядку

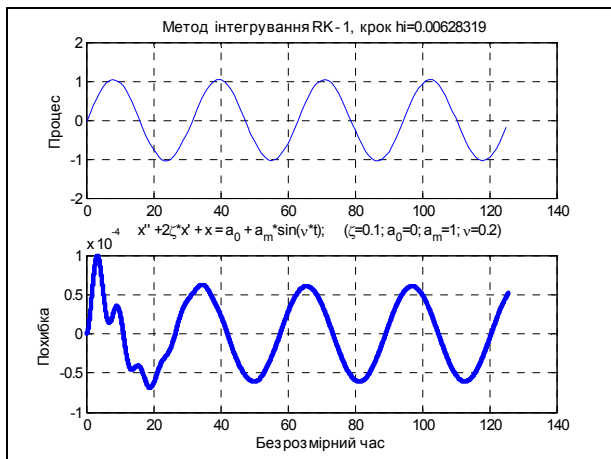


г) метод 4-го порядку

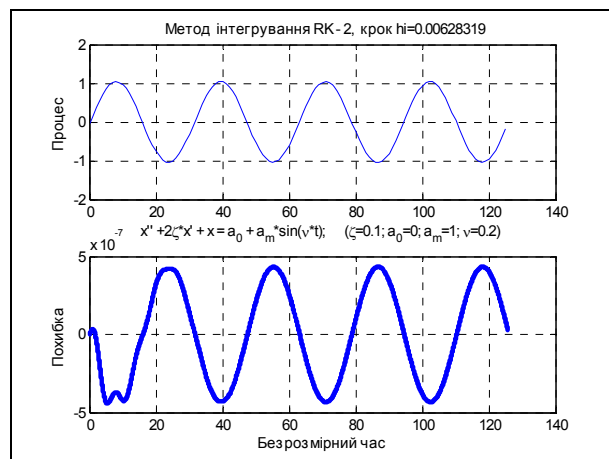


д) метод 5-го порядку

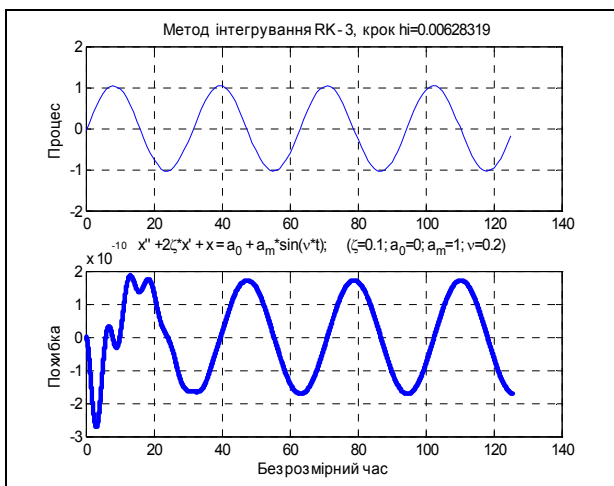
Рис. 4.3. Результати чисельного інтегрування при кроці інтегрування $h = 0.01 \cdot T_0$



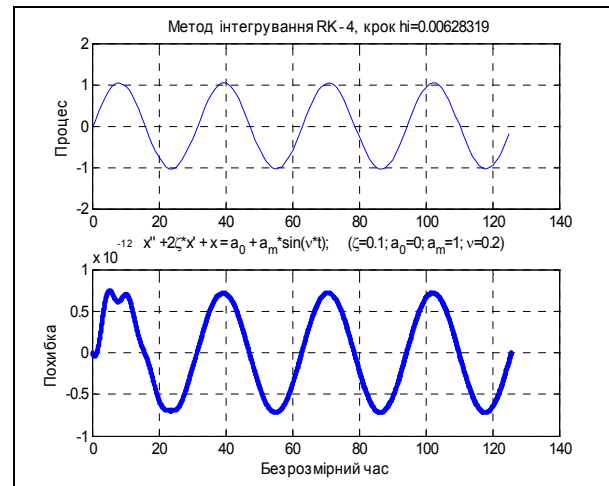
а) метод 1-го порядку



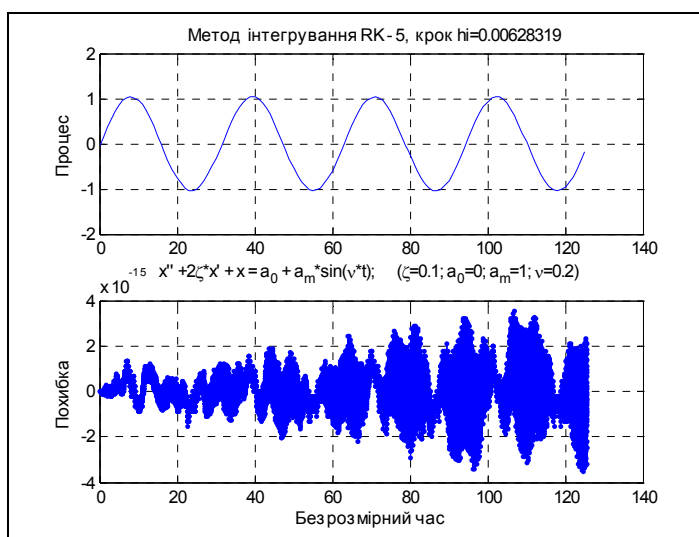
б) метод 2-го порядку



в) метод 3-го порядку



г) метод 4-го порядку



д) метод 5-го порядку

Рис. 4.4. Результати чисельного інтегрування при кроці інтегрування $h = 0,001 \cdot T_0$

3. Результати вимірів амплітуд усталеної складової в експериментах, показаних на рис. 4.2 – 4.4, наведені у таблиці 4.2.

Таблиця 4.2. Амплітуди похибок

Крок h	Порядок метода				
	перший	другий	третій	четвертий	п'ятий
$0,1 T_0$	∞	$5 \cdot 10^{-3}$	$2,3 \cdot 10^{-4}$	$8 \cdot 10^{-5}$	$8 \cdot 10^{-7}$
$0,01 T_0$	$7 \cdot 10^{-4}$	$4 \cdot 10^{-5}$	$1,8 \cdot 10^{-7}$	$8 \cdot 10^{-9}$	$5 \cdot 10^{-12}$
$0,001 T_0$	$7 \cdot 10^{-5}$	$4 \cdot 10^{-7}$	$1,8 \cdot 10^{-10}$	$8 \cdot 10^{-13}$	$< 4 \cdot 10^{-15}$

4. Порівнюючи між собою похибки кожного з методів за двох значень кроку інтегрування, що відрізняються у 10 разів, можна впевнитися, що похибка змінюється у десять в степені, що дорівнює порядку метода.
5. У випадку кроку інтегрування $h = 0,001 T_0$ і методу п'ятого порядку (рис. 4.4, д) спостерігається і якісно і кількісно дещо відмінний від описаного результат. По-перше, похибка вже не змінюється з частотою вимушених коливань, а значно частіше. По-друге, амплітуда цих коливань вже не є постійною величиною, а змінюється хаотично, випадково. По-третє, очікувана амплітуда похибки для метода п'ятого порядку має бути $5 \cdot 10^{-17}$, а спостерігається похибка на два порядки більша. Все це говорить про те, що на перший план висувається не похибка метода інтегрування, а похибка через накопичувані похибки округлення.
6. Другий різко відмінний результат має місце при інтегруванні методом першого порядку з досить великим кроком інтегрування $h = 0,1 T_0$ (рис. 4.2, а). Тут спостерігається необмежене з часом накопичення похибки до зовсім неприйнятних величин, на багато порядків перевищуючих сам процес, що інтегрується. Це так званий "вибух" метода, пов'язаний з тим, що крок інтегрування занадто наблизився до сталої часу системи, яка у розглядованому випадку дорівнює T_0 .

"Вибух" метода, хоча і пов'язаний з однією величиною – мінімальною сталою часу системи, що інтегрується, – але все ж є різним для методів різних порядків. Так, при застосуванні метода першого порядку він спостерігається вже тоді, коли крок інтегрування у 18 разів менший за цю сталу часу, у метода другого порядку – коли крок у 5 разів менший, у метода третього порядку – при кроці, меншим у 3 рази, а в методах четвертого і п'ятого порядків – коли крок всього у 2 рази менший за сталу часу системи.

4.3. Задачі і особливості моделювання гіроскопічних пристроїв

Специфіку динамічних процесів у гіроскопічних пристроях і приладах складають наступні особливості.

1. Основним режимом роботи практично усіх гіроприладів є коливання головної осі гіроскопа. Ці коливання можуть бути обумовлені різними причинами і мати різний характер – власні прецесійні і нутаціальні коливання (гірокомпас, гіромаятник); вимушені коливання, обумовлені зовнішніми і внутрішніми вібраціями і хитавицею основи; автоколивання у гіростабілізаторах; сполучення різних видів коливань, наприклад, у наземного гірокомпаса при вібрації основи.
2. У робочому режимі коливання гіроскопа здійснюються одночасно з кількома частотами. Наприклад, власні частоти гірокомпаса носять двочастотний характер (прецесійні і нутаційні коливання). Зовнішні дії є також багаточастотними.
3. Частоти коливань гіроскопа можуть значно різнитися за величиною. Так, період прецесійних коливань морського гірокомпаса приблизно дорівнює півтори години, період його нутаційних коливань складає відсотки секунд, тобто вони відрізняються на п'ять порядків. Робочий діапазон частот зовнішніх дій також велими широкий – від довгоперіодичних (порядка десятків хвилин – циркуляція корабля, віраж і фугоїдні коливання літака) до високочастотних (порядка кількох сотен герц), обумовлених вібраціями маршового двигуна рухомого транспортного засобу, на якому встановлений гіроскопічний прилад.
4. Вплив коливань зі настільки відмінними за величиною частотами на точність гіроприладу зазвичай є порівняним (того самого порядку), хоча й обумовлений різними причинами і носить різний характер. Тому неможливо досліджувати поведінку гіроприладу, відкидаючи (не враховуючи) або високочастотні, або низькочастотні коливання.

Хоча коливання з настільки різними частотами допускають, здавалося б, розділення рухів, тобто роздільне вивчення високочастотних і низькочастотних рухів, але на практиці цьому перешкоджають дві обставини:

а) через те, що рівняння руху гіроприладу є нелінійними, принцип суперпозиції тут не може бути застосованим: дія на гіроскоп сукупності низькочастотних і високочастотних збурень не дорівнюватиме сумі впливів окремо розглянутих низькочастотного і високочастотного збурень;

б) збурення, які розглядаються як сталі при дослідженні високочастотного процесу, можуть при переході до вивчення низькочастотних процесів вияв-

ляти себе як повільно змінювані, закон змінювання у часі яких визначатиметься низькочастотним рухом гіроскопа.

Описані характерні особливості динамічних процесів у гіроскопічних приладах приводять, з одного боку, до необхідності моделювати рух гіроприладу за повними рівняннями руху, враховуючи увесь спектр діючих збурень, а з іншого боку, – до необхідності вивчати і застосовувати засоби, які б дозволяли на кілька порядків зменшити час досліджування цих рівнянь на ЕОМ при збереженні припустимої точності опису динамічних процесів, що вивчаються.

Основні задачі, які вирішує проектувальник моделюванням поведінки гіроприладу на ЕОМ, можуть бути наступними:

- 1) вивчення стійкості власних коливань гіроприладу з врахуванням його нелінійних властивостей; підбір параметрів гіроприладу, які б забезпечували заданий запас стійкості; об'єктивно це відноситься до можливих автоколивань;
- 2) вивчення динамічної похибки гіроприладу, тобто тієї накопиченої сталої складової похибки вимірювання вхідної величини, яка обумовлена зовнішніми збурювальними моментами сил; підбір таких параметрів гіроприладу і корегуючих пристроїв, які б забезпечували мінімізацію цієї похибки у заданих умовах експлуатації (зовнішніх дій).

Слід додати, що часто проектувальнику немає потреби вивчати закони коливань похибок по відношенню до її середньої величини, достатньо лише знання самої цієї середньої величини (сталого складової похибки). Звідси, здавалося б, можна зробити висновок про можливість відкидання високочастотних складових руху. Це було б дійсно можливим, якщо б параметри високочастотного руху не впливали би суттєво на появу додаткових постійних складових похибок внаслідок так званого "випрямного ефекту", обумовленого нелінійністю диференціальних рівнянь гіроскопа. Ці сталі складові зазвичай є порівнянними з середньою сумарною похибкою гіроприладу і нехтувати ними не можна. Це ще раз підкреслює необхідність вивчення високочастотних рухів гіроскопа, хоча метою дослідження зазвичай є вивчення саме низькочастотної складової.

4.4. Запитання для самоперевірки

1. Яке поняття вкладається у термін "динамічна система"? Що таке "нормальна форма Коші" диференційних рівнянь?
1. Як привести систему диференційних рівнянь до форми Коші?
2. Що таке змінні стану? – фазові змінні?
3. Чим визначається кількість змінних стану?
4. У чому полягає задача чисельного інтегрування системи диференційних рівнянь?
5. Чим вирізняються однокрокові методи різних порядків?
6. Що визначає величина порядку методу чисельного інтегрування?
7. Які основні особливості однокрокових методів чисельного інтегрування ДР?
8. Що таке багатокрокові методи чисельного інтегрування ДР? Чим вони вирізняються від однокрокових?
9. Які основні особливості багатокрокових методів чисельного інтегрування ДР?
10. Перелікуйте головні переваги і недоліки однокрокових і багатокрокових методів у порівнювальному плані?
11. Які головні джерела похибок чисельного інтегрування ДР?
12. Що таке похибка обмежування? Чим вона обумовлена?
13. Як залежить величина похибки методу чисельного інтегрування ДР від величини кроку інтегрування? – від модельного часу?
14. Що таке похибка округлень? Чим вона обумовлена?
15. Як залежить величина похибки округлень при чисельному інтегруванні ДР від величини кроку інтегрування? – від модельного часу?
16. Що таке оптимальний крок чисельного інтегрування? За яких умов він має місце?
17. Що таке "вибух" методу чисельного інтегрування? За яких умов він має місце?
18. Що таке "жорстка" система диференційних рівнянь? За яких умов систему ДР можна вважати жорсткою?
19. У чому полягають труднощі чисельного інтегрування жорстких систем ДР? Чому у цьому випадку неможливо використовувати однокрокові і багатокрокові методи?
20. Що таке неявні методи чисельного інтегрування ДР? У чому їхня перевага у порівнянні з явними методами?
21. Що таке сталі часу системи диференційних рівнянь? Як їх можна визначити?
22. Як пов'язана похибка методу чисельного інтегрування з порядком цього методу?

5. ЗНАЙОМСТВО З MATLAB

Комп'ютерна система Matlab є наразі однією з найбільш пристосованих для моделювання поведінки динамічних систем і широко розповсюджена в інженерних і університетських колах завдяки видатним перевагам, до яких відносяться:

- 1) простота опанування і досяжність текстів практично усіх програмних засобів, окрім вбудованих;
- 2) велика бібліотека досяжних математичних програм, яка містить майже всі сучасні чисельні методи і функції;
- 3) можливість утворювати власні програмні засоби і навіть корегувати існуючі;
- 4) вельми зручний і пристосований для практичних потреб інженерів і науковців апарат графічного оформлення результатів обчислень;
- 5) наявність інтегрованого пакету програм **Simulink** візуального програмування, який дозволяє суттєво спростити процес утворення складних програм і автоматизувати процес організації чисельного інтегрування диференціальних рівнянь досліджуваної системи.

Ознайомленню з головними особливостями і можливостями Matlab і присвячено цей розділ.

5.1. Командне вікно

Після виклику Matlab із середовища Windows на екрані виникає вікно, подане на рис 5.1

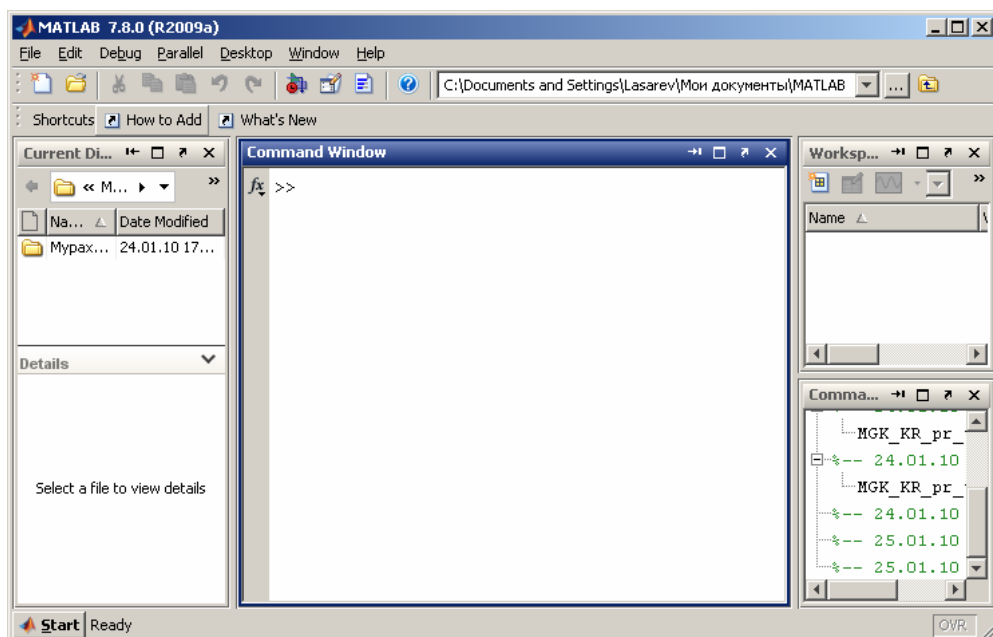


Рис. 5.1. Вид повного вікна Matlab

Якщо закрити усі бокові допоміжні підвікна, залишиться одне вікно, яке називають "командним вікном" (Command Window) середовища Matlab (рис. 5.2).

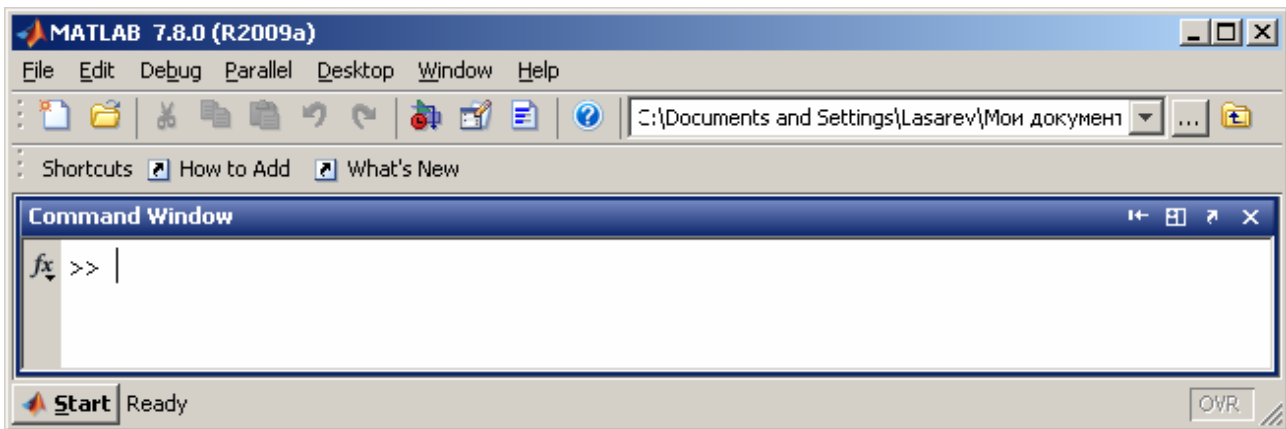


Рис. 5.2. Вид командного вікна Matlab

Це вікно є основним у Matlab. У ньому виникають символи команд, що набираються користувачем на клавіатурі дисплея, відображуються результати виконання цих команд, текст програми, яка виконується, і інформація про помилки виконання програми, розпізнані системою.

Ознакою того, що Matlab готова до сприйняття і виконання чергової команди, є поява в останньому рядку текстового поля командного вікна знака запитання (»), після якого миготить вертикальна риса.

У верхній частині вікна (під заголовком) розміщений рядок меню, в якому містяться меню *File*, *Edit*, *Debug*, *Parallel*, *Desktop*, *Windows*, *Help*. Щоб відчинити якесь меню, потрібно встановити на ньому курсор миші і клацнути її лівою кнопкою.

Тут відзначимо лише, що для виходу із середовища Matlab достатньо відчинити меню *File* і обрати у ньому команду *Exit MATLAB*, або просто зачинити командне вікно, клацнувши лівою клавішою миші, коли курсор миші встановлений на зображенні верхньої крайньої правої кнопки цього вікна (з позначенням хрестика).

5.2. Операції з числами

Комп'ютерна система Matlab може бути використана як найпотужніший калькулятор з надзвичайно розвиненими арифметичними і алгебричними можливостями.

5.2.1. Введення дійсних чисел

Введення чисел із клавіатури здійснюється по загальних правилах, прийнятих для мов програмування високого рівня:

для відділення дробової частини мантиси числа застосовується десяткова крапка (замість коми при звичайному записі);

десятковий показник числа записується у вигляді цілого числа після попереднього запису символу "e";

між записом мантиси числа й символом "e" (який відокремлює мантису від показника) не повинно бути ніяких символів, включаючи і символ пропуску.

Якщо, наприклад, ввести в командному вікні Matlab рядок (рис. 5.3) 120357.9245e-78, то після натискання клавіші <Enter> у цьому вікні виникне запис:

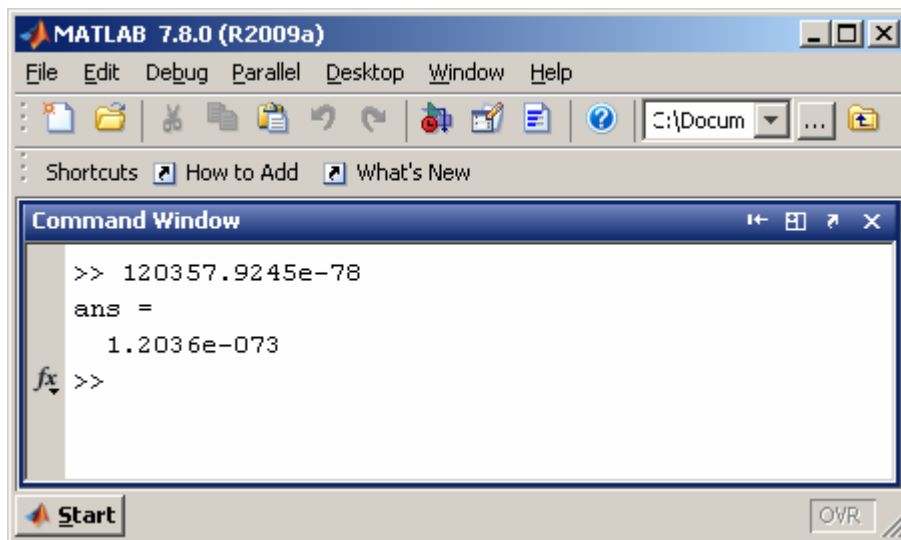


Рис. 5.3. Введення і відображення чисел

Слід відмітити, що результат виводиться у вигляді (форматі), що визначається попередньо встановленим форматом подання чисел. Цей формат може бути встановлений за допомогою команди *Preferences* меню *File* (рис. 5.4).

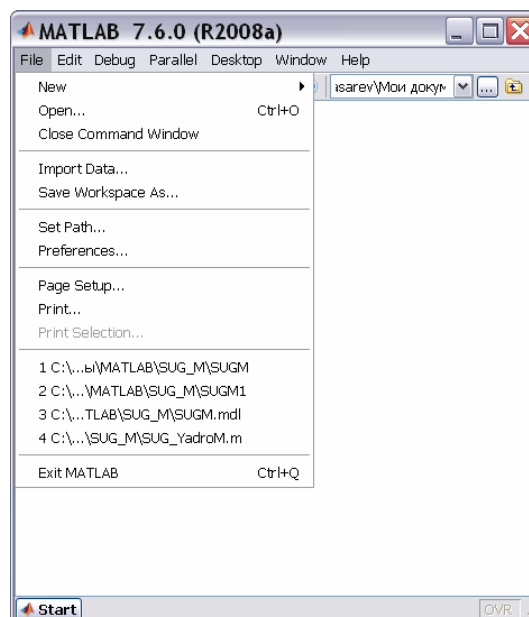


Рис. 5.4. Меню "File"

Після її виклику на екрані виникне однойменне вікно (рис. 5.5). У поділі *Command Window* одна з ділянок цього вікна має назву *Numeric Format*. Її призначено для встановлення і змінювання формату подання чисел, які виводяться в командне вікно в процесі розрахунків. Передбачені такі формати:

- Short (default)* - стислий запис (застосовується за умовчанням);
- Long* - довгий запис;
- Hex* - запис у виді шістнадцяткового числа;
- Bank* - запис до сотих часток;
- Plus* - записується тільки знак числа;
- Short E* - стислий запис у форматі із плаваючою комою;
- Long E* - довгий запис у форматі із плаваючою комою;
- Short G* - друга форма стислого запису у форматі з плаваючою комою;
- Long G* - друга форма довгого запису у форматі з плаваючою комою;
- Rational* - запис у вигляді раціонального дробу.

Обираючи за допомогою мишки потрібний вид подання чисел, можна забезпечити надалі виведення чисел у командне вікно саме в цій формі.

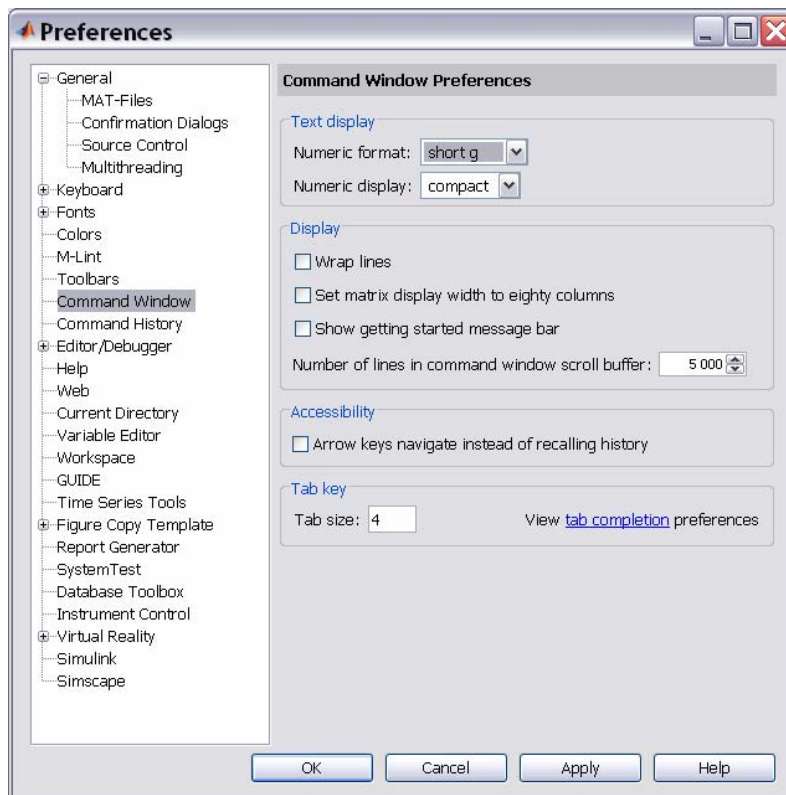


Рис. 5.5. Вікно Preferences меню "File"

Як видно з рис. 5.3, число, яке виведено на екран, не збігається з введеним. Це обумовлено тим, що встановлений за замовчуванням формат подання чисел (*Short G*) не дозволяє вивести більше 5 значущих цифр числа. Насправді введене число усередині Matlab зберігається з усіма введеними його цифрами. Наприклад, якщо обрати мишкою селекторну кнопку **Long E** (тобто установити цей формат подання чисел), то, повторюючи ті ж дії, одержимо (рис. 5.6):

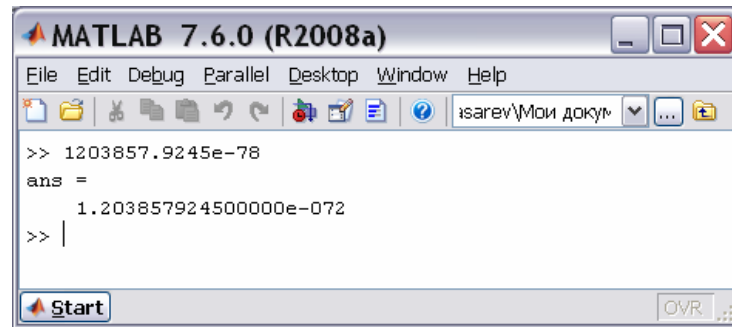


Рис. 5.6. Виведення числа у форматі Long E

де вже всі цифри відображені вірно.

Слід пам'ятати:

- уведене число й результати всіх обчислень у системі Matlab зберігаються в пам'яті ПК із відносною похибкою біля $6 \cdot 10^{-16}$ (тобто з точними значеннями в 15 десяткових розрядах);

- діапазон подання модуля дійсних чисел лежить у проміжку між 10^{-308} і 10^{+308} .

5.2.2. Найпростіші арифметичні дії

В арифметичних виразах мови Matlab використовуються такі знаки арифметичних операцій:

- | | |
|---|---------------------------|
| + | – додавання; |
| – | – віднімання; |
| * | – множення; |
| / | – ділення зліва праворуч; |
| \ | – ділення справа ліворуч; |
| ^ | – піднесення до степеня. |

Використання Matlab у режимі калькулятора може відбуватися шляхом простого запису в командний рядок послідовності арифметичних дій з числами, тобто звичайного арифметичного виразу, наприклад:

$$(4.5)^2 * 7.23 - 3.14 * 10.4$$

Якщо після введення із клавіатури цієї послідовності натиснути клавішу <Enter>, у командному вікні виникне результат виконання у виді, поданому на рис. 5.7, тобто на екран під ім'ям системної змінної **ans** виводиться результат дії останнього виконаного оператора.

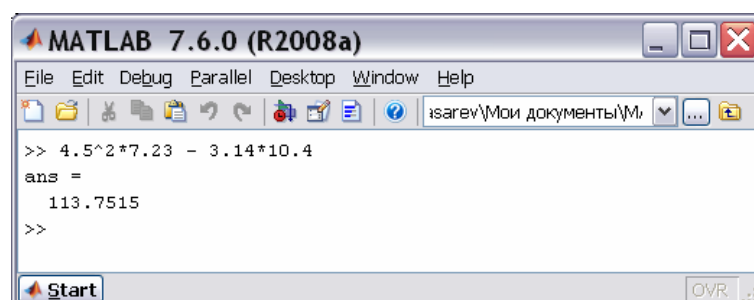


Рис. 5.7. Введення арифметичного виразу і відображення результату

Взагалі виведення проміжної інформації у командне вікно підпорядковується таким правилам:

- якщо запис оператора не закінчується символом ';' , результат дії цього оператора одразу ж виводиться в командне вікно;
- якщо оператор закінчується символом ';' , результат його дії не відображується в командному вікні;
- якщо оператор не містить знака присвоєння (=), тобто є просто записом деякої послідовності дій над числами і змінними, значення результату присвоюється спеціальній системній змінній за ім'ям **ans**;
- отримане значення змінної **ans** можна використовувати в наступних операторах обчислень, використовуючи це ім'я **ans**; при цьому варто пам'ятати, що значення системної змінної **ans** змінюється після дії чергового оператора без знака присвоєння;
- у загальному випадку форма подання результату в командне вікно має вид:

$\langle \text{Ім'я змінної} \rangle = \langle \text{результат} \rangle.$

Приклад. Нехай потрібно обчислити вираз $(25+17)*7$. Це можна зробити у такий спосіб. Спочатку набираємо послідовність **25+17** і натискаємо <Enter>. Одержуємо на екрані результат у виді **ans = 46**. Тепер записуємо послідовність **ans*7** і натискаємо <Enter>. Одержуємо **ans = 294** (рис. 5.8, а). Щоб запобігти виведення проміжного результату дії 25+17, достатньо після запису цієї послідовності додати символ ';' . Тоді будемо мати результати у виді, поданому на рис. 5.8, б.

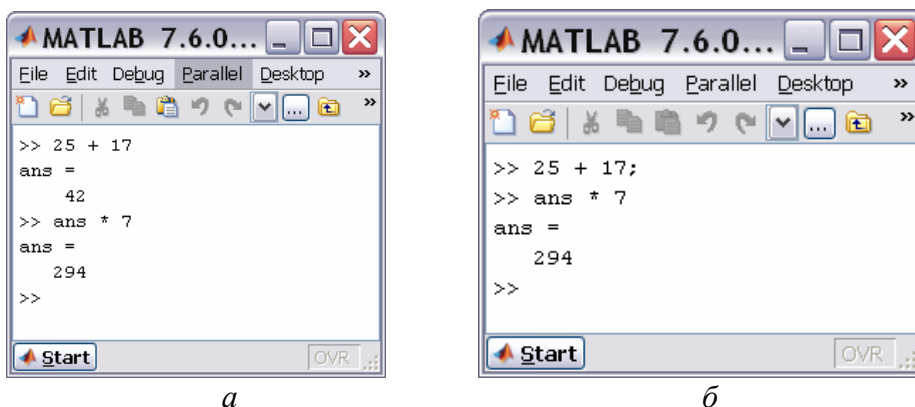


Рис. 5.8. Використання системної змінної **ans**

Особливістю Matlab як калькулятора є можливість використання імен змінних для запису проміжних результатів у пам'ять ПК. Для цього застосовується операція присвоєння, що вводиться знаком рівності '=' у відповідності зі схемою :

$\langle \text{Ім'я змінної} \rangle = \langle \text{вираз} \rangle ;$

Ім'я змінної може містити до 30 символів і повинно не збігатися з іменами функцій, процедур системи і системних змінних. При цьому система роз-

різниця великі й малі букви в змінних. Так, імена 'amenu', 'Amenu', 'aMenu' у Matlab позначають різні змінні.

Вираз справа від знака присвоювання може бути просто числом, арифметичним виразом, рядком символів (тоді ці символи потрібно укласти в апострофи) або символічним виразом. Якщо вираз не закінчується символом ';', після натискання клавіші <Enter> у командному вікні виникне результат виконання у виді :

<Ім'я змінної> = <результат>.

Наприклад, якщо ввести в командне вікно рядок 'x = 25 + 17', на екрані то виникне запис (рис. 5.9) :

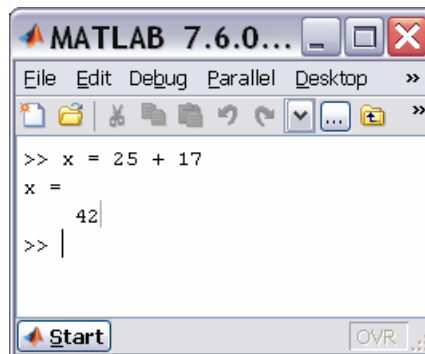


Рис. 5.9. Присвоювання значення змінній

Система Matlab має кілька імен змінних, що використовуються самою системою і входять до складу зарезервованих:

i, j – уявна одиниця (корінь квадратний з -1);

pi – число π (зберігається у виді 3.141592653589793);

inf – позначення машинної нескінченності;

NaN – позначення невизначеного результату (наприклад, типу 0/0 або inf/inf);

eps – похибка операцій над числами із плаваючою комою

ans – результат останньої операції без знака присвоювання;

realmax – максимальна величина числа, що може бути використана;

realmin – мінімальна величина числа, що може бути використана.

Ці змінні можна використовувати в математичних виразах.

5.2.3. Введення комплексних чисел

Мова системи Matlab (М-мова), на відміну від багатьох мов програмування високого рівня, містить у собі дуже просту в користуванні вбудовану арифметику комплексних чисел. Більшість елементарних математичних функцій побудовано таким чином, що аргументи припускаються комплексними числами, а результати також формуються як комплексні числа. Ця особливість мови робить її дуже привабливою й корисною для інженерів і науковців.

Для позначення уявної одиниці в мові Matlab зарезервовано два ймення i і j . Уведення із клавіатури значення комплексного числа здійснюється шляхом запису в командне вікно рядка виду:

$\langle \text{ім'я комплексної змінної} \rangle = \langle \text{значення ДЧ} \rangle + i [j] * \langle \text{значення УЧ} \rangle$,

де ДЧ – дійсна частина комплексного числа, УЧ – уявна частина. Приклад наведений на рис. 5.10:

```

MATLAB 7.6.0...
File Edit Debug Parallel Desktop >>
>> x = 1 + 2i
x =
    1.0000 + 2.0000i
>> y = -3 + 4j
y =
   -3.0000 + 4.0000i
>>
Start OVR

```

Рис. 5. 10. Введення комплексних чисел і виведення результату

З наведеного прикладу очевидно, у якому виді система виводить комплексні числа на екран (і "до друку").

5.2.4. Елементарні математичні функції

Загальна форма використання функції у Matlab така:

$\langle \text{ім'я результату} \rangle = \langle \text{ім'я функції} \rangle (\langle \text{перелік аргументів або їх значень} \rangle)$.

У мові Matlab передбачені наступні елементарні арифметичні функції.

Тригонометричні й гіперболічні функції

$\sin(Z)$	- синус числа Z ;
$\sinh(Z)$	- гіперболічний синус;
$\asin(Z)$	- арксинус (у радіанах, у діапазоні від $-\pi/2$ до $+\pi/2$);
$\asinh(Z)$	- обернений гіперболічний синус;
$\cos(Z)$	- косинус;
$\cosh(Z)$	- гіперболічний косинус;
$\acos(Z)$	- арккосинус (у діапазоні від 0 до π);
$\acosh(Z)$	- обернений гіперболічний косинус;
$\tan(Z)$	- тангенс;
$\tanh(Z)$	- гіперболічний тангенс;
$\atan(Z)$	- арктангенс (у діапазоні від $-\pi/2$ до $+\pi/2$);
$\atan2(X, Y)$	- чотириквadrантний арктангенс (кут у діапазоні від $-\pi$ до $+\pi$ між горизонтальним правим променем і променем, що проходить через точку з координатами X і Y);
$\operatorname{atanh}(Z)$	- обернений гіперболічний тангенс;
$\sec(Z)$	- секанс;

<i>sech</i> (Z)	- гіперболічний секанс;
<i>asec</i> (Z)	- арксеканс;
<i>asech</i> (Z)	- обернений гіперболічний секанс;
<i>csc</i> (Z)	- косеканс;
<i>csch</i> (Z)	- гіперболічний косеканс;
<i>acsc</i> (Z)	- арккосеканс;
<i>acsch</i> (Z)	- обернений гіперболічний косеканс;
<i>cot</i> (Z)	- котангенс;
<i>coth</i> (Z)	- гіперболічний котангенс;
<i>acot</i> (Z)	- арккотангенс;
<i>acoth</i> (Z)	- обернений гіперболічний котангенс.

Експоненціальні функції

<i>exp</i> (Z)	- експонента числа Z;
<i>log</i> (Z)	- натуральний логарифм;
<i>log10</i> (Z)	- десятковий логарифм;
<i>sqrt</i> (Z)	- квадратний корінь із числа Z;
<i>abs</i> (Z)	- модуль числа Z.

Цілочислові функції

<i>fix</i> (Z)	- округлення до найближчого цілого у бік нуля;
<i>floor</i> (Z)	- округлення до найближчого цілого у бік від'ємної нескінченності;
<i>ceil</i> (Z)	- округлення до найближчого цілого у бік додатної нескінченності;
<i>round</i> (Z)	- звичайне округлення числа Z до найближчого цілого;
<i>mod</i> (X,Y)	- цілочислове ділення X на Y;
<i>rem</i> (X,Y)	- обчислення остачі від ділення X на Y;
<i>sign</i> (Z)	- обчислення сигнум-функції числа Z (0 при Z=0, -1 при Z<0, 1 при Z>0).

5.2.5. Спеціальні математичні функції

Крім елементарних у мові Matlab передбачений цілий ряд спеціальних математичних функцій. Нижче наведений перелік і стислий зміст цих функцій. Правила звернення до них і використання користувач може відшукати в описах цих функцій, що виводяться на екран, якщо набрати команду *help* і вказати в тому ж рядку ім'я функції.

Функції перетворення координат

<i>cart2sph</i>	- перетворення декартових координат у сферичні;
<i>cart2pol</i>	- перетворення декартових координат у полярні;
<i>pol2cart</i>	- перетворення полярних координат у декартові;
<i>sph2cart</i>	- перетворення сферичних координат у декартові.

Функції Бесселя

<i>besselj</i>	- функція Бесселя першого роду;
<i>bessely</i>	- функція Бесселя другого роду;
<i>besseli</i>	- модифікована функція Бесселя першого роду;
<i>besselk</i>	- модифікована функція Бесселя другого роду.

Бета-функції

<i>beta</i>	- бета-функція;
<i>betainc</i>	- неповна бета-функція;
<i>betaln</i>	- логарифм бета-функції.

Гамма-функції

<i>gamma</i>	- гамма-функція;
<i>gammainc</i>	- неповна гамма-функція;
<i>gammaln</i>	- логарифм гамма-функції.

Еліптичні функції й інтеграли

<i>ellipj</i>	- еліптичні функції Якобі;
<i>ellipke</i>	- повний еліптичний інтеграл;
<i>expint</i>	- функція експоненціального інтегралу.

Функції похибок

<i>erf</i>	- функція похибок;
<i>erfc</i>	- додаткова функція похибок;
<i>erfcx</i>	- масштабована додаткова функція похибок;
<i>erfinv</i>	- обернена функція похибок.

Інші функції

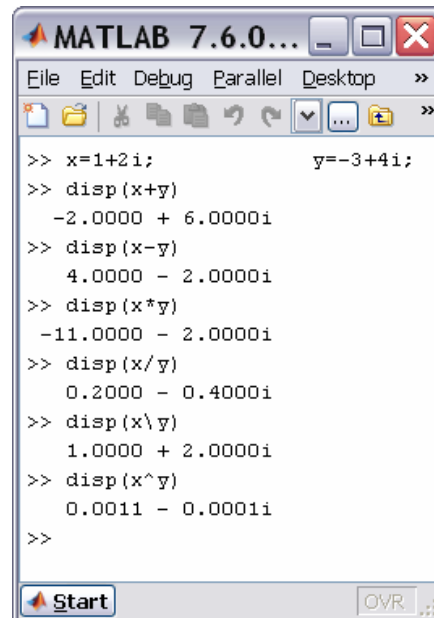
<i>gcd</i>	- найбільший загальний дільник;
<i>lcm</i>	- найменше загальне кратне;
<i>legendre</i>	- узагальнена функція Лежандра;
<i>log2</i>	- логарифм за основою 2;
<i>pow2</i>	- піднесення 2 до зазначеного степеня;
<i>rat</i>	- подання числа у виді раціонального дробу;
<i>rats</i>	- подання чисел у виді раціонального дробу.

5.2.6. Елементарні дії з комплексними числами

Найпростіші дії з комплексними числами – додавання, віднімання, множення, ділення й піднесення до степеня – здійснюються за допомогою звичайних арифметичних знаків +, -, *, /, \ і ^ відповідно.

Приклади використання наведені на рис. 5.11.

Примітка. У наведеному фрагменті використана функція *disp* (від слова 'дисплей'), яка також дозволяє виводити в командне вікно результати обчислень або деякий текст. При цьому чисельний результат, як очевидно, виводиться вже без указівки ймення змінної або *ans*.



```

MATLAB 7.6.0...
File Edit Debug Parallel Desktop >>
>> x=1+2i;          y=-3+4i;
>> disp(x+y)
-2.0000 + 6.0000i
>> disp(x-y)
4.0000 - 2.0000i
>> disp(x*y)
-11.0000 - 2.0000i
>> disp(x/y)
0.2000 - 0.4000i
>> disp(x\y)
1.0000 + 2.0000i
>> disp(x^y)
0.0011 - 0.0001i
>>
Start OVR

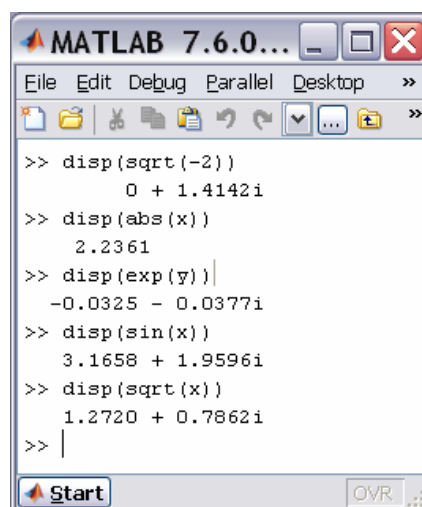
```

Рис. 5.11. Приклади використання функції *disp*

5.2.7. Функції комплексного аргументу

Практично всі елементарні математичні функції, наведені в п. 5.2.4, обчислюються за комплексних значень аргументу й одержують у результаті цього комплексні значення результату.

Завдяки цьому, наприклад, функція *sqrt* обчислює, на відміну від інших мов програмування, квадратний корінь із від'ємного аргументу, а функція *abs* при комплексному значенні аргументу обчислює модуль комплексного числа. Приклади наведені на рис. 5.12.

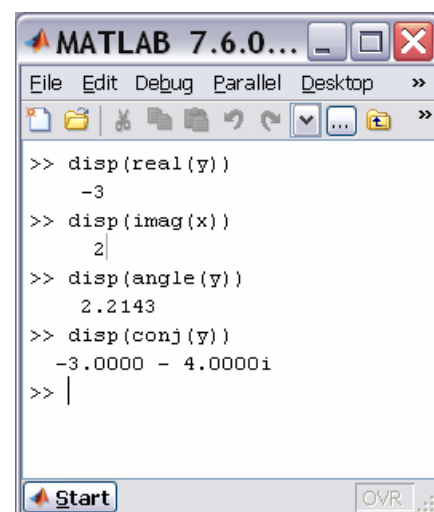


```

MATLAB 7.6.0...
File Edit Debug Parallel Desktop >>
>> disp(sqrt(-2))
0 + 1.4142i
>> disp(abs(x))
2.2361
>> disp(exp(y))
-0.0325 - 0.0377i
>> disp(sin(x))
3.1658 + 1.9596i
>> disp(sqrt(x))
1.2720 + 0.7862i
>>
Start OVR

```

Рис. 5.12. Застосування функцій з комплексним аргументом



```

MATLAB 7.6.0...
File Edit Debug Parallel Desktop >>
>> disp(real(y))
-3
>> disp(imag(x))
2
>> disp(angle(y))
2.2143
>> disp(conj(y))
-3.0000 - 4.0000i
>>
Start OVR

```

Рис. 5.13. Застосування функцій комплексного аргументу

У Matlab є кілька додаткових функцій, розрахованих тільки на комплексний аргумент:

real(Z) - виділяє дійсну частину комплексного аргументу Z;

imag(Z) - виділяє уявну частину комплексного аргументу;

angle(Z) - обчислює значення аргументу комплексного числа Z (у радіанах від $-\pi$ до $+\pi$);

conj(Z) - видає число, комплексно спряжене щодо Z.

Приклади наведені на рис. 5.13.

Крім того, у Matlab є спеціальна функція **cplxpair(V)**, яка здійснює сортування заданого вектора V із комплексними елементами у такий спосіб, що комплексно-спряжені пари цих елементів розташовуються у вихідному векторі в порядку зростання їхніх дійсних частин, при цьому елемент із від'ємною уявною частиною завжди розташовується першим. Дійсні елементи завершують комплексно-спряжені пари.

Наприклад (надалі в прикладах команди, що набираються із клавіатури, будуть написані масним шрифтом, а результат їхнього виконання – звичайним шрифтом):

```
» v = [-1, -1+2i, -5, 4, 5i, -1-2i, -5i]
v =
Columns 1 through 4
-1.0000    -1.0000 + 6.0000i    -5.0000    4.0000
Columns 5 through 7
0 + 5.0000i    -1.0000 - 6.0000i    0 - 5.0000i
» disp(cplxpair(v))
Columns 1 through 4
-1.0000 - 6.0000i    -1.0000 + 6.0000i    0 - 5.0000i    0 + 5.0000i
Columns 5 through 7
-5.0000    -1.0000    4.0000
```

Пристосованість більшості функцій Matlab до оперування з комплексними числами дозволяє значно простіше будувати обчислення з дійсними числами, результат яких є комплексним, наприклад, знаходити комплексні корені квадратних рівнянь.

5.2.8. Знайомство з програмуванням в Matlab

Робота в режимі калькулятора в середовищі Matlab, незважаючи на досить значні можливості, має істотні незручності. Неможливо повторити всі попередні обчислення й дії при нових значеннях початкових даних без повторного набирання всіх попередніх операторів. Не можна повернутися назад і повторити деякі дії, або за деякою умовою перейти до виконання іншої послідовності операторів. І взагалі, якщо кількість операторів є значною, стає проблемою налагодити правильну їхню роботу через неминучі помилки при набиранні команд. Тому складні, із перериваннями, складними переходами по певних умовах, із часто повторюваними однотипними діями обчислення, які, до того ж, необхідно проводити неодноразово при змінених первинних даних, потребують їхнього спеціального оформлення у виді записаних на диску файлів,

тобто у виді програм. Перевага програм у тому, що, унаслідок того, що вони зафіксовані у виді записаних файлів, стає можливим багаторазове звернення до тих самих операторів і до програми в цілому. Це дозволяє спростити процес налагоджування програми, зробити процес обчислень більш наочним і прозорим, а завдячуючи цьому – різко зменшити можливість появи помилок при розробці програм. Крім того, у програмах виникає можливість автоматизувати також і процес змінювання значень первісних параметрів у діалоговому режимі.

Створення програми в середовищі Matlab здійснюється за допомогою вбудованого редактора. Вікно цього вбудованого редактора виникає на екрані, якщо перед цим використано команду "M-file" із поділу *New* або обрано назву одного з існуючих M-файлів при виклику команди *Open M-file* із меню **File** командного вікна. У першому випадку вікно текстового редактора є порожнім (рис. 5.14), у другому – у ньому міститься текст викликаного M-файлу. В обох випадках вікно текстового редактора готове для введення нового тексту або коригування існуючого.

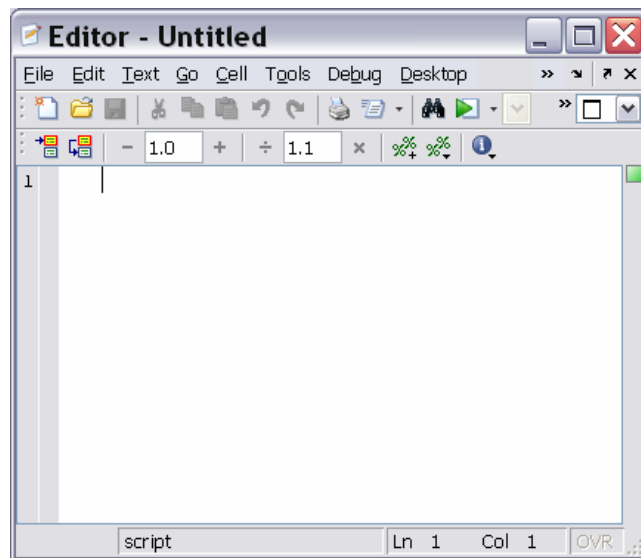


Рис. 5.14. Вікно вбудованого текстового редактора Matlab

Запис тексту програми (M-файлу) мовою Matlab має підпорядковуватися наступним правилам.

1. Зазвичай кожний оператор записується в окремому рядку тексту програми. Ознакою кінця оператора є символ `;` (він не виникає у вікні) повернення каретки й переходу на наступний рядок, який вводиться в програму при натисканні клавіші `<Enter>`, тобто при переході при записі тексту програми на наступний рядок.

2. Можна розміщувати кілька операторів в одному рядку. Тоді попередній оператор у тому ж рядку має закінчуватися символом `' ; '` або `' , '`.

3. Можна довгий оператор записувати в декілька рядків. При цьому попередній рядок оператора має завершуватися трьома крапками (`' ... '`).

4. Якщо черговий оператор не закінчується символом `' ; '`, результат його дії при виконанні програми буде виведений у командне вікно. Щоб запобігти

виведенню на екран результатів дії оператора програми, запис цього оператора в тексті програми має закінчуватися символом ' ; '.

5. Рядок програми, що починається із символу ' % ', не виконується. Цей рядок сприймається системою Matlab як *коментар*. Тому для введення коментарю в будь-яке місце тексту програми достатньо почати відповідний рядок із символу ' % '.

6. Рядки коментаря, які передують першому виконуваному (тобто такому, що не є коментарем) оператору програми, сприймаються системою Matlab як опис програми. Саме ці рядки виводяться в командне вікно, якщо в ньому набрано команду

help <ім'я файла>

7. У програмах мовою Matlab *відсутній символ закінчення тексту програми*.

8. У мові Matlab *змінні не описуються і не оголошуються*. Будь-яке нове ім'я, що зустрічається в тексті програми при її виконанні, сприймається системою Matlab як ім'я матриці. Розмір цієї матриці встановлюється при введенні значень її елементів або визначається діями по встановленню значень її елементів, описаними у попередньому операторі або процедурі. Ця особливість робить мову Matlab дуже простою у вжитку і привабливою. У мові MatLAB неможливо використання вхідної матриці або змінної, у якій попередньо не введені або обчислені значення її елементів (а значить - і визначені розміри цієї матриці). У протилежному випадку при виконанні програми Matlab з'явиться повідомлення про помилку – "Змінна не визначена".

9. Імена змінних можуть містити лише букви латинського алфавіту або цифри і мають починатися з букви. Загальна кількість символів в імені може сягати 30. В іменах змінних можуть використовуватися як великі, так і малі букви. Особливістю мови Matlab є те, що *великі й малі букви в іменах розрізняються системою*. Наприклад, символи "a" і "A" можуть використовуватися в одній програмі для позначення різних величин.

5.2.9. Запитання для самоперевірки

1. Як подаються дійсні числа при обчисленнях у системі Matlab?
6. Як змінити формат подання дійсних чисел у командному вікні?
3. Яким чином оголошуються змінні в мові Matlab?
4. Як зробити так, щоб результат дій, записаних у черговому рядку а) виводився в командне вікно; б) не виводився на екран?
5. Яку роль грає системна змінна *ans* ?
6. Які є системні змінні у Matlab? Що вони позначають і як їх використовувати?
7. Як повернути в командний рядок раніше введену команду ?
8. Які вбудовані елементарні математичні функції є у Matlab? Які з них відсутні у всіх інших мовах високого рівня?

9. В яких одиницях має бути поданий аргумент тригонометричних функцій? В яких одиницях виходить результат застосування обернених тригонометричних функцій?
10. Як виразити значення кута у градусах через радіани і навпаки?
11. Як увести значення комплексного числа й у якому виді воно виведеться на екран?
16. Як мовою Matlab забезпечити додавання, віднімання, множення, ділення й піднесення до степеня комплексних чисел?
13. Які функції роботи з комплексними числами є в мові Matlab?
14. Як складаються програми мовою Matlab, як вони записуються, як викликаються до виконання?
15. Які правила написання тексту програми мовою Matlab? Як ввести коментарі у текст програми Matlab? Яку роль відіграють закоментовані перші рядки програми Matlab?
16. Яку функцію виконує команда *disp*? Як нею користуватися? В яких випадках нею доцільно користуватися?
17. Як за допомогою функції *disp* вивести у командне вікно значення визначених змінних?
18. Як за допомогою функції *disp* вивести у командне вікно одночасно і значення змінних, і деякий супроводжуючий їх текст?
19. За допомогою яких функцій можна перетворити числове значення деякої змінної у символьний рядок, що містить це числове значення?
20. За якою формулою визначається комплексне число, якщо задані значення його модуля і аргументу?
21. Як відображується на комплексній площині комплексне число? його дійсна частина? його уявна частина? його модуль? його аргумент? Як пов'язані між собою уявна і дійсна частина, з одного боку, і модуль і аргумент – з іншого боку?
26. З допомогою якої функції Matlab можна знайти модуль комплексного числа?
23. З допомогою якої функції Matlab можна знайти аргумент комплексного числа?
24. Які елементарні вбудовані математичні функції Matlab можуть використовувати як аргументи комплексні числа?
25. Чи є у Matlab функції, які призначені для роботи лише з комплексними числами? Назвіть їх і охарактеризуйте.

5.3. Операції з векторами

Matlab є системою, спеціально призначеною для здійснювання складних обчислень із векторами, матрицями й поліномами.

Під вектором у Matlab розуміється одновимірний масив чисел, а під матрицею – двовимірний масив. При цьому за замовчуванням передбачається,

що будь-яка задана змінна є вектором або матрицею. Наприклад, окреме задане число система сприймає як матрицю розміром (1×1) , а вектор-рядок із N елементів – як матрицю розміром $(1 \times N)$.

5.3.1. Введення векторів і матриць

Початкові значення векторів можна задавати із клавіатури шляхом поелементного введення. Для цього в рядку треба спочатку вказати ім'я вектора, потім поставити знак присвоювання '=' , а далі, – відкриваючу квадратну дужку, а за нею ввести задані значення елементів вектора, відділяючи їх пропусками або комами. Закінчується рядок записом квадратної дужки, що закриває.

Наприклад, запис рядка $V = [1.2 \ -0.3 \ 1.2e-5]$ задає вектор V , що містить три елементи зі значеннями 1.2, -0.3 і $1.2e-5$ (рис. 5.15):

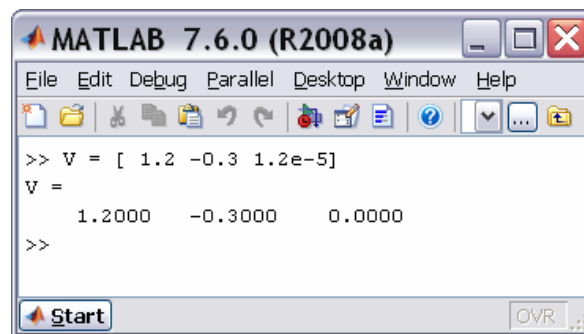


Рис. 5.15. Введення вектора

Після введення вектора система виводить його на екран. Те, що в наведеному прикладі останній елемент виведений як 0, обумовлено встановленим форматом `short`, відповідно до якого виводяться дані на екран.

Довгий вектор можна вводити частинами, які потім об'єднують за допомогою операції об'єднання векторів у рядок : $v = [v1 \ v2]$. Приклад наведений на рис. 5.16.

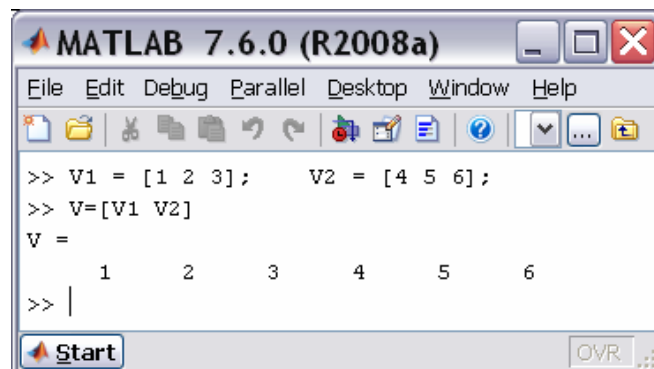


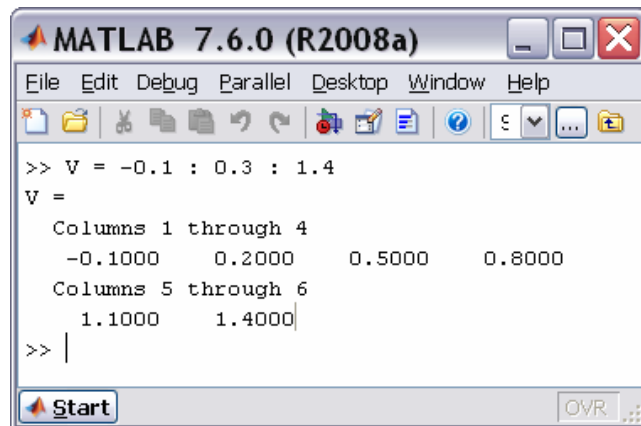
Рис. 5.16. Конкатенація (об'єднання) векторів

Мова Matlab дає користувачеві можливість скороченого введення вектора, значення елементів якого є арифметичною прогресією. Якщо

позначити: nz – початкове значення цієї прогресії (значення першого елемента вектора); kz – кінцеве значення прогресії (значення останнього елемента вектора); h – різницю прогресії (крок), то вектор можна ввести за допомогою короткого запису

$$V = nz : h : kz .$$

Наприклад, введення рядка $V = -0.1 : 0.3 : 1.4$ приведе до результату, показаному на рис. 5.17.



```

MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
>> V = -0.1 : 0.3 : 1.4
V =
Columns 1 through 4
-0.1000    0.2000    0.5000    0.8000
Columns 5 through 6
1.1000    1.4000
>>
  
```

Рис. 5.17. Введення вектора арифметичної прогресії

Якщо середній параметр (різниця прогресії) не зазначений, то він за замовчуванням приймається рівним одиниці. Наприклад, команда

```
>> -6. 1 : 5
```

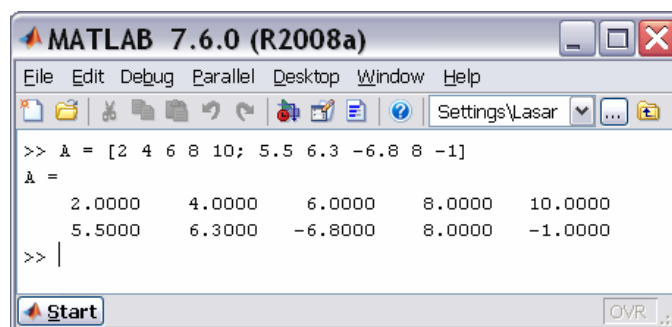
приводить до формування такого вектора

```

ans =
Columns 1 through 7
-6.1000 -1.1000 -0.1000 0.9000 1.9000 6.9000 3.9000
Column 8
4.9000
  
```

У такий спосіб уводяться вектори-рядки. *Вектор-стовпець* вводиться аналогічно, але значення елементів відокремлюються знаком ";".

Введення значень елементів матриці здійснюється в Matlab у квадратних дужках, по рядках. При цьому елементи рядка матриці один від одного відокремлюються пропуском або комою, а рядки один від одного відокремлюються знаком ";" (рис. 5.18).



```

MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
Settings\Lasar
>> A = [2 4 6 8 10; 5.5 6.3 -6.8 8 -1]
A =
2.0000    4.0000    6.0000    8.0000   10.0000
5.5000    6.3000   -6.8000    8.0000   -1.0000
>>
  
```

Рис. 5.18. Приклад введення матриці

5.3.2. Дії над векторами

Розрізняватимемо дві групи дій над векторами:

а) *векторні дії* – тобто такі, що передбачені векторним зчисленням у математиці;

б) *дії по перетворенню елементів* - це дії, що перетворюють елементи вектора, але не є операціями, дозволеними математикою з векторами.

Векторні дії над векторами

Додавання векторів. Як відомо, складатися (підсумовуватися) можуть тільки вектори однакового типу (тобто такі, які обидва є або векторами-рядками, або векторами-стовпцями), що мають однакову довжину (тобто однакову кількість елементів). Якщо X і Y є саме такими векторами, то їхню суму Z можна одержати, увівши команду $Z = X + Y$, наприклад:

```
» x = [1 2 3]; y = [4 5 6];
```

```
» v = x + y
```

```
v = 5 7 9
```

Аналогічно за допомогою арифметичного знака " - " здійснюється **віднімання векторів**, що мають однакову структуру ($Z = X - Y$).

Наприклад:

```
» v = x - y
```

```
v = -3 -3 -3
```

Транспонування вектора здійснюється застосуванням знака апострофу, що записується відразу за записом імені вектора, який транспонується. Наприклад:

```
» x'
```

```
ans =
```

```
1
```

```
2
```

```
3
```

Множення вектора на число здійснюється в Matlab за допомогою знака арифметичного множення ($*$) у такий спосіб: $Z = X*r$ або $Z = r*X$, де r - деяке дійсне число.

Приклад:

```
» v = 2*x
```

```
v = 2 4 6
```

Множення двох векторів визначено у математиці тільки для векторів однакового розміру (довжини) і лише тоді, коли один із векторів-множників є рядком, а другий - стовпчиком. Тобто, якщо вектори X і Y є рядками, то математичний зміст мають лише дві форми множення цих векторів: $U = X' * Y$ і $V = X * Y'$. Причому в першому випадку результатом буде квадратна матриця, а в другому - число.

У MatLAB множення векторів здійснюється застосуванням звичайного знака множення ($*$), який записується між множниками-векторами.

Приклад:

```
» x = [1 2 3]; y = [4 5 6];
```

```
» v = x' * y
```

```
v =
```

```

4   5   6
8  10  12
12  15  18
» v = x * y'
v = 32

```

Для *трикомпонентних векторів* у Matlab існує функція *cross*, яка дозволяє знайти *векторний добуток двох векторів*. Для цього, якщо задані два трикомпонентних вектори $v1$ і $v2$, достатньо ввести оператор

```
cross(v1, v2).
```

Приклад:

```

» v1 = [1 2 3]; v2 = [4 5 6];
» cross(v1,v2)
ans = -3 6 -3

```

На цьому перелік припустимих математичних операцій з векторами вичерпується.

Поелементне перетворення векторів

У мові Matlab є ряд операцій, які перетворюють заданий вектор в інший того ж розміру й типу, але в той же час не є математичними операціями з вектором як математичним об'єктом. Усі ці операції перетворюють елементи вектора як елементи звичайного одновимірного масиву чисел. До таких операцій належать, наприклад, усі елементарні математичні функції, наведені в розділі 5.6.4 і які залежать від одного аргументу. У мові Matlab запис, наприклад, виду $Y = \sin(X)$, де X – деякий відомий вектор, приводить до формування нового вектора Y , що має той самий тип і розмір, але елементи якого дорівнюють синусам відповідних елементів вектора-аргументу X . Наприклад:

```

» x = [-2,-1,0,1,2];
» y = sin(x)
y = -0.9093 -0.8415 0 0.8415 0.9093
» z = tan(x)
z = 6.1850 -1.5574 0 1.5574 -6.1850
» v = exp(x)
v = 0.3679 1.0000 6.7183 7.389

```

Крім цих операцій у MatLAB передбачено декілька операцій поелементного перетворення, що здійснюються за допомогою знаків звичайних арифметичних дій. Ці операції застосовуються до векторів однакового типу й розміру. Результатом їх є вектор того ж типу й розміру.

Додавання (віднімання) числа до (із) кожного елемента вектора. Здійснюється за допомогою знаку "+" ("–").

Поелементне множення векторів. Проводиться за допомогою сукупності знаків ".*", що записується між іменами векторів, які перемножуються. У результаті утворюється вектор, кожний елемент якого є добутком відповідних елементів векторів – "співмножників".

Поелементне ділення векторів. Здійснюється за допомогою сукупності знаків "./". Результат – вектор, кожний елемент якого є часткою від ділення відповідного елемента першого вектора на відповідний елемент другого вектора.

Поелементне ділення векторів в оберненому напрямку. Здійснюється за допомогою сукупності знаків "\". В результаті одержують вектор, кожний елемент якого є часткою від ділення відповідного елемента другого вектора на відповідний елемент першого вектора.

Поелементне піднесення до степеня. Здійснюється за допомогою сукупності знаків ". ^". Результат - вектор, кожний елемент якого є відповідним елементом першого вектора, піднесеним до степеня, розмір якого дорівнює значенню відповідного елемента другого вектора. Приклад :

```

» x = [1,2,3,4,5];      y = [-2,1,4,0,5];
» disp(x + 2)
   3   4   5   6   7
» disp(y - 3)
  -5  -2   1  -3   2
» disp(x. *y)
  -2   2  12   0  25
» disp(x. /y)
Warning: Divide by zero
 -0.5000   6.0000   0.7500   Inf   1.0000
» disp(x. \y)
 -6.0000   0.5000   1.3333   0   1.0000
» disp(x. ^y)
   1   2   81   1  3125

```

Вищевказані операції дозволяють дуже просто обчислювати (а потім – будувати графіки) складних математичних функцій, не використовуючи при цьому оператори циклу, тобто робити побудову графіків у режимі калькулятора.

Для цього достатньо задати значення аргументу як арифметичну прогресію так, як це було показано в поділі 5.3.1, а потім записати потрібну функцію, використовуючи знаки поелементного перетворення векторів.

Наприклад, нехай потрібно обчислити значення функції:

$$y = ae^{-hx} \sin x$$

при значеннях аргументу x від 0 до 10 із кроком 1. Обчислення масиву значень цієї функції у зазначених умовах можна здійснити за допомогою лише двох простих операторів :

```

» a = 3; h = 0.5;
» x = 0:1:10;
» y = a * exp(-h*x) . * sin(x)
y =
Columns 1 through 7
   0  1.5311  1.0035  0.0945 -0.3073 -0.2361 -0.0417
Columns 8 through 11
   0.0595  0.0544  0.0137 -0.0110

```

5.3.3. Операції з поліномами

В системі Matlab передбачені деякі додаткові можливості математичного оперування з поліномами.

Поліном (багаточлен) як функція визначається виразом :

$$P(x) = a_n \cdot x^n + \dots + a_2 \cdot x^2 + a_1 \cdot x + a_0.$$

В системі Matlab поліном задається й зберігається у виді вектора, елементами якого є коефіцієнти полінома від a_n до a_0 :

$$P = [a_n \dots a_2 \ a_1 \ a_0].$$

Уведення полінома у Matlab здійснюється у такий самий спосіб, як і введення вектора довжиною $n+1$, де n – порядок полінома.

Множення поліномів. Добутком двох поліномів степенів n і m відповідно, як відомо, називають поліном степеня $n+m$, коефіцієнти якого визначають простим перемноженням цих двох поліномів. Фактично операція множення двох поліномів зводиться до побудови розширеного вектора коефіцієнтів по заданих векторах коефіцієнтів поліномів-співмножників. Цю операцію в математиці називають *згорткою векторів* (а самий вектор, одержуваний у результаті такої процедури – *вектором-згорткою двох векторів*). У Matlab її здійснює функція *conv*(P1, P2).

Аналогічно, функція *deconv*(P1, P2) здійснює *ділення* полінома P1 на поліном P2, тобто *обернену згортку векторів* P1 і P2. Вона визначає коефіцієнти полінома, що є часткою від ділення P1 на P2.

Приклад :

```
» p1 = [1,2,3]; p2 = [1,2,3,4,5,6];
» p = conv(p1,p2)
p = 1 4 10 16 22 28 27 18
» deconv(p,p1)
ans = 1 2 3 4 5 6
```

У загальному випадку ділення двох поліномів призводить до одержання двох поліномів – полінома-результату (частки) і полінома-остачі. Щоб одержати обидва ці поліноми, треба оформити звернення до функції у такий спосіб:

$$[Q,R] = \text{deconv}(B,A).$$

Тоді результат буде виданий у виді вектора Q з остачею у виді вектора R так, що буде виконане співвідношення

$$B = \text{conv}(A,Q) + R.$$

Система Matlab має функцію *roots*(P), яка *обчислює вектор, елементи якого є коренями заданого полінома P*.

Нехай потрібно знайти корені полінома:

$$P(x) = x^5 + 8x^4 + 31x^3 + 80x^2 + 94x + 20.$$

Нижче показано, як просто це зробити :

```
» p = [1,8,31,80,94,20];
» disp(roots(p))
-1.0000 + 3.0000i
-1.0000 - 3.0000i
-3.7321
-6.0000
-0.2679
```

Обернена операція – побудова вектора *p* коефіцієнтів полінома за заданим вектором його коренів – здійснюється функцією *poly*:

$$p = \text{poly}(r).$$

Тут *r* - заданий вектор значень коренів, *p* - обчислений вектор коефіцієнтів полінома. Наведемо приклад:

```
» p = [1,8,31,80,94,20]
```

```

p = 1 8 31 80 94 20
» r = roots(p)
r =
-1.0000 + 3.0000i
-1.0000 - 3.0000i
-3.7321
-6.0000
-0.2679

```

```

» p1 = poly(r)
p1 = 8.0000 31.0000 80.0000 94.0000 20.0000

```

Зауважимо, що одержуваний вектор не показує старшого коефіцієнта, який за замовчуванням покладається рівним одиниці.

Ця ж функція у випадку, коли аргументом її є деяка квадратна матриця A розміром $(n \times n)$, будує вектор характеристичного полінома цієї матриці. Звернення

```
p = poly(A)
```

формує вектор p коефіцієнтів полінома

$$p(s) = \det(s \cdot E - A) = p_1 \cdot s^n + \dots + p_n \cdot s + p_{n+1},$$

де E – позначення одиничної матриці розміром $(n \times n)$.

Розглянемо приклад:

```
» A = [1 2 3; 5 6 0; -1 2 3]
```

```
A =
1 2 3
5 6 0
-1 2 3
```

```
» p = poly(A)
```

```
p =
1.0000 -10.0000 20.0000 -36.0000
```

Для обчислення значення полінома за заданим значенням його аргументу в Matlab передбачена функція *polyval*. Звернення до неї здійснюється за схемою:

```
y = polyval(p,x),
```

де p – заданий вектор коефіцієнтів полінома, а x – задане значення аргументу.

Приклад:

```
» y = polyval(p,2)
y = 936
```

Якщо як аргумент поліному зазначена матриця X , то функція *polyval(p,X)* обчислює матрицю Y , кожний елемент якої є значенням зазначеного полінома при значенні аргументу, рівному відповідному елементу матриці X , наприклад:

```
p = 1 8 31 80 94 20
```

```
» X = [1 2 3; 0 -1 3; 2 2 -1]
```

```
X =
1 2 3
0 -1 3
2 2 -1
```

```
» disp(polyval(p,X))
```

```
234 936 2750
20 -18 2750
936 936 -18
```

У цьому випадку функція обчислює значення полінома для кожного елемента матриці X , і тому розміри вхідної й вихідної матриць однакові $\text{size}(Y) = \text{size}(X)$.

Обчислення похідної від полінома здійснюється функцією *polyder*. Ця функція створює вектор коефіцієнтів полінома, який є похідною від заданого полінома. Вона має три види звернень:

$dp = \text{polyder}(p)$ за заданим поліномом p обчислює вектор dp , елементи якого є коефіцієнтами полінома-похідної від заданого:

```
» dp = polyder(p)
dp = 5 32 93 160 94;
```

$dp = \text{polyder}(p1,p2)$ обчислює вектор dp , елементи якого є коефіцієнтами полінома-похідної від добутку двох поліномів $p1$ і $p2$:

```
» p1 = [1,8,31,80,94,20];
» p2 = [1,2,16];
» p = conv(p1,p2)
p =
Columns 1 through 6
    1    10    63   270   750  1488
Columns 7 through 8
  1544    320
```

```
» dp = polyder(p)
dp =
Columns 1 through 6
    7    60   315  1080  2250  2976
Column 7
  1544
```

```
» dp1 = polyder(p1,p2)
dp1 =
Columns 1 through 6
    7    60   315  1080  2250  2976
Column 7
  1544;
```

$[q,p] = \text{polyder}(p1,p2)$ обчислює похідну від відношення $(p1/p2)$ двох поліномів $p1$ і $p2$ і видає результат у виді відношення (q/p) поліномів q і p :

```
» p1 = [1,8,31,80,94,20];
» p2 = [1,2,16];
» [q,p] = polyder(p1,p2)
q =    3    24   159   636   1554   2520   1464
p =    1    4   36   64  256
» z = deconv(q,p)
z =    3   12    3
» y = deconv(p1,p2)
y =    1    6    3  -22
» z1 = polyder(y)
z1 =    3   12    3.
```

5.3.4. Процедура *plot*

Виведення графіків у системі Matlab є настільки простою і зручною процедурою, що нею можна користуватися навіть при обчисленнях у режимі калькулятора.

Основною функцією, що забезпечує побудову графіків на екрані дисплея, є функція *plot*. Загальна форма звернення до цієї процедури така:

```
plot(x1,y1,s1,x2,y2,s2,...).
```

Тут $x1$, $y1$ - задані вектори, елементами яких є масиви значень аргументу ($x1$) і функції ($y1$), що відповідають першій кривій графіка; $x2$, $y2$ - масиви значень аргументу й функції другої кривої і т.д. При цьому передбачається, що

значення аргументу відкладаються уздовж горизонтальної осі графіка, а значення функції - уздовж вертикальної осі. Змінні s_1, s_2, \dots є символічними (їхня вказівка не є обов'язковою). Кожна з них може містити до трьох спеціальних символів, що визначають відповідно: а) тип лінії, що з'єднує окремі точки графіка; б) тип точки графіка; в) колір лінії. Якщо змінні s не зазначені, то тип лінії за умовчанням - відрізок прямої, тип точки - піксел, а колір встановлюється (у версії 5) за такою черговістю: - синій, зелений, червоний, блакитний, фіолетовий, жовтий, чорний і білий - у залежності від того, яка по черзі лінія виводиться на графік. Наприклад, звернення виду **plot**($x_1, y_1, x_2, y_2, \dots$) призведе до побудови графіка, у якому перша крива буде лінією з відрізків прямих синього кольору, друга крива - такого ж типу зеленою лінією і т.д.

Графіки в Matlab завжди виводяться в окреме (графічне) вікно, яку називають *фігурою*.

Наведемо приклад. Нехай потрібно вивести графік функції

$$y = 3\sin(x + \pi/3)$$

на проміжку від -3π до $+3\pi$ із кроком $\pi/100$.

Спочатку треба сформуванати масив значень аргументу x :

$$x = -3*\pi : \pi/100 : 3*\pi,$$

потім обчислити масив відповідних значень функції:

$$y = 3*\sin(x+\pi/3)$$

і, нарешті, побудувати графік залежності $y(x)$.

У цілому в командному вікні ця послідовність операцій буде виглядати так:

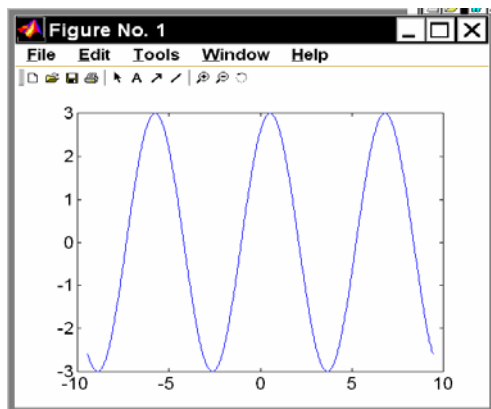
```
» x = -3*pi:pi/100:3*pi;
» y = 3*sin(x+pi/3);
» plot(x,y)
```

У результаті на екрані з'явиться додаткове вікно із графіком (див. рис. 5.19, а).

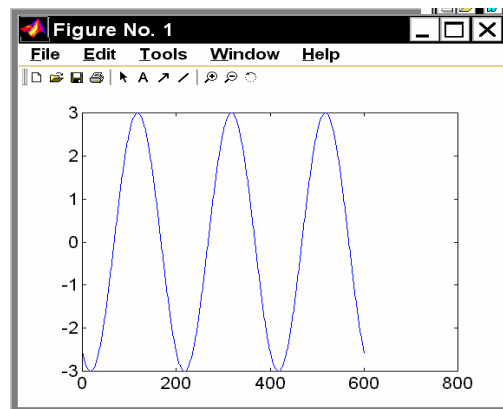
Якщо вектор аргументу при зверненні до функції **plot** не зазначено явно, то система обирає за умовчанням як аргумент номер елемента вектора функції. Наприклад, якщо ввести команду

```
» plot(y),
```

то результатом буде поява графіка у виді, наведеному на рис. 5.19, б.



а



б

Рис. 5.19. Виведення графіка синусоїди

Графіки, наведені на рис. 5.19, мають деякі недоліки:

- на них не нанесено сітку з координатних ліній, що утруднює "зчитування" графіків;
- немає загальної інформації про криві графіка (заголовка);
- невідомо, які величини відкладені по осях графіка.

Перший недолік усувається за допомогою функції **grid**. Якщо цю функцію записати одразу після звернення до функції **plot**:

```
» x = -3*pi:pi/100:3*pi;
» y = 3*sin(x+pi/3);
» plot(x,y), grid,
```

то графік буде споряджений координатною сіткою (рис. 5.20).

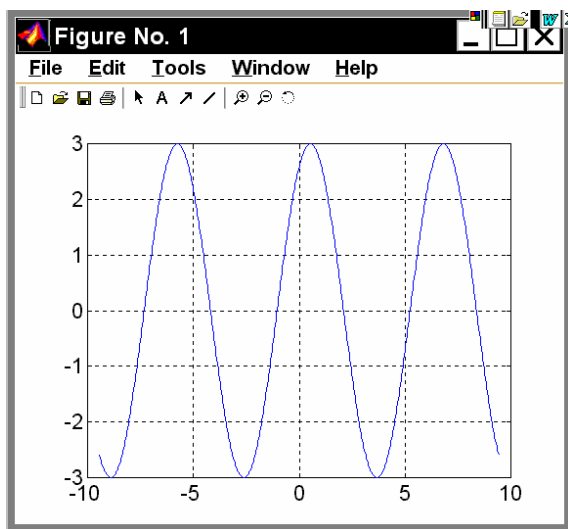


Рис. 5.20. Застосування **grid**

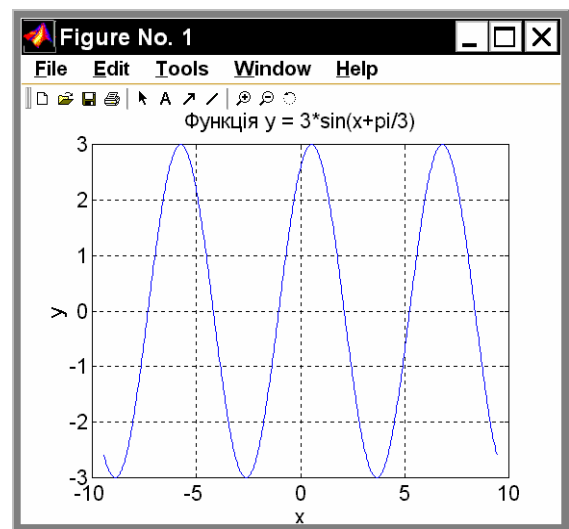


Рис. 5.21. Застосування **title**, **xlabel**, **ylabel**

Цінною особливістю графіків, побудованих у системі Matlab, є те, що сітка координат завжди відповідає "цілим" крокам змінювання, що робить графіки "читабельними", тобто за графіком можна робити "відлік" значення функції при будь-якому заданому значенні аргументу і навпаки. Такої властивості не має жодний із графічних пакетів-додатків до мов програмування високого рівня.

Заголовок графіка виводиться за допомогою процедури **title**. Якщо після звернення до процедури **plot** викликати **title** у такий спосіб:

```
title('<текст>'),
```

то понад графіком виникне текст, записаний між апострофами у дужках. При цьому варто пам'ятати, що текст завжди має міститися в апострофах.

Аналогічно можна вивести пояснення до графіка, що розміщуються уздовж горизонтальної осі (функція **xlabel**) і уздовж вертикальної осі (функція **ylabel**).

Наприклад, сукупність операторів

```
» x = -3*pi : pi/100 : 3*pi;
» y = 3*sin(x+pi/3);
» plot(x,y), grid
» title('Функція y = 3*sin(x+pi/3)');
```

```
» xlabel('x'); ylabel('y');
```

приведе до оформлення поля фігури у виді, поданому на рис. 5.21. Очевидно, така форма вже цілком задовольняє вимоги, що висуваються до інженерних графіків.

Не більш складним є виведення у середовищі Matlab графіків функцій, заданих *параметрично*. Нехай, наприклад, необхідно побудувати графік функції $y(x)$, що задана параметричними формулами:

$$x = 4e^{-0,05t} \sin t; \quad y = 0,2e^{-0,05t} \sin 2t.$$

Виберемо діапазон змінювання параметра t від 0 до 50 із кроком 0.1. Тоді, набираючи сукупність операторів

```
» t = 0:0.1:50;
» x = 4*exp(-0.05*t).*sin(t);
» y = 0.2*exp(-0.1*t).*sin(2*t);
» plot(x,y)
» title('Параметрична функція x=4*exp(-0.05t)*sin(t); y=0.2*exp(-0.1t)*sin(2t)')
» grid,
```

одержимо графік рис. 5.22.

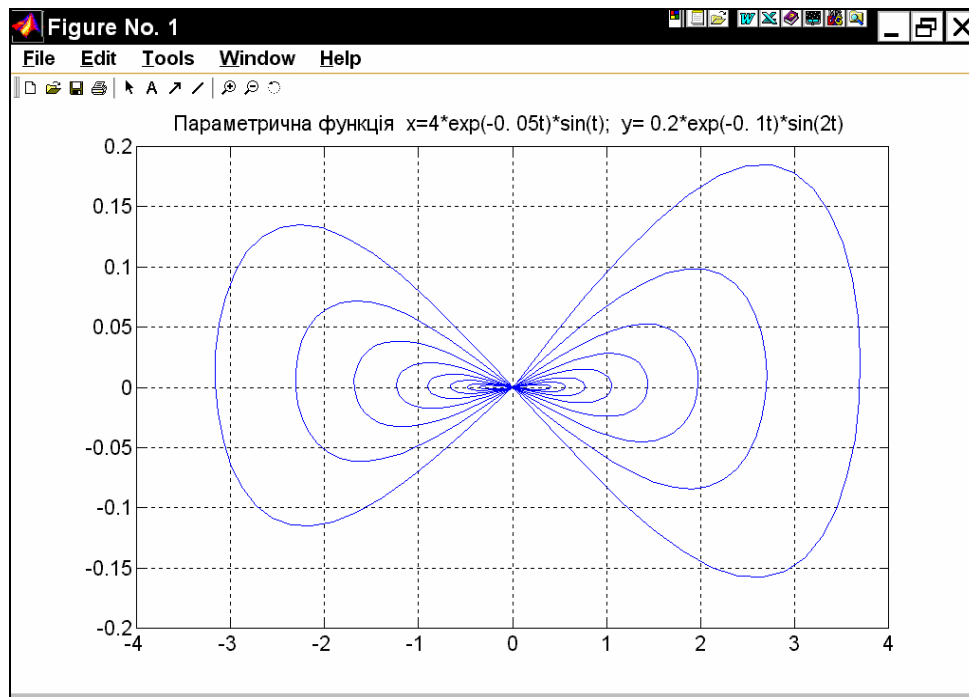


Рис. 5. 22. Графік параметрично заданої функції

5.3.5. Спеціальні графіки

Великою зручністю, надаваною системою Matlab, є зазначена раніше можливість не вказувати аргумент функції при побудові її графіка. У цьому випадку як аргумент система приймає номер елемента вектора, графік якого буде створено. Користуючись цим, наприклад, можна побудувати "графік вектора":

```
» x = [ 1 3 2 9 6 8 4 6];
» plot (x)
» grid
» title('Графік вектора X')
```

- » `ylabel('Значення елементів')`
- » `xlabel('Номер елемента')`.

Результат поданий на рис. 5.23.

Ще більш наочним є подання вектора у виді *стовпцевої діаграми* за допомогою функції *bar* (див. рис. 5.24):

- » `bar(x)`
- » `title('Графік вектора X')`
- » `xlabel('Номер елемента')`
- » `ylabel('Значення елементів')`

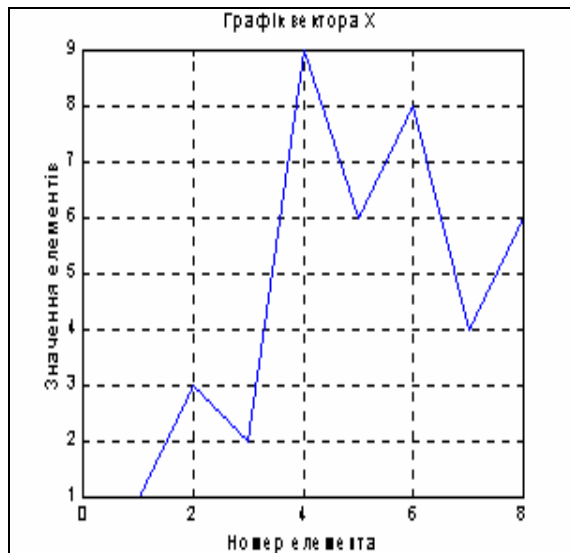


Рис. 5.23. Застосування *plot*

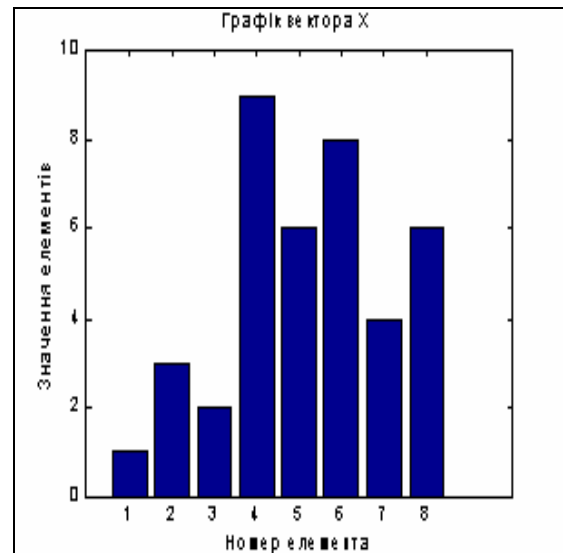


Рис. 5.24. Застосування *bar*

Якщо функція задана своїми значеннями при дискретних значеннях аргументу, і невідомо, як вона може змінюватися в проміжках між значеннями аргументу, зручніше подавати графік такої функції у виді окремих вертикальних ліній для кожного із заданих значень аргументу. Це можна зробити, застосовуючи процедуру *stem*, звернення до якої цілком аналогічно зверненню до процедури *plot*:

```
x = [ 1 3 2 9 6 8 4 6];
stem(x,'k')
grid
set(gca,'FontName','Arial','FontSize',14),
title('Графік вектори X')
ylabel('Значення елементів')
xlabel('Номер елемента')
```

На рис. 5.25 зображено одержаний при цьому графік.

Інший приклад - побудова графіка функції $y = e^{-x^2}$ у виді стовпцевої діаграми (рис. 5.26):

```
» x = - 6.9 : 0.2 : 6.9;
» bar(x, exp(-x . * x))
» title('Стовпцева діаграма функції y = exp(-x^2)')
» xlabel (' Аргумент x')
» ylabel (' Значення функції y')
```

Ще одна корисна інженеру функція – *hist* (побудова графіка *гістограми* заданого вектора). Стандартне звернення до неї має вид:

```
hist(y,x),
```

де y – вектор, гістограму якого потрібно побудувати; x – вектор, що визначає інтервали змінювання першого вектора, усередині яких підраховується кількість елементів вектора "y".

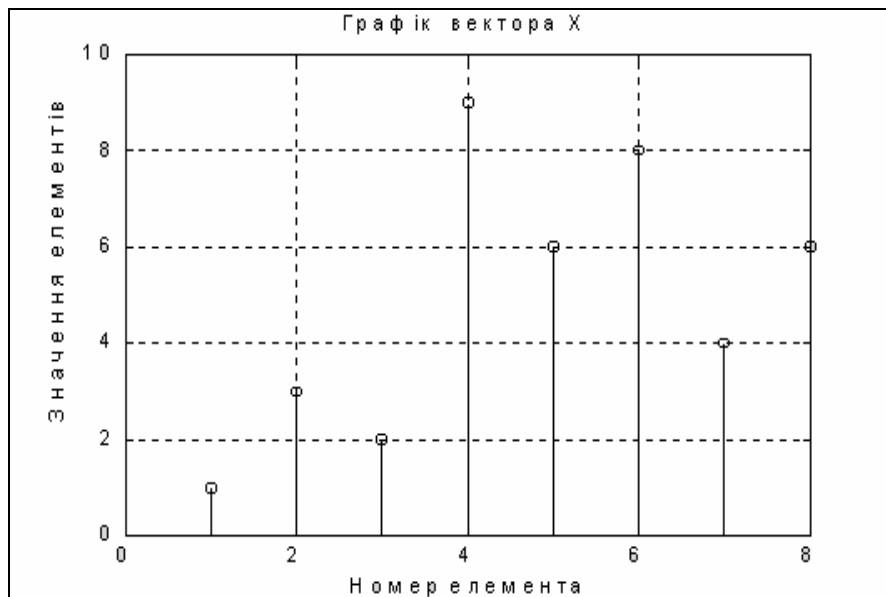


Рис. 5.25. Застосування *stem*

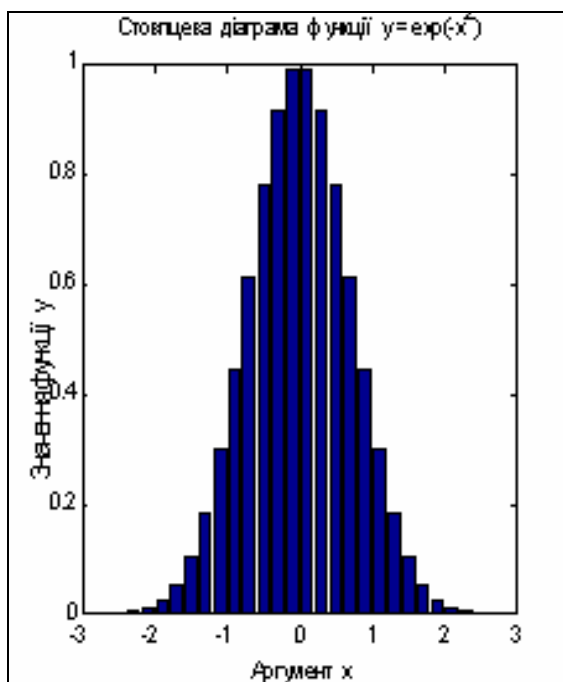


Рис. 5.26. Процедура *bar*

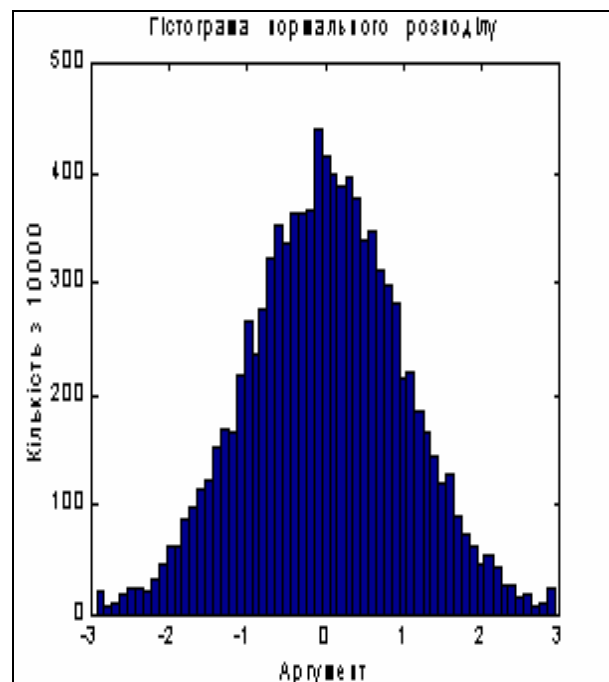


Рис. 5.27. Процедура *hist*

Ця функція виконує дві операції

- підраховує кількість елементів вектора "y", значення яких потрапляють усередину відповідного діапазону, зазначеного вектором "x";
- будує стовпцеву діаграму підрахованих чисел елементів вектора "y" як функцію діапазонів, зазначених вектором "x".

Як приклад роздивимося побудову гістограми випадкових величин, що формуються вмонтованою функцією *randn*. Візьмемо загальну кількість елементів вектора цих випадкових величин 10 000. Побудуємо гістограму для діапазону змінювання цих величин від -2,9 до +2,9. Інтервали змінювання нехай будуть рівні 0,1. Тоді графік гістограми можна побудувати за допомогою сукупності таких операторів:

```
» x = -6.9:0.1:6.9;
» y = randn(10000,1);
» hist(y,x)
» ylabel('Кількість з 10000')
» xlabel('Аргумент')
» title('Гістограма нормального розподілу')
```

Результат поданий на рис. 5.27. З нього, зокрема, випливає, що вмонтована функція *randn* достатньо вірно відображає нормальний гауссовий закон розподілу випадкової величини.

Процедура *comet(x,y)* ("комета") будує графік залежності $y(x)$ поступово у часі у виді траєкторії комети. При цьому зображуюча точка на графіку має вид маленької комети (із голівкою й хвостиком), що плавно переміщується від однієї точки до іншої. Наприклад, якщо ввести сукупність операторів:

```
» t = 0:0.1:50;
» x = 4 * exp(-0.05*t) .* sin(t);
» y = 0.2 * exp(-0.1*t) .* sin(2*t);
» comet(x,y),
```

то графік, наведений на рис. 5.22, буде побудований як траєкторія послідовного руху комети. Ця обставина може бути корисною при побудові просторових траєкторій для виявлення характеру змінювання траєкторії з часом.

Matlab має декілька функцій, що дозволяють будувати графіки в логарифмічному масштабі.

Функція *logspace* із зверненням

```
x = logspace(d1, d2, n)
```

формує вектор-рядок "x", що містить "n" рівновіддалених у логарифмічному масштабі одна від одної точок, що покривають діапазон від 10^{d1} до 10^{d2} .

Функція *loglog* цілком аналогічна функції *plot*, але графіки по обох осях будуються в логарифмічному масштабі.

Для побудови графіків, які використовують логарифмічний масштаб тільки по одній з координатних осей, користуються процедурами *semilogx* і *semilogy*. Перша процедура будує графіки з логарифмічним масштабом уздовж горизонтальної осі, друга - уздовж вертикальної осі.

Звернення до останніх трьох процедур цілком аналогічно зверненню до функції *plot*.

Як приклад розглянемо побудову графіків амплітудно-частотної й фазочастотної характеристик ланки, що описується передатною функцією:

$$W(p) = \frac{p + 4}{p^2 + 4 \cdot p + 100}.$$

Для цього треба, по-перше, створити поліном чисельника $P_c = [1 \ 4]$ і знаменника передатної функції $P_z = [1 \ 4 \ 100]$. По-друге, визначити корені цих двох поліномів:

```

» P1 = [1 4]; P2 = [1 4 100];
» roots(P1)
ans = -4
» roots(P2)
ans =
-6.0000e+000 +9.7980e+000i
-6.0000e+000 -9.7980e+000i

```

По-третє, задати діапазон змінювання частоти так, щоб він охоплював усі знайдені корені:

```
om0 = 1e-2; omk = 1e6.
```

Тепер потрібно задатися кількістю точок майбутнього графіка (наприклад, $n = 41$), і сформувати масив точок по частоті

```
OM = logspace(-2,2,41),
```

де значення -2 і $+2$ відповідають десятковим порядкам початкового $om0$ і кінцевого omk значень частоти.

Користуючись функцією *polyval*, можна обчислити спочатку вектор "ch" комплексних значень чисельника частотної передатної функції, що відповідають заданій передатній функції за Лапласом, якщо як аргумент функції *polyval* використати сформований вектор частот OM, елементи якого помножені на уявну одиницю (див. визначення Частотної Передатної Функції). Аналогічно обчислюється комплекснозначний вектор "zn" знаменника ЧПФ.

Вектор значень АЧХ (амплітудно-частотної характеристики) можна знайти, вираховуючи модулі векторів чисельника і знаменника ЧПФ і поелементно ділячи отримані вектори. Щоб знайти вектор значень ФЧХ (фазочастотної характеристики) треба розділити поелементно комплекснозначні вектори чисельника й знаменника ЧПФ і визначити вектор аргументів елементів отриманого вектора. Для того щоб фази подати в градусах, отримані результати варто помножити на 180 і розділити на π .

Нарешті, для побудови графіка АЧХ у логарифмічному масштабі, достатньо застосувати функцію *loglog*, а для побудови ФЧХ зручніше скористатися функцією *semilogx*.

У цілому послідовність дій може бути такою:

```

» OM = logspace(-2,2,40)
» ch = polyval(P1,i*OM);
» zn = polyval(P2,i*OM);
» ACH = abs(ch) ./abs(zn);
» loglog(OM,ACH); grid;
>> title('Графік Амплітудно-Частотної Характеристики')
>> xlabel('Частота (рад/с)');
>> ylabel('Відношення амплітуд')
» FCH = angle(ch ./zn)*180/pi;
» semilogx(OM,FCH); grid;
>> title('Фазо-Частотна Характеристика'),
>> xlabel('Частота (рад/с)'),
>> ylabel('Фаза (градуси)')

```

В результаті утворюються графіки, зображені на рис. 5.28 і 5.29.

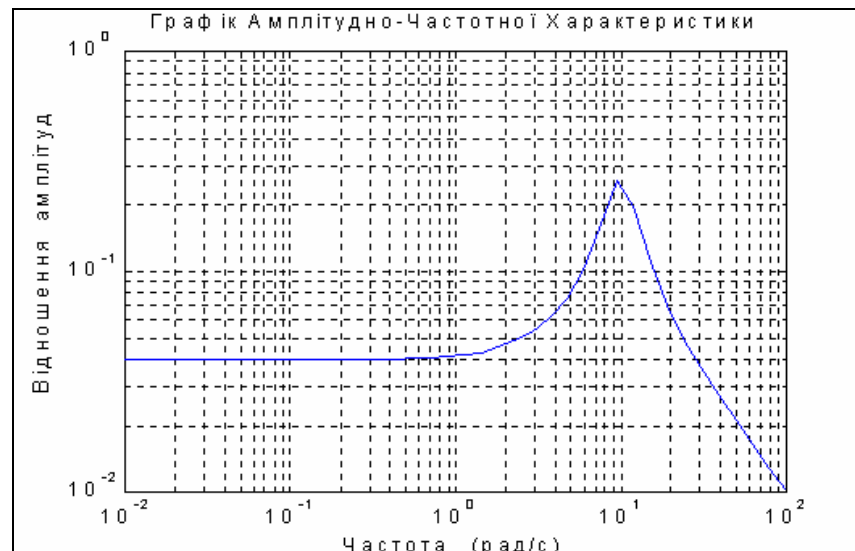


Рис. 5.28. Графік АЧХ

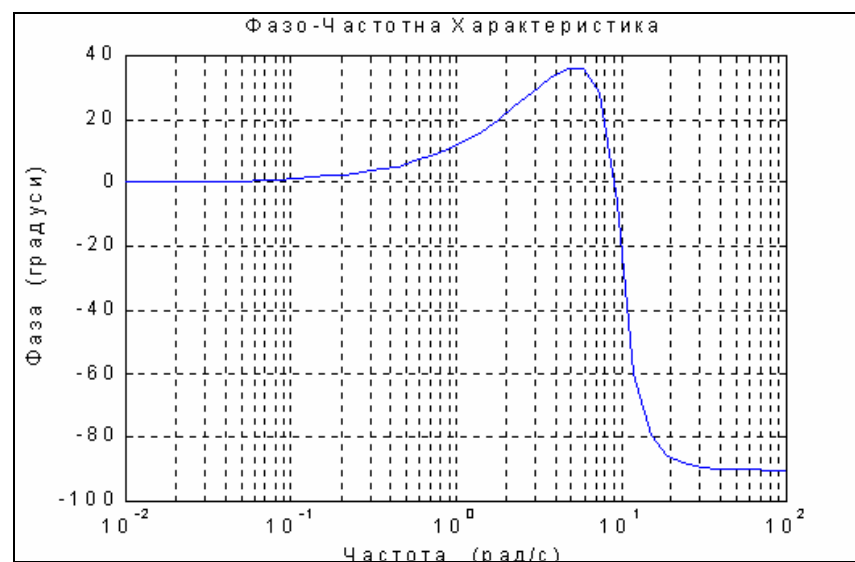


Рис. 5.29. Графік ФЧХ

5.3.6. Додаткові функції графічного вікна

Зазвичай графіки, одержувані за допомогою процедур *plot*, *loglog*, *semilogx* і *semilogy*, автоматично будуються в таких масштабах по осях, щоб у поле графіка помістилися всі обчислені точки графіка, включаючи максимальні і мінімальні значення аргументу й функції. Проте Matlab має можливості встановлення й інших режимів масштабування. Це досягається за рахунок використання процедури *axis*.

Команда

axis([xmin xmax ymin ymax])

установлює жорсткі межі поля графіка в одиницях величин, що відкладаються по осях.

Команда *axis*('auto') повертає масштаби по осях до їх штатних значень (прийнятих за замовчуванням).

Команда `axis('ij')` переміщує початок відліку в лівий верхній ріг і реалізує відлік від верхнього лівого рогу (матрична система координат).

Команда `axis('xu')` повертає декартову систему координат із початком відліку в лівому нижньому рогу.

Команда `axis('square')` установлює однаковий діапазон змінювання змінних по осях графіка.

Команда `axis('equal')` забезпечує однаковий масштаб по обох осях графіка.

В одному графічному вікні, але на окремих графічних полях можна побудувати декілька графіків, використовуючи процедуру `subplot`. Звернення до цієї процедури повинно передувати зверненню до процедур `plot`, `loglog`, `semilogx` і `semilogy` і мати такий вид:

`subplot(m,n,p).`

Тут m – указує, на скільки частин розділяється графічне вікно по вертикалі, n – по горизонталі, а p – номер підвікна, у якому буде будуватися графік. При цьому підвікна нумеруються зліва праворуч порядково зверху вниз (так, як по рядках читається текст книги).

Наприклад, два попередні графіки можна помістити в одне графічне вікно в такий спосіб:

```
subplot(2,1,1);
loglog(OM,ACH,'k'); grid;
set(gca,'FontName','Arial','FontSize',14),
title('Амплітудно-Частотна Характеристика')
ylabel('Амплітуда')
subplot(2,1,2);
semilogx(OM,FCH,'k'); grid
set(gca,'FontName','Arial','FontSize',14),
title('Фазо-Частотна Характеристика')
xlabel('Частота (рад/с)'), ylabel('Фаза (гр.)')
```

Результат поданий на рис. 5.30.

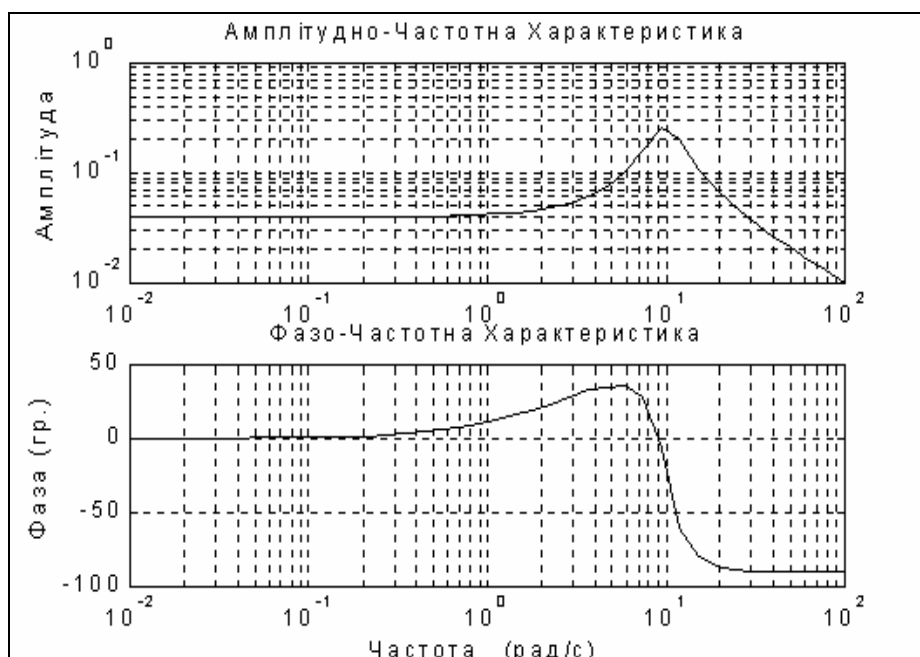


Рис. 5.30. Використання функції `subplot`

Команда *text*(*x*, *y*, '<текст>') дозволяє розмістити зазначений текст на полі графіка, при цьому початок тексту поміщається в точку з координатами *x* і *y*. Значення зазначених координат повинні бути подані в одиницях величин, що відкладаються по осях графіка, і знаходитися усередині діапазону змінювання цих величин. Часто це незручно, бо потребує попереднього знання цього діапазону, що рідко коли є можливим.

Більш зручним для розміщення тексту усередині поля графіка є використання команди *gtext*('<текст>'), яка висвічує в активному графічному вікні перехрестя, переміщення якого за допомогою "мишки" дозволяє вказати місце початку виведення зазначеного тексту. Після цього натисканням лівої клавіші "мишки" або будь-якої клавіші текст вводиться в зазначене місце.

Щоб створити декілька графічних вікон, у кожному з яких розташовані відповідні графіки, можна скористатися командою *figure*, яка створить таке графічне вікно, залишаючи попередні.

Нарешті, для того, щоб декілька послідовно обчислюваних графіків були зображені в одному графічному вікні в одному стилі, можна використати команду *hold on*, тоді кожний такий графік буде будуватися в тому ж попередньо відкритому графічному вікні, тобто кожна нова лінія буде додаватися до раніше побудованих. Команда *hold off* виключає режим зберігання графічного вікна, встановленого попередньою командою.

5.3.7. Вставлення графіків до документу

Щоб вставити графік із графічного вікна (фігури) до документу Word, слід скористатися командами меню, розташованого у верхній частині вікна фігури. У меню *Edit* оберіть команду *Copy Figure*. Перейдіть у вікно документу і натисніть клавіші <Ctrl>+<C>. Вміст графічного вікна виникне у тому місці документу, де містився курсор.

5.3.8. Апроксимація та інтерполяція даних

Поліноміальна апроксимація даних вимірів, які сформовані у вигляді деякого вектора *Y*, при деяких значеннях аргументу, які утворюють вектор *X* такої самої довжини, що й вектор *Y*, здійснюється процедурою *polyfit*(*X*, *Y*, *n*). Тут *n* – порядок апроксимуючого полінома. Результатом дії цієї процедури є вектор довжиною (*n* + 1) із коефіцієнтів апроксимуючого полінома.

Нехай масив значень аргументу є таким:

$$x = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8],$$

а масив відповідних значень вимірної величини - таким:

$$y = [-1. \ 1 \ 0.2 \ 0.5 \ 0.8 \ 0.7 \ 0.6 \ 0.4 \ 0.1].$$

Тоді, застосовуючи зазначену функцію при різних значеннях порядку апроксимуючого полінома, одержимо:

```
» x = [1 2 3 4 5 6 7 8];  
» y = [-1.1 0.2 0.5 0.8 0.7 0.6 0.4 0.1];  
» polyfit(x,y,1)
```

```
ans = 0.1143 -0.2393
» polyfit(x,y,2)
ans = -0.1024 1.0357 -1.7750
» polyfit(x,y,3)
ans = 0.0177 -0.3410 1.9461 -6.6500
» polyfit(x,y,4)
ans = -0.0044 0.0961 -0.8146 3.0326 -3.3893.
```

Це означає, що задану залежність можна апроксимувати або прямою

$$y(x) = 0,1143x - 0,2393,$$

або квадратною параболою

$$y(x) = -0,1024x^2 + 1,0357x - 1,775,$$

або кубічною параболою

$$y(x) = 0,0177x^3 - 0,341x^2 + 1,9461x - 2,65,$$

або параболою четвертого степеня

$$y(x) = -0,0044x^4 + 0,0961x^3 - 0,8146x^2 + 3,0326x - 3,3893.$$

Побудуємо в одному графічному полі графіки заданої дискретної функції й графіки всіх отриманих при апроксимації поліномів:

```
x = [1 2 3 4 5 6 7 8];
y = [-1.1 0.2 0.5 0.8 0.7 0.6 0.4 0.1];
P1=polyfit(x,y,1);
P2=polyfit(x,y,2);
P3=polyfit(x,y,3);
P4=polyfit(x,y,4);
stem(x,y);
x1 = 0.5 : 0.2 : 8.5;
y1=polyval(P1,x1);
y2=polyval(P2,x1);
y3=polyval(P3,x1);
y4=polyval(P4,x1);
hold on
plot(x1,y1,'-.',x1,y2,'x-',x1,y3,'o-',x1,y4,'^-',
grid, set(gca, 'FontName', 'Arial Cyr', 'FontSize', 16),
title('Поліноміальна апроксимація ');
xlabel('Аргумент');
ylabel('Функція')
legend('Задані значення','1','2','3','4',0)
```

Результат поданий на рис. 5.31.

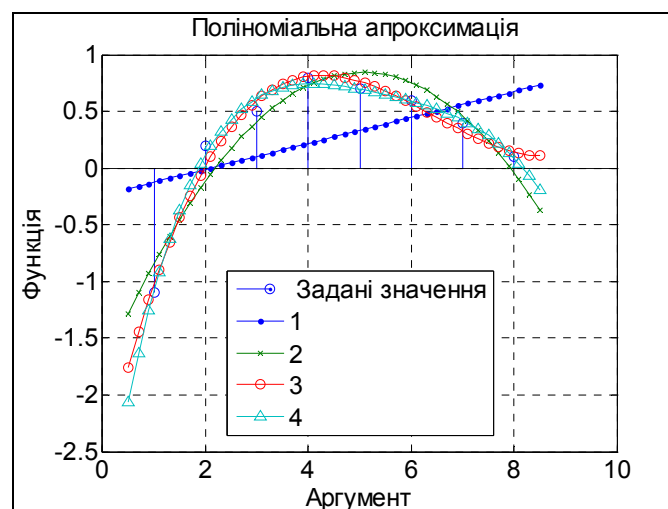


Рис. 5.31. Поліноміальна апроксимація функцією *polyfit*

Функція `spline(X,Y,Xi)` здійснює *інтерполяцію кубічними сплайнами*.
При зверненні

$$Y_i = \text{spline}(X, Y, X_i)$$

вона інтерполює значення вектора Y , заданого при значеннях аргументу, поданих у векторі X , і видає значення інтерполюючої функції у виді вектора Y_i при значеннях аргументу, заданих вектором X_i . У випадку, коли вектор X не зазначений, за замовчуванням приймається, що він має довжину вектора Y і кожний його елемент дорівнює номеру цього елемента.

Як приклад розглянемо інтерполяцію вектора

```
x = -0.5:0.1:0.2; y = [-1.1 0.2 0.5 0.8 0.7 0.6 0.4 0.1];
x1 = -0.5:0.02:0.2; y2 = spline(x,y,x1);
plot(x,y,x1,y2,'-'), grid
set(gca,'FontName','Arial Cyr','FontSize',16),
title('Інтерполяція процедурою SPLINE ');
xlabel('Аргумент'); ylabel('Функція'); legend('лінійна','сплайнова',0)
```

Результат наведений на рис. 5.32.

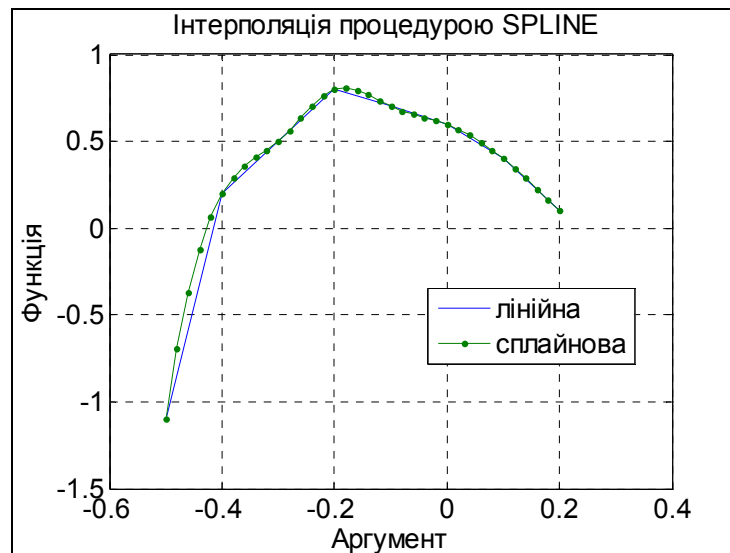


Рис. 5.32. Інтерполяція функцією `spline`

Одновимірну табличну інтерполяцію робить процедура `interp1`. Звернення до неї у загальному випадку має вид:

$$Y_i = \text{interp1}(X, Y, X_i, \text{'<метод>'}),$$

і дозволяє додатково зазначити метод інтерполяції у четвертому вхідному аргументі:

'nearest' - ступінчаста інтерполяція;

'linear' - лінійна;

'cubic' - кубічна;

'spline' - кубічними сплайнами.

Якщо метод не зазначений, здійснюється за умовчанням лінійна інтерполяція. Наприклад, (для того самого вектора):

```
x = -0.5:0.1:0.2; y = [-1.1 0.2 0.5 0.8 0.7 0.6 0.4 0.1];
```

```

x1 = -0.5:0.02:0.2;    y1 = interp1(x,y,x1);
y4 = interp1(x,y,x1,'nearest'); y2 = interp1(x,y,x1,'cubic');
y3 = interp1(x,y,x1,'spline');
plot (x1,y1,x1,y2,'.',x1,y3,x1,y4), grid
legend('лінійна','кубічна','сплайнова','ступінчаста')
set(gca,'FontName','Arial Cyr','FontSize',16),
title('Інтерполяція процедурою INTERP1 ');
xlabel('Аргумент');    ylabel('Функція')

```

Результат наведений на рис.5.33.

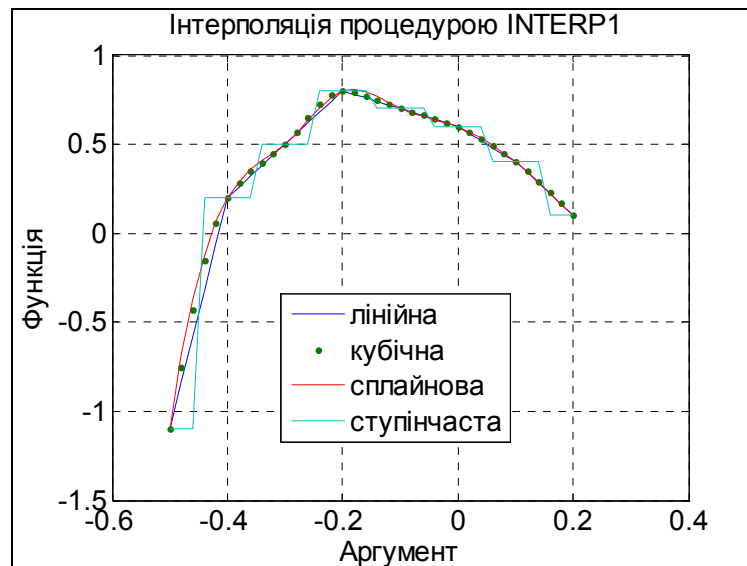


Рис. 5.33. Інтерполяція функцією `interp1`

5.3.9. Векторна фільтрація й спектральний аналіз

У системі Matlab є декілька функцій для проведення цифрового аналізу даних спостережень (вимірів).

Так, функція $y = \text{filter}(b,a,x)$ забезпечує формування вектора y по заданих векторах b , a , x відповідно до співвідношення:

$$y(k) = b(1)*x(k) + b(2)*x(k-1) + \dots + b(nb+1)*x(k-nb) - a(2)*y(k-1) - a(3)*y(k-3) - \dots - a(na+1)*y(k-na), \quad (5.1)$$

де вектор b має такий склад

$$b = [b(1), b(2), \dots, b(nb+1)],$$

а вектор a

$$a = [1, a(2), a(3), \dots, a(na+1)].$$

Співвідношення (5.1) можна розглядати як кінцево-різницеве рівняння фільтра з дискретною передатною функцією виду раціонального дроби, коефіцієнти чисельника якого утворюють вектор b , а знаменника – вектор a , на вхід якого подається сигнал $x(t)$, а на виході формується сигнал $y(t)$.

Тоді вектор y буде являти собою значення вихідного сигналу цього фільтра в дискретні моменти часу, що відповідають заданим значенням вхідного сигналу $x(t)$ (вектор x).

Нижче наведений приклад застосування функції `filter`.

» $x = 0:0.1:1;$


```

» b = [1 2];
» a = [ 1 0.1 4];
» y = filter(b,a,x)
y =
Columns 1 through 7
    0    0.1000    0.3900    0.2610   -0.5861    0.3146    3.9129
Columns 8 through 11
    0.2503  -13.4768    6.8466   56.4225

```

Функції *fft* (*Fast Fourier Transformation*) і *ifft* (*Invers Fast Fourier Transformation*) здійснюють перетворення заданого вектора, що відповідають дискретному прямому й оберненому перетворенням Фур'є.

Звернення до цих функцій виду:

$$y = \text{fft}(x, n); \quad x = \text{ifft}(y, n)$$

призводить до формування вектора y у першому випадку і x – у другому по формулах:

$$y(k) = \sum_{m=1}^n x(m) \cdot e^{-j \cdot 2\pi \cdot (m-1) \cdot (k-1) / n}; \quad (5.2)$$

$$x(m) = \frac{1}{n} \sum_{k=1}^n y(k) \cdot e^{j \cdot 2\pi \cdot (m-1) \cdot (k-1) / n}, \quad (5.3)$$

де j – позначення уявної одиниці; n - число елементів заданого вектора x (воно є також розміром вихідного вектора y).

Наведемо приклад. Сформуємо вхідний сигнал у виді вектора, елементи якого дорівнюють значенням функції, що є сумою двох синусоїд із частотами 5 і 12 Гц. Знайдемо Фур'є-зображення цього сигналу і виведемо графічні подання вхідного процесу й модуля його Фур'є-зображення:

```

t=0:0.001:2;
x = sin(2*pi*5*t) + cos(2*pi*12*t);
plot(t, x); grid
set(gca,'FontName','Arial Cyr','FontSize',16),
title('Вхідний процес ');
xlabel('Час (с)');
ylabel('X(t)')
y = fft(x);
a =abs(y);
plot(a); grid
set(gca,'FontName','Arial Cyr','FontSize',16),
title('Модуль Фур'є - зображення ');
xlabel('Номер елемента вектора');
ylabel('abs(F(X(t)))')

```

Результати відображені відповідно на рис. 5.34 і 5.35.

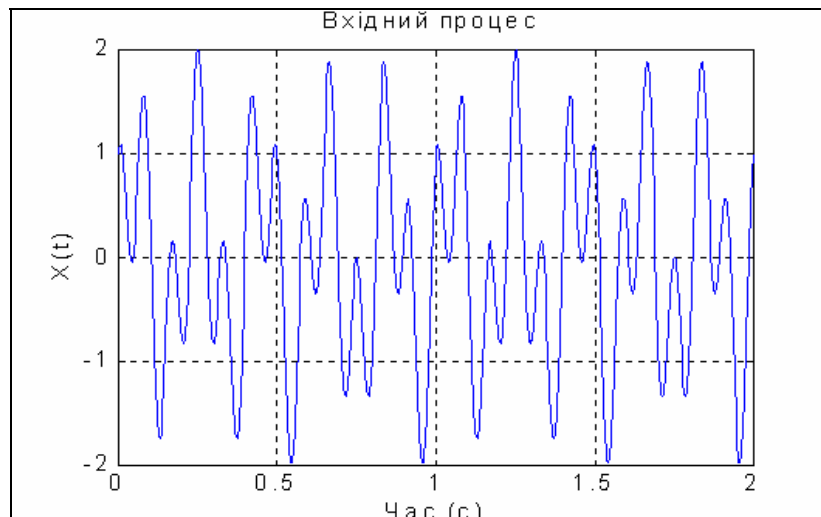
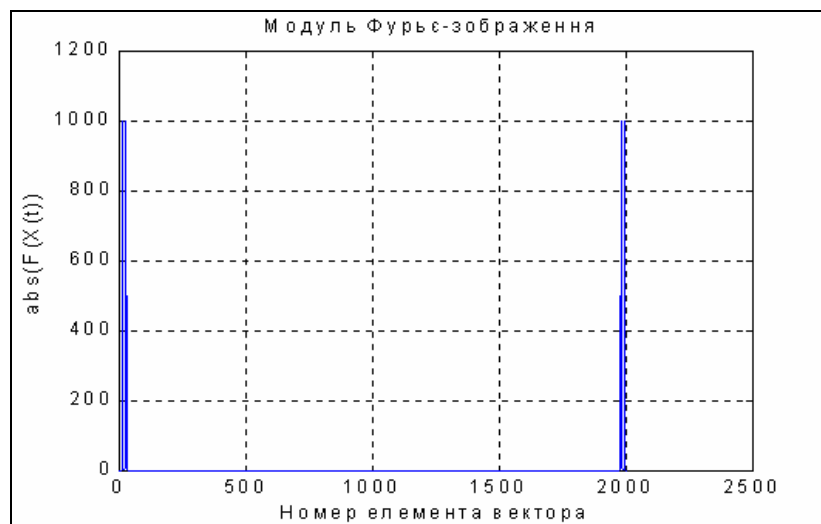


Рис. 5.34. Початковий процес

Рис. 5.35 Результат застосування процедури *fft*

Тепер здійснимо зворотне перетворення за допомогою функції *ifft* і результат також виведемо у формі графіка:

```
z = ifft(y);
plot(t, z); grid
set(gca, 'FontName', 'Arial Cyr', 'FontSize', 16),
title('Зворотне Фур'є-перетворення ');
xlabel('Час (с)');
ylabel('Z(t)')
```

На рис. 5.36 зображено результат. Розглядаючи його, можна переконатися, що відтворений процес збігається з початковим.

Рис. 5.36. Результат застосування процедури *ifft*

Уважно вивчаючи формулу дискретного перетворення Фур'є, можна дійти висновків:

а) номер m відповідає моменту часу t_m , у який виміряний вхідний сигнал $x(m)$; при цьому $t_1 = 0$;

б) номер k – це індекс значення частоти f_k , якому відповідає знайдений елемент $y(k)$ дискретного перетворення Фур'є;

в) щоб перейти від індексів до часової й частотної областей, треба знати значення h дискрету (кроку) часу, через який виміряний вхідний сигнал $x(t)$ і проміжок T часу, протягом якого він вимірюється; тоді крок (дискрет) по частоті в зображенні Фур'є визначиться співвідношенням:

$$\Delta f = 1/T, \quad (5.4)$$

а діапазон змінювання частоти - формулою

$$F = 1/h; \quad (5.5)$$

так, в аналізованому прикладі ($h = 0.001$, $T = 2$, $n = 21$)

$$\Delta f = 0.5; \quad F = 1000;$$

г) із (5.2) випливає, що індексу $k=1$ відповідає нульове значення частоти ($f_0 = 0$); інакше кажучи, перший елемент вектора $y(1)$ є значенням Фур'є-зображення при нульовій частоті, тобто є просто сумою всіх заданих значень вектора x ; звідси одержуємо, що вектор $y(k)$ містить значення Фур'є-зображення, починаючи з частоти $f_0 = 0$ (якій відповідає $k = 1$) до максимальної частоти $f_{max} = F$ (якій відповідає $k = n$); таким чином, Фур'є-зображення визначається функцією *fft* тільки для додатних частот у діапазоні від 0 до F ; це незручно для побудови графіків Фур'є-зображення від частоти; більш зручним і звичним є перехід до вектора Фур'є-зображення, якого визначено в діапазоні частот від $(-F/2)$ до $F/2$; частота $F_N = F/2$ одержала назву частоти Найквіста;

д) як відомо, функція e^{jz} є періодичною за z із періодом 2π ; тому інформація про Фур'є-зображення при від'ємних частотах розташована в другій половині вектора $y(k)$.

Сформуємо для аналізованого прикладу масив частот, виходячи з вищезазначеного:

```
f = 0:0.5:1000;
```

і виведемо графік з аргументом-частотою (рис. 5.37):

```
plot(f,a); grid
set(gca,'FontName','Arial Cyr','FontSize',16),
title('Модуль Фур'є - зображення');
xlabel('Частота (Гц)');
ylabel('abs(F(X(t)))')
```

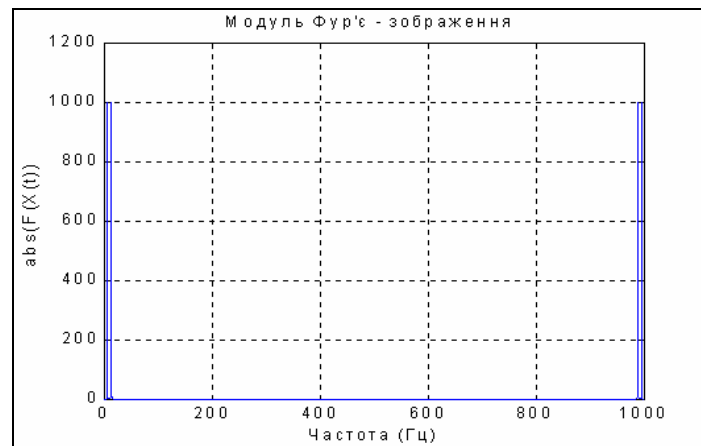


Рис. 5.37. Результат застосування функції *fft* у частотній області

Як впливає з розгляду рис. 5.37, за ним важко розпізнати ті частоти (5 і 12 Гц), із якими змінюється вхідний сигнал. Це є наслідком тієї обставини, яку було відзначено в примітці г). Щоб визначити справжній спектр вхідного сигналу, потрібно спочатку дещо перетворити отриманий вектор у Фур'є-зображення за допомогою процедури *fftshift*.

Функція *fftshift* (звернення до неї здійснюється у такий спосіб: $z = \text{fftshift}(y)$) призначена для формування нового вектора z із заданого вектора y шляхом переставлення другої половини вектора y у першу половину вектора z . При цьому друга половина вектора z складається з елементів першої половини вектора y . Більш точно цю операцію можна задати співвідношеннями:

$$z(1) = y(n/2+1); \dots, z(k) = y(n/2+k); \dots, z(n/2) = y(n); z(n/2+1) = y(1); \dots$$

$$\dots, z(n/2+k) = y(k); \dots z(n) = y(n/2).$$

Примітка. Операцію *fftshift* зручно використовувати для визначення масиву Фур'є-зображення з метою побудови його графіка в частотній області. Проте цей масив не може бути використаний для зворотного перетворення Фур'є.

Проілюструємо застосування цієї функції до попереднього прикладу:

```
f1 = -500 : 0.5 : 500; % Перебудова вектора частот
v = fftshift(y); % Перебудова вектора Фур'є-зображення
a = abs(v); % Відшукання модуля
% Відбудова графіка
plot(f1(970:1030),a(970:1030)); grid
set(gca,'FontName','Arial Cyr','FontSize',16),
title('Модуль Фур'є - зображення');
xlabel('Частота (Гц)'); ylabel('abs(F(X(t)))')
```

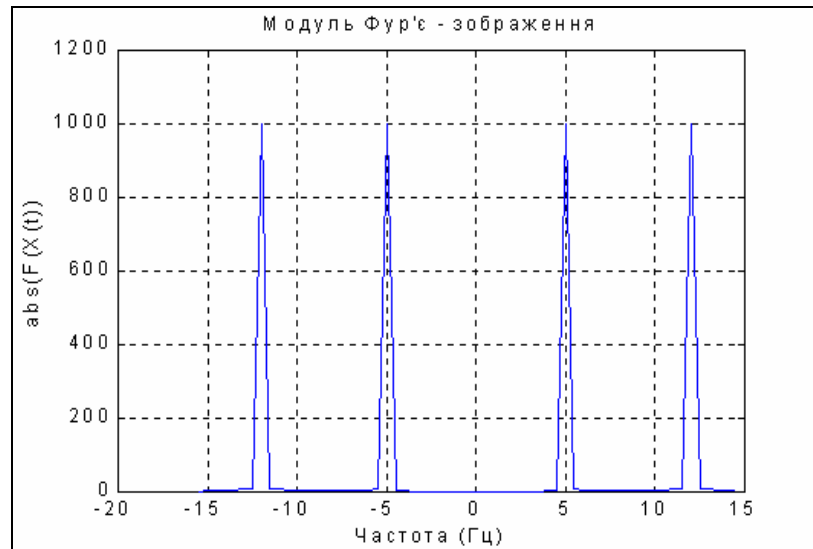


Рис. 5.38. Результат застосування процедури *fftshift*

З графіка рис. 5.38 вже стає очевидним, що в спектрі вхідного сигналу є дві гармоніки – із частотами 5 і 12 Гц.

Залишається лише та незручність, що з графіка спектра неможливо встановити амплітуди цих гармонік. Щоб уникнути цього, потрібно весь вектор y Фур'є-зображення поділити на число його елементів (n), щоб одержати вектор комплексного спектра сигналу:

```
N=length(y); a=abs(v)/N;
plot(f1(970:1030),a(970:1030)); grid
set(gca,'FontName','Arial Cyr','FontSize',16,'Color','white'),
title('Модуль комплексного спектра');
xlabel('Частота (Гц)'); ylabel('abs(F(X(t)) / N')
```

Результат наведений на рис. 5.39.

Як бачимо, "амплітуди" усіх складових гармонік рівні 0.5. При цьому потрібно взяти до уваги, що "амплітуди" розподілені між додатними й від'ємними частотами порівну, тому вони вдвічі менше за справжню амплітуду відповідної гармоніки.

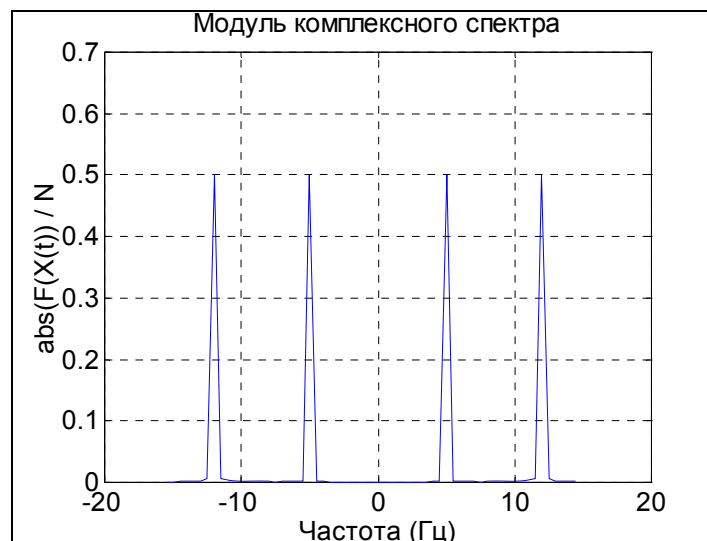


Рис. 5.39. Модуль комплексного спектру

5.3.10. Запитання для самоперевірки

1. Як вводяться вектори в мові Matlab? Якими функціями можна формувати вектори в мові Matlab?
2. Які функції Matlab дозволяють перетворювати вектор поелементно?
3. За допомогою яких засобів у Matlab здійснюються основні операції з векторами?
4. Як здійснити скалярний добуток двох векторів? Які вектори можуть бути скалярно перемножені?
5. Як здійснити векторний добуток двох векторів? Які вектори можуть бути векторно перемножені?
6. Який об'єкт у Matlab називається поліномом?
7. Як у Matlab здійснюється перемноження й ділення поліномів?
8. За допомогою яких функцій можна знайти корені заданого полінома, значення полінома за відомим значенням аргументу?
9. Які функції дозволяють знайти похідну від полінома, поділити один поліном на другий, помножити поліноми?
10. Як розрахувати значення поліному по заданому значенню його аргументу? Яким має бути аргумент – дійсним числом, комплексним числом, вектором з комплексних чисел, матрицею з дійсних чисел?
11. Як можна знайти комплексні корені поліному?
12. Які функції Matlab здійснюють виведення графіків на екран?
13. Якими функціями забезпечується супровід графіка координатними лініями й написами?
14. Що таке "графік вектора" і як його побудувати?
15. Як вивести графік у виді стовпцевої діаграми? Як побудувати гістограму?
16. Чи можна побудувати декілька графіків в одній системі координат і в однім графічному вікні?
17. Як вивести декілька окремих графіків у різних графічних вікнах?
18. Як побудувати декілька окремих графіків в одному графічному вікні?
19. Які функції Matlab дозволяють виводити у графічне вікно текст і як?
20. Що роблять функції *text*, *axis*, *gtext*? Як ними користуватися?
21. Як у Matlab ввести довгий вектор, значення елементів якого утворюють арифметичну прогресію?
22. Чи можна як аргумент у функціях Matlab використовувати вектор? Що це дає?

5.4. Операції з матрицями

5.4.1. Формування матриць

Matlab має декілька функцій, що дозволяють формувати вектори й матриці деякого певного виду. До таких функцій належать:

zeros(M,N) – створює матрицю розміром (M*N) із нульовими елементами, наприклад:

```
» zeros(3,5)
ans =
    0    0    0    0    0
    0    0    0    0    0
    0    0    0    0    0
```

ones(M,N) – створює матрицю розміром (M*N) з одиничними елементами, наприклад:

```
» ones(3,5)
ans =
    1    1    1    1    1
    1    1    1    1    1
    1    1    1    1    1
```

eye(M,N) – створює матрицю розміром (M*N) з одиницями по головній діагоналі й інших нульових елементах, наприклад:

```
» eye(3,5)
ans =
    1    0    0    0    0
    0    1    0    0    0
    0    0    1    0    0
```

rand(M,N) – створює матрицю розміром (M*N) із випадкових чисел, рівномірно розподілених у діапазоні від 0 до 1, наприклад:

```
» rand(3,5)
ans =
 6.1896e-001  6.7930e-001  5.1942e-001  5.3462e-002  7.6982e-003
 4.7045e-002  9.3469e-001  8.3097e-001  5.2970e-001  3.8342e-001
 6.7886e-001  3.8350e-001  3.4572e-002  6.7115e-001  6.6842e-002
```

randn(M,N) – створює матрицю розміром (M*N) із випадкових чисел, розподілених за нормальним (гауссовим) законом із нульовим математичним сподіванням і стандартним (середньоквадратичним) відхиленням, рівним одиниці, наприклад:

```
» randn(3,5)
ans =
 1.1650e+000  3.5161e-001  5.9060e-002  8.7167e-001  1.2460e+000
 6.2684e-001 -6.9651e-001  1.7971e+000 -1.4462e+000 -6.3898e-001
 7.5080e-002  1.6961e+000  6.6407e-001 -7.0117e-001  5.7735e-001
```

hadamard(N) – створює матрицю Адамара розміром (N*N), наприклад:

```
» hadamard(4)
ans =
    1    1    1    1
    1   -1    1   -1
    1    1   -1   -1
    1   -1   -1    1
```

hilb(N) – створює матрицю Гілберта розміром (N*N), наприклад:

```
» hilb(4)
ans =
```

```

1.0000e+000  5.0000e-001  3.3333e-001  6.5000e-001
5.0000e-001  3.3333e-001  6.5000e-001  6.0000e-001
3.3333e-001  6.5000e-001  6.0000e-001  1.6667e-001
6.5000e-001  6.0000e-001  1.6667e-001  1.4286e-001

```

invhilb(N) – створює обернену матрицю Гільберта розміром (N*N), наприклад:

```

» invhilb(4)
ans =
    16          -120         240         -140
   -120     1200    -2700     1680
    240    -2700     6480    -4200
   -140     1680    -4200     2800

```

pascal(N) – створює матрицю Паскаля розміром (N*N), наприклад:

```

» pascal(5)
ans =
    1     1     1     1     1
    1     2     3     4     5
    1     3     6    10    15
    1     4    10    20    35
    1     5    15    35    70

```

У мові Matlab передбачено декілька функцій, що дозволяють формувати матрицю на основі іншої (заданої) або, використовуючи деякий заданий вектор. До таких функцій належать:

fliplr(A) – формує матрицю, переставляючи стовпчики відомої матриці A щодо вертикальної осі, наприклад, якщо

```

A =
    1     2     3     4     5     6
    7     8     9    10    11    12
   13    14    15    16    17    18

```

то застосування цієї функції приводить до результату

```

» fliplr(A)
ans =
    6     5     4     3     2     1
   12    11    10     9     8     7
   18    17    16    15    14    13

```

flipud(A) – формує матрицю, переставляючи рядки заданої матриці A щодо горизонтальної осі, наприклад:

```

» flipud(A)
ans =
   13    14    15    16    17    18
    7     8     9    10    11    12
    1     2     3     4     5     6

```

rot90(A) – формує матрицю шляхом "повороту" заданої матриці A на 90 градусів проти годинникової стрілки:

```

» rot90(A)
ans =
    6    12    18
    5    11    17
    4    10    16

```



```

3  9 15
2  8 14
1  7 13

```

reshape(A,m,n) – утворює матрицю розміром (m*n) шляхом послідовної вибірки елементів заданої матриці A по стовпчиках і наступному розподілі цих елементів по 'n' стовпчиках, кожний з яких містить 'm' елементів; при цьому число елементів матриці A повинно дорівнювати m*n, наприклад:

```

» reshape(A,2,9)
ans =
  1 13  8  3 15 10  5 17 12
  7  2 14  9  4 16 11  6 18

```

tril(A) – утворює нижню трикутну матрицю на основі матриці A шляхом онулювання її елементів вище головної діагоналі:

```

» tril(A)
ans =
  1  0  0  0  0  0
  7  8  0  0  0  0
 13 14 15  0  0  0

```

triu(A) – утворює верхню трикутну матрицю на основі матриці A шляхом онулювання її елементів нижче головної діагоналі:

```

» triu(A)
ans =
  1  2  3  4  5  6
  0  8  9 10 11 12
  0  0 15 16 17 18

```

hankel(V) – утворює квадратну матрицю Ганкеля, перший стовпчик якої збігається із заданим вектором V, наприклад:

```

>> V = [-5 6 7 4]
V =
-5  6  7  4

```

```

» hankel(V)
ans =
-5  6  7  4
  6  7  4  0
  7  4  0  0
  4  0  0  0

```

Процедура **diag**(x) – формує або витягає діагональ матриці.

Якщо x - вектор, то функція **diag**(x) створює квадратну матрицю з вектором x на головній діагоналі:

```

» diag(V)
ans =
-5  0  0  0
  0  6  0  0
  0  0  7  0
  0  0  0  4

```

Щоб установити заданий вектор на іншу діагональ, при зверненні до функції необхідно зазначити ще один параметр (ціле число) - номер діагоналі (при цьому діагоналі відлічуються від головної нагору), наприклад:

```

» diag(V, -1)

```

```
ans =
  0  0  0  0  0
 -5  0  0  0  0
  0  6  0  0  0
  0  0  7  0  0
  0  0  0  4  0
```

Якщо x - матриця, то функція **diag** створює вектор-стовпчик, що складається з елементів головної діагоналі заданої матриці x , наприклад, для матриці A , зазначеної перед прикладом застосування процедури **fliplr**:

```
» diag(A)
ans =
  1
  8
 15
```

Якщо при цьому зазначити додатково номер діагоналі, то можна одержати вектор-стовпчик з елементів будь-якої діагоналі матриці x , наприклад:

```
» diag(A,3)
ans =
  4
 11
 18
```

Функція **zeros(1,N)** формує (створює) вектор-рядок із N нульових елементів. Аналогічно **zeros(N,1)** створює вектор-стовпчик із N нулів.

Вектори, значення елементів яких є випадковими, рівномірно розподіленими, формуються в такий спосіб: **rand(1,n)** - для вектора-рядка і **rand(m,1)** - для вектора-стовпчика.

5.4.2. Витягання й вставляння частин матриць

Насамперед зазначимо, що звернення до будь-якого елемента певної матриці в Matlab здійснюється шляхом указівки (у дужках, через кому) після ймення матриці двох цілих додатних чисел, що визначають відповідно номери рядка й стовпця матриці, на перетинанні яких розташований цей елемент.

Нехай маємо деяку матрицю A :

```
>> A = [ 1 2 3 4; 5 6 7 8; 9 10 11 12]
A =
  1  2  3  4
  5  6  7  8
  9 10 11 12
```

Тоді одержати значення елемента цієї матриці, розташованого на перетинанні другого рядка із третім стовпчиком, можна в такий спосіб:

```
>> A(2,3)
ans = 7
```

Якщо потрібно, навпаки, встановити на це місце деяке число, наприклад, π , то це можна зробити так:

```
>> A(2, 3) = pi; A
A =
  1.0000  6.0000  3.0000  4.0000
  5.0000  6.0000  3.1416  8.0000
  9.0000 10.0000 11.0000 16.0000
```

Іноді потрібно створити меншу матрицю з більшої, формуючи її шляхом витягання з останньої матриці елементів її кількох рядків і стовпчиків, або,

навпаки, вставити меншу матрицю таким чином, щоб вона стала певною частиною матриці більшого розміру. Це в Matlab робиться за допомогою знака двокрапки (" : ").

Розглянемо ці операції на прикладах.

Нехай потрібно створити вектор V1, що складається з елементів третього стовпчика попередньої матриці A. Для цього зробимо такі дії:

```
>> V1 = A(:, 3)
V1 =
    3.0000
    3.1416
   11.0000
```

Щоб створити вектор V2, який складається з елементів другого рядка матриці A, роблять так:

```
>> V2 = A(2, :)
V2 =  5.0000  6.0000  3.1416  8.0000
```

Припустимо, що необхідно з матриці A утворити матрицю B розміром (2*2), яка складається з елементів лівого нижнього рогу матриці A. Тоді роблять таке:

```
>> B = A(2:3, 1:2)
B =
    5    6
    9   10
```

Аналогічно можна вставити матрицю B в верхню середину матриці A:

```
>> A(1:2,2:3)=B
A =
    1    5    6    4
    5    9   10    8
    9   10   11   12
```

Як очевидно, для цього замість указівки номерів елементів матриці можна вказувати діапазон змінювання цих номерів шляхом указівки нижньої й верхньої меж, розділяючи їх двокрапкою.

Примітка. Якщо верхньою межею змінювання номерів елементів матриці є її розмір у цьому вимірі, замість нього можна використовувати службове слово *end*. Наприклад:

```
>> A(2:end,2:end)
ans =
    9   10    8
   10   11   12
```

Ці операції дуже зручні для формування матриць, більшість елементів яких однакові, зокрема, так званих розріджених матриць, що складаються, в основному, із нулів, за винятком окремих елементів. Для прикладу розглянемо формування розрідженої матриці розміром (5*7) з одиничними елементами в її центрі:

```
>> A = zeros(5,7);
>> B = ones(3,3);
>> A(2:4,3:5)=B
A =
    0    0    0    0    0    0    0
    0    0    1    1    1    0    0
    0    0    1    1    1    0    0
    0    0    1    1    1    0    0
    0    0    0    0    0    0    0
```

"Розтягнути" матрицю (A) у єдиний вектор (V) можна за допомогою звичайного запису $V = A(:)$. При цьому створюється вектор-стовпчик із кількістю елементів ($m \cdot n$), у якому стовпчики заданої матриці розміщені зверху униз у порядку самих стовпчиків:

```
» A = [1 2 3; 4 5 6]
```

```
A =
```

```
1 2 3
4 5 6
```

```
» v = A(:)
```

```
v =
```

```
1
4
2
5
3
6
```

Нарешті, "розширювати" матрицю, укладаючи її з окремих заданих матриць ("блоків") можна теж досить просто. Якщо задані кілька матриць-блоків A1, A2,... AN з однаковою кількістю рядків, то з них можна "зліпити" єдину матрицю A, об'єднуючи блоки в один "рядок" у такий спосіб:

$$A = [A1, A2, \dots, AN].$$

Цю операцію називають горизонтальною конкатенацією (зчепленням) матриць. Аналогічно, вертикальна конкатенація матриць реалізується (за умови, що всі складові блоки-матриці мають однакову кількість стовпчиків) аналогічним чином, шляхом застосування для відділення блоків замість коми крапки з комою:

$$A = [A1; A2; \dots ; AN].$$

Наведемо приклади. Приклад горизонтальної конкатенації :

```
>> A1 = [1 2 3; 4 5 6; 7 8 9];
```

```
>> A2 = [10;11;12];
```

```
>> A3 = [14 15; 16 17; 18 19];
```

```
>> A = [A1, A2, A3]
```

```
A =
```

```
1 2 3 10 14 15
4 5 6 11 16 17
7 8 9 12 18 19
```

Приклад вертикальної конкатенації:

```
>> B1 = [1 2 3 4 5];
```

```
>> B2 = [ 6 7 8 9 10; 11 12 13 14 15];
```

```
>> B3 = [17 18 19 20 21];
```

```
>> B = [ B1; B2; B3]
```

```
B =
```

```
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
17 18 19 20 21
```

5.4.3. Обробка даних вимірів

Система Matlab дає користувачеві додаткові можливості для обробки даних, що задані у векторній або матричній формі.

Припустимо, що є деяка залежність $y(x)$, яка задана рядом точок

x	2	4	6	8	10
y	5.5	6.3	6.8	8	8.6

Її можна задати в командному вікні Matlab як матрицю **xydata**, що містить два рядки - значення x і значення y :

```
>> xydata =[2 4 6 8 10; 5.5 6.3 6.8 8 8.6]
xydata =
 6.0000  4.0000  6.0000  8.0000 10.0000
 5.5000  6.3000  6.8000  8.0000  8.6000
```

На прикладі цієї залежності розглянемо основні засоби для обробки даних.

Функція **size(xydata)** призначена для визначення числа рядків і стовпчиків матриці **xydata**. Вона формує вектор $[n, p]$, який містить ці величини:

```
>> size(xydata)
ans =
 2 5
```

Звернення до неї виду

```
>> [n, p] = size(xydata);
```

дозволяє зберегти в пам'яті машини і використовувати потім при подальших обчисленнях дані про кількість рядків n і кількість стовпчиків p цієї матриці:

```
>> n, p
n =
 2
p =
 5
```

За допомогою цієї функції можна встановити довжину й тип (рядок або стовпчик) вектора :

```
» v = xydata(:)
v =
 6.0000
 5.5000
 4.0000
 6.3000
 6.0000
 6.8000
 8.0000
 8.0000
10.0000
 8.6000
» n = size(v)
n = 10 1
» v1 = v'
v1 =
Columns 1 through 7
 6.0000  5.5000  4.0000  6.3000  6.0000  6.8000  8.0000
Columns 8 through 10
 8.0000 10.0000  8.6000
» size(v')
ans = 1 10
```

Функція **max(V)**, де V - деякий вектор, видає значення максимального елемента цього вектора. Аналогічно, функція **min(V)** витягає мінімальний елемент вектора V . Функції **mean(V)** і **std(V)** визначають, відповідно, середнє значення і середньоквадратичне відхилення від нього значень елементів вектора V .

Функція сортування *sort(V)* формує вектор, елементи якого розподілені в порядку зростання їхніх значень.

Функція *sum(V)* обчислює суму елементів вектора V.

Функція *prod(V)* видає добуток усіх елементів вектора V.

Функція *cumsum(V)* формує вектор того ж типу й розміру, будь-який елемент якого є сумою всіх попередніх елементів вектора V (вектор кумулятивної суми).

Функція *cumprod(V)* створює вектор, елементи якого є добутком усіх попередніх елементів вектора V.

Функція *diff(V)* видає вектор, що має розмір на одиницю менший за розмір вектора V, елементи якого є різницею між суміжними елементами вектора V.

Застосування описаних функцій проілюстровано нижче.

» `v = [1, 0.1, 0.5, 0.1, 0.1, 0.4];`

» `disp(size(v))`

1 6

» `disp(max(v))`

1

» `disp(min(v))`

0.1000

» `disp(mean(v))`

0.3667

» `disp(std(v))`

0.3559

» `disp(sort(v))`

0.1000 0.1000 0.1000 0.4000 0.5000 1.0000

» `disp(sum(v))`

6.2000

» `disp(prod(v))`

6.0000e-004

» `disp(cumsum(v))`

1.0000 1.1000 1.6000 1.7000 1.8000 6.2000

» `disp(cumprod(v))`

1.0000 0.1000 0.0500 0.0050 0.0005 0.0002

» `disp(diff(v))`

-0.9000 0.4000 -0.4000 0 0.3000

Якщо вказати другий вихідний параметр, то можна одержати додаткову інформацію про індекс першого елемента, значення якого є максимальним або мінімальним:

>> `[M,n]=max(v)`

M = 1

n = 1

>> `[N,m]=min(v)`

N = 0.1000

m = 2

Інтегрування методом трапецій здійснює процедура *trapz*. Звернення до неї вигляду *trapz(x,y)* призводить до обчислення площі під графіком функції $y(x)$, у якому всі точки, задані векторами x і y , з'єднані відрізками прямих. Якщо перший вектор x не зазначений у зверненні, за умовчанням припускається, що крок інтегрування дорівнює одиниці (тобто вектор x є вектором із номерів елементів вектора y).

Приклад. Обчислимо інтеграл від функції $y = \sin(x)$ у діапазоні від 0 до π . Його точне значення дорівнює 6. Візьмемо рівномірну сітку із 100 елементів. Тоді обчислення зведуться до сукупності операцій:

```
» x = 0 : pi/100 : pi;
» y = sin(x);
» disp(trapz(x,y))
1.9998
```

Ті ж функції *size*, *max*, *min*, *mean*, *std*, *sort*, *sum*, *prod*, *cumsum*, *cumprod*, *diff* можуть бути застосовані і до матриць. Основною відмінністю використання як аргументів цих функцій саме матриць є те, що відповідні описані вище операції провадяться не по відношенню до рядків матриць, а до кожного зі стовпців заданої матриці. Тобто кожний стовпець матриці *A* розглядається як змінна, а кожний рядок – як окреме спостереження. Так, у результаті застосування функцій *max*, *min*, *mean*, *std* утворюються вектори-рядки з кількістю елементів, яка дорівнює кількості стовпців заданої матриці. Кожний елемент містить, відповідно, максимальне, мінімальне, середнє або середньоквадратичне значення елементів відповідного стовпця заданої матриці.

Наведемо приклади. Нехай маємо 3 величини y_1 , y_2 і y_3 , що виміряні за деяких п'яти значень аргументу (які не зазначені). Тоді дані вимірів утворять 3 вектори по 5 елементів:

```
>> y1 = [ 5.5 6.3 6.8 8 8.6];
>> y2 = [-1. 2 0.5 -0.6 1 0.1];
>> y3 = [ 3.4 5.6 0 8.4 10.3]; .
```

Сформуємо з них матрицю вимірів так, щоб вектори y_1 , y_2 , y_3 утворювали стовпці цієї матриці:

```
» A = [ y1', y2', y3' ]
A =
5.5000 -1.2000 3.4000
6.3000 0.5000 5.6000
6.8000 -0.6000 0
8.0000 1.0000 8.4000
8.6000 0.1000 10.3000
```

Застосуємо до цієї матриці вимірів описані вище функції. Одержимо

```
» size(A)
ans = 5 3
» max(A)
ans = 8.6000 1.0000 10.3000
» min(A)
ans = 5.5000 -1.2000 0
» mean(A)
ans = 7.0400 -0.0400 5.5400
» std(A)
ans = 1.2582 0.8735 4.0655
```

Якщо при зверненні до функцій *max* і *min* зазначити другий вихідний параметр, то він дасть інформацію про номер рядка, де знаходиться у відповідному стовпчику перший елемент із максимальним (або мінімальним) значенням. Наприклад:

```
>> [M,n]=max(A)
M = 8.6000 1.0000 10.3000
n = 5 4 5
>> [N,m]=min(A)
N = 5.5000 -1.2000 0
```

m = 1 1 3

Функція **sort** сортує елементи кожного зі стовпців матриці. Результатом є матриця того ж розміру.

Функції **sum** і **prod** формують вектор-рядок, кожний елемент якого є сумою або добутком елементів відповідного стовпця початкової матриці.

Функції **cumsum**, **cumprod** утворюють матриці того самого розміру, елементи кожного стовпця яких є сумою або добутком елементів цього ж стовпця початкової матриці, починаючи з відповідного елемента і вище.

Нарешті, функція **diff** створює із заданої матриці розміром (m*n) матрицю розміром ((m-1)*n), елементи якої є різницею між елементами суміжних рядків початкової матриці.

Застосовуючи ці процедури до тієї самої матриці *A* вимірів, одержимо:

```

» sort(A)
ans =
    5.5000   -1.2000    0
    6.3000   -0.6000    3.4000
    6.8000    0.1000    5.6000
    8.0000    0.5000    8.4000
    8.6000    1.0000   10.3000
» sum(A)
ans = 35.2000  -0.2000  27.7000
» prod(A)
ans = 1.0e+004 *
    1.6211    0.0000    0
» cumsum(A)
ans =
    5.5000   -1.2000    3.4000
   11.8000   -0.7000    9.0000
   18.6000   -1.3000    9.0000
   26.6000   -0.3000   17.4000
   35.2000  -0.2000   27.7000
» cumprod(A)
ans = 1.0e+004 *
    0.0006  -0.0001    0.0003
    0.0035  -0.0001    0.0019
    0.0236    0.0000    0
    0.1885    0.0000    0
    1.6211    0.0000    0
» diff(A)
ans =
    0.8000    1.7000    6.2000
    0.5000   -1.1000   -5.6000
    1.2000    1.6000    8.4000
    0.6000   -0.9000    1.9000

```

Розглянемо деякі інші функції, надані користувачеві системою Matlab.

Функція **cov(A)** обчислює *матрицю коваріацій* вимірів. При цьому утворюється квадратна симетрична матриця з кількістю рядків і стовпчиків, рівним кількості виміряних величин, тобто кількості стовпчиків матриці вимірів.

Наприклад, при застосуванні до прийнятої матриці вимірів вона дає такий результат:

```

» cov(A)
ans =
    1.5830    0.6845    3.6880

```



```
0.6845 0.7630 6.3145
3.6880 6.3145 16.5280
```

На діагоналі матриці коваріацій розміщені *дисперсії* виміряних величин, а поза нею – *взаємні кореляційні моменти* цих величин.

Функція **corrcoef(A)** обчислює *матрицю коефіцієнтів кореляції* за тих самих умов. Елементи матриці $S = \text{corrcoef}(A)$ пов'язані з елементами матриці коваріацій $C = \text{cov}(A)$ таким співвідношенням:

$$S(k, l) = \frac{C(k, l)}{\sqrt{C(k, k) \cdot C(l, l)}}$$

Приклад:

» **corrcoef(A)**

ans =

```
1.0000 0.6228 0.7210
0.6228 1.0000 0.6518
0.7210 0.6518 1.0000
```

5.4.4. Поелементне перетворення матриць

Для поелементного перетворення матриці придатні всі зазначені раніше в п. 5.6.4 алгебричні функції. Кожна така функція формує матрицю того самого розміру, що й задана, кожний елемент якої обчислюється як зазначена функція від відповідного елемента заданої матриці. Крім цього, у Matlab визначені операції *поелементного множення* матриць однакового розміру (сполученням **".*"**, що записується між іменами матриць, що перемножуються), *поелементного ділення* (сполученням **"/."** і **".\"**), *поелементного піднесення до степеня* (сполученням **".^"**), коли кожний елемент першої матриці підноситься до степеня, який дорівнює значенню відповідного елемента другої матриці.

Наведемо кілька прикладів:

» **A = [1,2,3,4,5; -2, 3, 1, 4, 0]**

A =

```
1 2 3 4 5
-2 3 1 4 0
```

» **B = [-1,3,5,-2,1; 1,8,-3,-1,2]**

B =

```
-1 3 5 -2 1
1 8 -3 -1 2
```

» **sin(A)**

ans =

```
0.8415 0.9093 0.1411 -0.7568 -0.9589
-0.9093 0.1411 0.8415 -0.7568 0
```

» **A .* B**

ans =

```
-1 6 15 -8 5
-2 24 -3 -4 0
```

» **A ./ B**

ans =

```
-1.0000 0.6667 0.6000 -6.0000 5.0000
-6.0000 0.3750 -0.3333 -4.0000 0
```

» **A .\ B**

Warning: Divide by zero

ans =

```

-1.0000  1.5000  1.6667 -0.5000  0.2000
-0.5000  6.6667 -3.0000 -0.2500   Inf
» A.^B
ans =
1.0e+003 *
  0.0010  0.0080  0.2430  0.0001  0.0050
-0.0020  6.5610  0.0010  0.0002   0

```

Оригінальною в мові Matlab є операція додавання до матриці числа. Вона записується в такий спосіб: $A + x$, або $x + A$ (A – матриця, а x – число). Такої операції немає в математиці. У Matlab вона є еквівалентною до сукупності операцій

$$A + x * E,$$

де E – позначення матриці, що складається саме з одиниць, тих самих розмірів, що і матриця A . Наприклад:

```

» A = [ 1 2 3 4 5; 6 7 8 9 11 ]
A =
  1  2  3  4  5
  6  7  8  9 11
» A + 2
ans =
  3  4  5  6  7
  8  9 10 11 13
» 2 + A
ans =
  3  4  5  6  7
  8  9 10 11 13

```

5.4.5. Матричні дії над матрицями

До матричних дій над матрицями відносять такі операції, які використовуються в матричному численні в математиці і не суперечать йому.

Базові дії з матрицями – *додавання, віднімання, транспонування, множення матриці на число, множення матриці на матрицю, піднесення матриці до цілого степеня* – здійснюються в мові Matlab за допомогою звичайних знаків арифметичних операцій. При використуванні цих операцій *важливо пам'ятати умови, за яких ці операції є можливими:*

при додаванні або відніманні матриць вони повинні мати однакові розміри;

при множенні матриць кількість стовпців першої матриці повинна збігатися з кількістю рядків другої матриці.

Невиконання цих умов призведе до появи в командному вікні повідомлення про помилку. Наведемо кілька прикладів.

Приклад додавання й віднімання:

```

» A = [ 1 2 3 4 5; 6 7 8 9 11 ]
A =
  1  2  3  4  5
  6  7  8  9 11
» B = [ 0 -1 -2 -3 -4; 5 6 7 8 9 ]
B =
  0 -1 -2 -3 -4

```

```

5 6 7 8 9
» A + B
ans =
1 1 1 1 1
11 13 15 17 20
» A - B
ans =
1 3 5 7 9
1 1 1 1 6.

```

Приклад множення на число:

```

» 5*A
ans =
5 10 15 20 25
30 35 40 45 55
» A*5
ans =
5 10 15 20 25
30 35 40 45 55.

```

Приклад транспонування матриці:

```

» A'
ans =
1 6
2 7
3 8
4 9
5 11.

```

Приклад множення матриці на матрицю:

```

» A' * B
ans =
30 35 40 45 50
35 40 45 50 55
40 45 50 55 60
45 50 55 60 65
55 61 67 73 79
» C = A * B'
C =
-40 115
-94 299.

```

Функція **обернення матриці** - $\text{inv}(A)$ – обчисляє матрицю, обернену до заданої матриці A . Початкова матриця A повинна бути квадратною, а її визначник не повинен дорівнювати нулеві.

Наведемо приклад:

```

» inv(C)
ans =
-6.6000e-001 1.0000e-001
-8.1739e-002 3.4783e-002

```

Перевіримо слушність виконання операції обернення, застосовуючи її ще раз до отриманого результату:

```

» inv(ans)
ans =
-4.0000e+001 1.1500e+002
-9.4000e+001 6.9900e+002

```

Як бачимо, ми одержали початкову матрицю C , що є ознакою правильності виконання обернення матриці.

Піднесення матриці до цілого степеня здійснюється в Matlab за допомогою знака " \wedge ": A^n . При цьому матриця має бути квадратною, а n

має бути цілим (додатним або від'ємним) числом. Ця матрична дія є еквівалентною до множення матриці A на себе n разів (якщо n - додатне) або множенню оберненої матриці на себе (при n від'ємному).

Наведемо приклад:

» A^2

ans =

```
8  -3  -10
-5  10  16
-2  4   9
```

» $A^{(-2)}$

ans =

```
1.5385e-001 -7.6923e-002  3.0769e-001
7.6923e-002  3.0769e-001 -4.6154e-001
6.1328e-018 -1.5385e-001  3.8462e-001
```

Дуже оригінальними в мові Matlab є дві нові, невідомі в математиці функції **ділення матриць**. При цьому вводяться поняття **ділення матриць зліва направо** і **ділення матриць справа наліво**. Перша операція записується за допомогою знаку $/$, а друга - \backslash , які розташовуються між іменами матриць, які діляться.

Операція B / A еквівалентна послідовності дій $B * inv(A)$, де функція inv здійснює **обернення матриці**. Її зручно використовувати для розв'язування матричного рівняння:

$$X * A = B.$$

Аналогічно операція $A \backslash B$ рівносильна сукупності операцій $inv(A) * B$, що є розв'язком матричного рівняння:

$$A * X = B.$$

5.4.6. Розв'язування систем лінійних алгебричних рівнянь

Система лінійних алгебричних рівнянь (СЛАР) n -го порядку має вигляд:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \dots \quad \dots \quad \dots \quad \dots = \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases}$$

Тут позначено: x_i ($i = 1, 2, \dots, n$) – деякі змінні, a_{ij} ($i, j = 1, 2, \dots, n$) – коефіцієнти при змінних, b_i ($i = 1, 2, \dots, n$) – так звані "вільні" члени.

Під розв'язуванням СЛАР розуміється відшукування таких значень змінних x_i , підстановка яких у кожне з n рівнянь, перетворює їх одночасно у тотожності.

Якщо використати матричні позначення:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}; \quad B = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}; \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix},$$

то СЛАР може бути поданою у матричній формі у такий спосіб:

$$AX = B. \quad (1)$$

У системі Matlab розв'язування рівняння (1) здійснюється вельми просто, з використанням дії зворотного ділення. Для прикладу розглянемо задачу відшукування коренів системи лінійних алгебричних рівнянь:

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 14 \\ 2x_1 - x_2 - 5x_3 = -15 \\ x_1 - x_2 = x_3 = -4 \end{cases}$$

Це можна зробити у такий спосіб:

```
» A = [ 1 2 3; 2 -1 -5; 1 -1 -1]
```

```
A =
```

```
 1  2  3
 2 -1 -5
 1 -1 -1
```

```
» B = [ 14;-15;-4]
```

```
B =
```

```
 14
-15
 -4
```

```
» x = A \ B
```

```
x =
```

```
 1
 2
 3
```

Розглянутий метод є точним. Точні (або прямі) методи працюють достатньо швидко і широко застосовуються на практиці, якщо є достатніми обсяги пам'яті для їхньої реалізації.

Але існують й інші шляхи відшукування коренів СЛАР, які відносяться до наближених. Наприклад, різні модифікації *методу ітерацій*. Вони знаходять за скінченну кількість кроків (ітерацій) лише наближені розв'язки із заданою припустимою відносною похибкою.

Метод ітерацій у загальному випадку полягає у тому, що попередньо первісне рівняння $f(x) = 0$ перетворюється до виду

$$x = \varphi(x). \quad (2)$$

З області ізоляції $[a, b]$ шуканого кореня обирається деяке певне значення x_0 аргументу, яке приймається за початкове наближення кореня. Наближені значення кореня у наступних наближеннях визначаються з співвідношення

$$x_k = \varphi(x_{k-1}); \quad (k = 1, 2, \dots, n), \quad (3)$$

при цьому k має зміст номера ітерації (наближення).

Очевидно, ітераційний процес може приводити до послідовності значень, які наближаються до деякого значення аргумента (шуканого кореня), і тоді цей

процес є стійким. Але такий процес може приводити і до послідовності значень аргумента, які все далі віддаляються від первісного значення, тобто бути нестійким. Стійкість або нестійкість ітераційного процесу суттєво залежить від виду залежності $\varphi(x)$, яка, як неважко впевнитися, визначається неоднозначно, і тому можна підібрати з можливих її варіантів і такі, які забезпечують стійкість ітерацій. Саме це й є найбільш складним у побудові ітераційного процесу.

Методи ітерацій можуть бути застосовані і для відшукування коренів СЛАР, якщо попередньо систему (1) подати у виді

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_n) \\ x_2 = \varphi_2(x_1, x_2, \dots, x_n) \\ \dots\dots\dots \\ x_n = \varphi_n(x_1, x_2, \dots, x_n) \end{cases} \quad (4)$$

MatLAB має кілька вбудованих процедур, які дозволяють розв'язувати систему лінійних алгебричних рівнянь виду (1) наближеними ітераційними методами. Їх застосовують при обчисленнях тоді, коли матриці коефіцієнтів СЛАР є розрідженими і великими за розмірами. До них відносяться:

bicg – метод біспряжених градієнтів;

bicgstab – стабілізований метод біспряжених градієнтів;

cgs – квадратичний метод спряжених градієнтів;

gmres – узагальнений метод мінімального відхилення;

qmr – метод квазімінімального відхилення;

pcg – передобумовлений метод спряжених градієнтів (застосовується лише для симетричних матриць A).

Загальне звернення до цих процедур має вигляд

$x = \text{bicg}(A, B, \text{tol}, \text{maxit})$.

Тут A – квадратна матриця розміром $(n \times n)$ коефіцієнтів при аргументах системи рівнянь (1), B – матриця-стовпець розміром $(n \times 1)$ вільних членів, tol – припустима межа відносна похибка визначення коренів, maxit – межа припустима кількість ітерацій, x – вектор одержаних наближених значень коренів рівняння (1).

За початкове наближення обирається вектор x_0 з нульових елементів.

Наведемо приклад. Введемо наступну послідовність операторів:

```
A=[1 1 2 3;3 -1 -1 -2;2 3 -1 -1; 1 2 3 -1]
```

```
B=[1; -4; -6; -4]
```

```
X=A\B;
```

```
X1=bicg(A,B);
```

```
X2=bicgstab(A,B);
```

```
X3=cgs(A,B);
```

```
X4=gmres(A,B);
```

```
X5=qmr(A,B);
```

```
disp(' ')
```

```
disp(' ')
```

```
disp([' Точно ',' BICG ',' BICGSTAB ',' CGs ',' GMRES ',' QMR '])
```

```
Y=[X X1 X2 X3 X4 X5]
```

```
disp(' ')
```

```
disp(' ПОХИБКИ')
```

```
disp([' BICG ',' BICGSTAB ',' CGs ',' GMRES ',' QMR '])
```

$dY=[X1-X X2-X X3-X X4-X X5-X]$

Виконавши їх, система MATLAB виведе результат у наступному вигляді:

```
A =
  1   1   2   3
  3  -1  -1  -2
  2   3  -1  -1
  1   2   3  -1
B =
  1
 -4
 -6
 -4

Точно      BICG      BICGSTAB      CGS      GMRES      QMR
Y =
-1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000
-1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000
  0          3.5638e-014  3.8956e-011  -5.8953e-014  -1.6653e-016  -4.1078e-015
 1.0000e+000 1.0000e+000 1.0000e+000 1.0000e+000 1.0000e+000 1.0000e+000

      П О Х И Б К И
      BICG      BICGSTAB      CGs      GMRES      QMR
dY =
 1.0436e-014 -4.6485e-012 -7.4163e-014  6.2204e-016 -3.1530e-014
-1.7319e-014  6.6083e-011  6.5535e-015  4.4409e-016  8.5487e-015
 3.5638e-014  3.8956e-011 -5.8953e-014 -1.6653e-016 -4.1078e-015
 4.8406e-014  6.0103e-011 -6.8422e-014  0          -6.0428e-014
```

Як бачимо, похибки майже усіх цих наближених методів (за виключенням методу *bicgstab*) близьки до граничної похибки округлення в системі Matlab.

5.4.7. Матричні функції

Окрім вищезазначених, у Matlab передбачені ще такі матричні функції й операції.

Операція *транспонування матриці* здійснюється за допомогою знака апострофа, записаного після ймення матриці, наприклад:

```
X=[1 2 3; 4 5 6]
X =
  1   2   3
  4   5   6
X'
ans =
  1   4
  2   5
  3   6
```

Обчислення *матричної експоненти* (e^A) здійснюється за допомогою функцій *expm*, *expm1*, *expm2*, *expm3*. Ці функції варто відрізнити від раніше розглянутої функції *exp(A)*, яка формує матрицю, кожний елемент якої дорівнює e в степені, що дорівнює відповідному елементу матриці A.

Функція *expm* є вмонтованою функцією Matlab. Функція *expm1(A)* є М-файлом, який обчислює матричну експоненту шляхом використання розкладення Паде матриці A. Функція *expm2(A)* обчислює матричну

експоненту, використовуючи розкладення Тейлора матриці A . Функція $\text{expm3}(A)$ обчислює матричну експоненту на основі використання спектрального розкладу A .

Наведемо приклади використання цих функцій:

» $A = [1,2,3; 0, -1,5; 7, -4, 1]$

$A =$

```
1  2  3
0 -1  5
7 -4  1
```

» $\text{expm}(A)$

ans =

```
131.3648  -9.5601  80.6685
 97.8030  -7.1768  59.9309
123.0245  -8.8236  75.4773
```

» $\text{expm1}(A)$

ans =

```
131.3648  -9.5601  80.6685
 97.8030  -7.1768  59.9309
123.0245  -8.8236  75.4773
```

» $\text{expm2}(A)$

ans =

```
131.3648  -9.5601  80.6685
 97.8030  -7.1768  59.9309
123.0245  -8.8236  75.4773
```

» $\text{expm3}(A)$

ans =

```
1.0e+002 *
1.3136 + 0.0000i  -0.0956 + 0.0000i   0.8067 - 0.0000i
0.9780 + 0.0000i  -0.0718 - 0.0000i   0.5993 - 0.0000i
1.2302 + 0.0000i  -0.0882 - 0.0000i   0.7548 - 0.0000i
```

Функція $\text{logm}(A)$ робить обернену операцію - логарифмування матриці за натуральною основою, наприклад:

$A =$

```
1  2  3
0  1  5
7  4  1
```

» $B = \text{expm3}(A)$

$B =$

```
1.0e+003 *
0.9378  0.7987  0.9547
1.0643  0.9074  1.0844
1.5182  1.2932  1.5459
```

» $\text{logm}(B)$

ans =

```
1.0000  6.0000  3.0000
0.0000  1.0000  5.0000
7.0000  4.0000  1.0000
```

Функція $\text{sqrtn}(A)$ обчислює таку матрицю Y , що $Y*Y = A$:

» $Y = \text{sqrtn}(A)$

$Y =$

```
0.7884 + 0.8806i   0.6717 - 0.1795i   0.8029 - 0.4180i
```



```

0. 8953 + 0. 6508i    0. 7628 + 0. 8620i    0. 9118 - 1. 0066i
1.2765 - 1. 4092i    1. 0875 - 0. 5449i    1. 3000 + 1. 2525i

```

```
» Y * Y
```

```
ans =
```

```

1. 0000 + 0. 0000i    6. 0000 - 0. 0000i    3. 0000 + 0. 0000i
0. 0000 - 0. 0000i    1. 0000 - 0. 0000i    5. 0000 - 0. 0000i
7.0000 + 0. 0000i    4. 0000 + 0. 0000i    1. 0000 + 0. 0000i

```

5.4.8. Функції лінійної алгебри

Традиційно до лінійної алгебри відносять такі задачі, як обернення і псевдообернення матриці, спектральне й сингулярне розкладання матриць, обчислення власних значень і векторів, сингулярних чисел матриць, обчислення функцій від матриць. Коротко ознайомимося з деякими основними функціями Matlab у цій області.

Функція $k = \mathit{cond}(A)$ обчислює й видає число обумовленості матриці стосовно операції обернення, яке дорівнює відношенню максимального сингулярного числа матриці до мінімального.

Функція $k = \mathit{norm}(v, p)$ обчислює p -норму вектора v за формулою:

$$k = \mathit{sum}(\mathit{abs}(v) . ^p)^{(1/p)},$$

де p - ціле додатне число. Якщо аргумент p при зверненні до функції не зазначений, обчислюється 2-норма.

Функція $k = \mathit{norm}(A, p)$ обчислює p -норму матриці A за формулою:

$$k = \mathit{max}(\mathit{sum}(\mathit{abs}(A) . ^p)^{(1/p)},$$

де $p = 1, 2, 'fro'$ або inf . Якщо аргумент p не зазначений, обчислюється 2-норма. При цьому є слушними співвідношення:

$$\begin{aligned} \mathit{norm}(A, 1) &= \mathit{max}(\mathit{sum}(\mathit{abs}(A))); \\ \mathit{norm}(A, \mathit{inf}) &= \mathit{max}(\mathit{sum}(\mathit{abs}(A'))); \\ \mathit{norm}(A, 'fro') &= \mathit{sqrt}(\mathit{sum}(\mathit{diag}(A * A))); \\ \mathit{norm}(A) &= \mathit{norm}(A, 2) = \sigma_{\mathit{max}}(A). \end{aligned}$$

Функція $rd = \mathit{rcond}(A)$ обчислює величину, обернену значенню числа обумовленості матриці A щодо 1-норми. Якщо матриця A добре обумовлена, значення rd близько до одиниці. Якщо ж вона погано обумовлена, rd наближається до нуля.

Функція $r = \mathit{rank}(A)$ обчислює ранг матриці, який визначається як кількість сингулярних чисел матриці, що перевищують поріг

$$\mathit{max}(\mathit{size}(A)) * \mathit{norm}(A) * \mathit{eps}.$$

Наведемо приклади застосування цих функцій:

```
A =
```

```

1  2  3
0  1  5
7  4  1

```

```
» disp(cond(A))
```

```
13. 8032
```

```
» disp(norm(A,1))
```

```
9
```

```
» disp(norm(A))
```

```
8. 6950
```

```
» disp(rcond(A))
0.0692
» disp(rank(A))
3
```

Процедура $d = \det(A)$ обчислює *визначник квадратної матриці* на основі трикутного розкладання методом виключення Гаусса.

Функція $t = \text{trace}(A)$ обчислює *слід матриці* A , який дорівнює сумі її діагональних елементів.

$Q = \text{null}(A)$ обчислює ортонормований базис *нуль-простору* матриці A .

$Q = \text{orth}(A)$ видає *ортонормований базис матриці* A .

Процедура $R = \text{rref}(A)$ здійснює *приведення матриці до трикутного виду на основі методу виключення Гаусса з частковим вибором провідного елемента*.

Приклади:

```
» disp(det(A))
30
» disp(trace(A))
3
» disp(null(A))
» disp(orth(A))
0.3395 0.4082 -0.8474
0.2793 0.8165 0.5053
0.8982 -0.4082 0.1632
» disp(rref(A))
1 0 0
0 1 0
0 0 1
```

Функція $R = \text{chol}(A)$ здійснює *розкладання Холецького* для дійсних симетричних і комплексних ермітових матриць. Наприклад:

```
» A = [ 1 2 3; 2 15 8; 3 8 400]
A =
1 2 3
2 15 8
3 8 400
» disp(chol(A))
1.0000 6.0000 3.0000
0 3.3166 0.6030
0 0 19.7645
```

Функція $lu(A)$ здійснює *LU-розкладання* матриці A в виді добутку нижньої трикутної матриці L (можливо, із перестановками) і верхньої трикутної матриці U так, що $A = L * U$.

Звернення до цієї функції виду

```
[ L, U, P ] = lu(A)
```

дозволяє одержати три складові цього розкладання - нижню трикутну матрицю L , верхню трикутну U і матрицю перестановок P такі, що

$P * A = L * U$.

Наведемо приклад:

```
A =
1 2 3
2 15 8
3 8 400
» disp(lu(A))
3.0000 8.0000 400.0000
```

```

-0.6667  9.6667 -258.6667
-0.3333  0.0690 -148.1724
» [ L, U, P ] = lu(A);
» L
L =
  1.0000    0    0
  0.6667    1.0000    0
  0.3333   -0.0690    1.0000
» U
U =
  3.0000  8.0000  400.0000
    0    9.6667 -258.6667
    0    0 -148.1724
» P
P =
  0  0  1
  0  1  0
  1  0  0

```

З нього випливає, що в першому, спрощеному варіанті звернення функція видає комбінацію з матриць L і U.

Обернення матриці здійснюється за допомогою функції *inv(A)*:

```

» disp(inv(A))
  1.3814  -0.1806  -0.0067
 -0.1806   0.0910  -0.0005
 -0.0067  -0.0005   0.0026

```

Процедура *pinv(A)* знаходить матрицю, *псевдообернену* матриці A, яка має розміри матриці A^T і задовольняє умови

$$A * P * A = A;$$

$$P * A * P = P.$$

Наприклад:

```

A =
  1  2  3  4  5
  5 -1  4  6  0
» P = pinv(A)
P =
 -0.0423  0.0852
  0.0704 -0.0480
  0.0282  0.0372
  0.0282  0.0628
  0.1408 -0.0704
» A*P*A,          % перевірка 1
ans =
  1.0000  6.0000  3.0000  4.0000  5.0000
  5.0000 -1.0000  4.0000  6.0000  0.0000
» P*A*P          % перевірка 2
ans =
 -0.0423  0.0852
  0.0704 -0.0480
  0.0282  0.0372
  0.0282  0.0628
  0.1408 -0.0704

```

Для квадратних матриць ця операція рівнозначна звичайному оберненню.

Процедура $[Q, R, P] = qr(A)$ здійснює розкладення матриці A на три - унітарну матрицю Q, верхню трикутну R із діагональними елементами, що зменшуються за модулем, і матрицю перестановок P такі що

$$A * P = Q * R.$$

Наприклад:

```

A =
    1     2     3     4     5
    5    -1     4     6     0
» [Q,R,P] = qr(A)
Q =
   -0.5547  -0.8321
   -0.8321   0.5547
R =
   -7.2111  -6.7735  -4.9923  -4.7150  -0.2774
    0    -4.1603  -0.2774   1.9415  -6.2188
P =
    0     0     0     1     0
    0     0     0     0     1
    0     0     1     0     0
    1     0     0     0     0
    0     1     0     0     0

```

Визначення характеристичного полінома матриці A можна здійснити за допомогою функції $\text{poly}(A)$. Звернення до неї виду $p = \text{poly}(A)$ дає можливість знайти вектор-рядок p коефіцієнтів характеристичного полінома

$$p(s) = \det(s * E - A) = p_1 * s^n + \dots + p_n * s + p_{n+1},$$

де E - позначення одиничної матриці розміром $(n * n)$. Наприклад :

```

» A = [1 2 3; 5 6 0; -1 2 3]
A =
    1     2     3
    5     6     0
   -1     2     3
» p = poly(A)
p =
    1.0000  -10.0000   20.0000  -36.0000

```

Обчислення власних значень і власних векторів матриці здійснює процедура $\text{eig}(A)$. Звичайне звернення до неї дозволяє одержати вектор власних значень матриці A , тобто коренів характеристичного полінома матриці. Якщо ж звернення має вид:

$$[R, D] = \text{eig}(A),$$

то в результаті одержують діагональну матрицю D власних значень і матрицю R правих власних векторів, що задовольняють умові

$$A * R = R * D.$$

Ці вектори є внормованими таким чином, що норма кожного з них дорівнює одиниці. Наведемо приклад:

```

A =
    1     2     3
   -1     8    16
   -5    10     3
» disp(eig(A))
    1.2234
   45.2658
  -34.4893
» [R,D] = eig(A)
R =
    0.9979  -0.0798  -0.0590
    0.0492  -0.3915  -0.3530
    0.0416  -0.9167   0.9338
D =
    1.2234     0     0
     0    45.2658     0
     0     0   -34.4893

```

Сингулярне розкладання матриці робить процедура $svd(A)$. Спрощене звернення до неї дозволяє одержати сингулярні числа матриці A . Більш складне звернення виду:

$$[U, S, V] = svd(A)$$

дозволяє одержати три матриці – U , що сформована з ортонормованих власних векторів, які відповідають найбільшим власним значенням матриці $A^T A$; V - з ортонормованих власних векторів матриці $A^T A$ и S - діагональну матрицю, яка містить невід'ємні значення квадратних коренів із власних значень матриці $A^T A$ (їх називають сингулярними числами). Ці матриці задовольняють співвідношенню:

$$A = U * S * V^T.$$

Розглянемо приклад:

```
» disp(svd(A))
100.5617
15.9665
1.1896
» [U,S,V] = svd(A)
U =
-0.0207 0.1806 -0.9833
-0.0869 0.9795 0.1817
-0.9960 -0.0892 0.0045
S =
100.5617 0 0
0 15.9665 0
0 0 1.1896
V =
0.0502 -0.0221 -0.9985
-0.9978 -0.0453 -0.0491
-0.0442 0.9987 -0.0243
```

Приведення матриці до форми Гессенберга здійснюється процедурою $hess(A)$. Наприклад:

```
A =
1 2 3
-1 8 16
-5 100 3
» disp(hess(A))
1.0000 -3.3340 -1.3728
5.0990 25.5000 96.5000
0 16.5000 -14.5000
```

Більш розгорнуте звернення $[P,H] = hess(A)$ дає можливість одержати, окрім матриці H в верхній формі Гессенберга, також унітарну матрицю перетворень P , яка задовольняє умови:

$$A = P * H * P'; \quad P' * P = eye(size(A)).$$

Приклад:

```
» [P,H] = hess(A)
P =
1.0000 0 0
0 -0.1961 -0.9806
0 -0.9806 0.1961
H =
1.0000 -3.3340 -1.3728
5.0990 25.5000 96.5000
0 16.5000 -14.5000
```

Процедура *schur* (A) призначена для *приведення матриці до форми Шура*. Спростене звернення до неї призводить до одержання матриці у формі Шура.

Комплексна форма Шура - це верхня трикутна матриця із власними значеннями на діагоналі. Дійсна форма Шура зберігає на діагоналі тільки дійсні власні значення, а комплексні зображуються у виді блоків (2*2), частково займаючи нижню піддіагональ.

Звернення $[U, T] = \text{schur}(A)$ дозволяє, крім матриці T Шура, одержати також унітарну матрицю U, що задовольняє умовам:

$$A = U * H * U'; \quad U' * U = \text{eye}(\text{size}(A)).$$

Якщо початкова матриця A є дійсною, то результатом буде *дійсна форма Шура*, якщо ж комплексною, то результат видається у виді *комплексної форми Шура*.

Наведемо приклад:

```
» disp(schur(A))
  1.2234 -6.0905  -4.4758
     0   45.2658  84.0944
     0    0.0000 -34.4893
```

```
» [U, T] = hess(A)
```

```
U =
  1.0000    0    0
     0  -0.1961 -0.9806
     0  -0.9806  0.1961
```

```
T =
  1.0000  -3.3340  -1.3728
  5.0990  25.5000  96.5000
     0    16.5000 -14.5000
```

Функція $[U, T] = \text{rsf2csf}(U, T)$ перетворює дійсну квазитрикутну форму Шура в комплексну трикутну:

```
» [U, T] = rsf2csf(U, T)
U =
 -0.9934 -0.1147    0
 -0.0449  0.3892 -0.9201
 -0.1055  0.9140  0.3917
```

```
T =
  1.4091  -8.6427  10.2938
     0   45.1689 -83.3695
     0    0   -34.5780
```

Процедура $[AA, BB, Q, Z, V] = \text{qz}(A, B)$ приводить *пару матриць A і B до узагальненої форми Шура*. При цьому AA й BB є комплексними верхніми трикутними матрицями, Q, Z - матрицями приведення, а V - вектором узагальнених власних векторів такими, що

$$Q * A * Z = AA; \quad Q * B * Z = BB.$$

Узагальнені власні значення можуть бути знайдені, виходячи з такої умови:

$$A * V * \text{diag}(BB) = B * V * \text{diag}(AA).$$

Необхідність в одночасному приведенні пари матриць до форми Шура виникає в багатьох задачах лінійної алгебри - розв'язуванні матричних рівнянь Сильвестра і Ріккаті, змішаних систем диференціальних і лінійних алгебричних рівнянь.

Приклад.

Нехай задана система звичайних диференціальних рівнянь у неявній формі Коші з одним входом u і одним виходом y такого виду:

$$Q \cdot \dot{x} + R \cdot x = b \cdot u;$$

$$y = c \cdot x + d \cdot u$$

причому матриці Q, R і вектори b, c і d дорівнюють відповідно

$$Q = \begin{bmatrix} 1.0000 & 0 \\ 0.1920 & 1.0000 \end{bmatrix}$$

$$R = \begin{bmatrix} 1.1190 & -1.0000 \\ 36.4800 & 1.5380 \end{bmatrix}$$

$$b = \begin{bmatrix} 31.0960 \\ 0.1284 \end{bmatrix}$$

$$c = \begin{bmatrix} 0.6299 & 0 \end{bmatrix}$$

$$d = -0.0723$$

Необхідно обчислити значення полюсів і нулів відповідної передатної функції.

Ця задача зводиться до відшукування власних значень λ , що задовольняють матричні рівняння:

$$R \cdot r = -\lambda \cdot Q \cdot r;$$

$$\begin{bmatrix} -R & b \\ c & d \end{bmatrix} \cdot r = \lambda \cdot \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} \cdot r.$$

Розв'язання першого рівняння дозволяє **обчислити полюси передатної функції**, а другого - **нули**.

Нижче наведено сукупність операторів, яка призводить до **розрахунку полюсів**:

```
» [AA, BB] = qz(R,-Q) % Приведення матриць до форми Шура
AA =
    5.5039 + 6.7975i    24.8121 -25.3646i
    0.0000 - 0.0000i     5.5158 - 6.8036i
BB =
   -0.6457 + 0.7622i   -0.1337 + 0.1378i
                   0          -0.6471 - 0.7638i
» diag(AA) ./diag(BB) % Розрахунок полюсів
ans =
   -1.4245 - 6.0143i
   -1.4245 + 6.0143i
```

Розрахунок нулів здійснюється в такий спосіб:

```
» A = [-R      b
       c      d] % Формування
           % першої матриці
A =
          -1.1190    1.0000    0.1284
          -36.4800   -1.5380   31.0960
           0.6299     0       -0.0723
» B = [-Q      zeros(size(b)) % Формування
       zeros(size(c))  0      ] % другої матриці
B =
   -1.0000     0     0
```

```
-0.1920 -1.0000 0
0 0 0
```

```
» [AA,BB] = qz(A,B) % Приведення матриць до форми Шура
```

```
AA =
```

```
31.0963 -0.7169 -36.5109
0.0000 1.0647 0.9229
0 0.0000 0.5119
```

```
BB =
```

```
0 0.9860 -0.2574
0 0.0657 0.9964
0 0 -0.0354
```

```
» diag(AA) ./diag(BB) % Обчислення нулів
```

```
ans =
```

```
Inf
16.2009
-14.4706
```

Обчислення *власних значень матричного полінома* здійснює процедура *polyeig*. Звернення

```
[ R, d ] = polyeig(A0, A1,... , Ap )
```

дозволяє розв'язати повну проблему власних значень для матричного полінома ступеня p виду

$$(A_0 + \lambda \cdot A_1 + \dots + \lambda^p \cdot A_p) \cdot r = 0.$$

Вхідними змінними цієї процедури є $p+1$ квадратні матриці A_0, A_1, \dots, A_p порядку n . Вихідними змінними - матриця власних векторів R розміром $(n \cdot (p+1))$ і вектор d власних значень розміром $(n \cdot (p+1))$.

Функція *polyvalm* призначена для *обчислення матричного полінома* виду

$$Y(X) = p_n \cdot X^n + p_{n-1} \cdot X^{n-1} + \dots + p_2 \cdot X^2 + p_1 \cdot X + p_0$$

за заданим значенням матриці X і вектора $p = [p_n, p_{n-1}, \dots, p_0]$ коефіцієнтів полінома. Для цього достатньо звернутися до цієї процедури за схемою:

$$Y = \text{polyvalm}(p, X).$$

Приклад:

```
p = 1 8 31 80 94 20
```

```
» X
```

```
X =
```

```
1 2 3
0 -1 3
2 2 -1
```

```
» disp(polyvalm(p,X))
```

```
2196 2214 2880
882 864 1116
1332 1332 1746
```

Примітка. Слід розрізнявати процедури *polyval* і *polyvalm*. Перша обчислює значення поліному для кожного з елементів матриці аргументу, а друга при обчисленні полінома підносить до відповідного степеня всю матрицю аргументу.

Процедура *subspace*(A,U) обчислює кут між двома підпросторами, які "натягнуті на стовпчики" матриць A і B . Якщо аргументами є не матриці, а вектори A і B , обчислюється кут між цими векторами.

5.4.9. Запитання для самоперевірки

1. Як вводяться матриці у системі Matlab?
2. Які дії над матрицями передбачені у системі Matlab?
3. Які функції є у Matlab задля формування матриць визначеного виду?
4. Як сформувати матрицю: а) по заданих векторах її рядків? б) по заданих векторах її стовпців? в) по заданих векторах її діагоналей?
5. Які функції поелементного перетворення матриці є у Matlab?
6. Які функції забезпечують обробку експериментальних даних? Що потрібно зробити з експериментальними даними, щоб можна було застосувати ці функції і в якій формі виходять результати обробки?
7. Які функції лінійної алгебри передбачені у Matlab?
8. Як здійснити знаходження середнього значення результатів вимірювань і середнього квадратичного відхилення їх від цього середнього?
9. Як здійснюються у Matlab звичайні матричні операції?
10. Як розв'язати у Matlab систему лінійних алгебричних рівнянь?
11. Як обчислити у Matlab визначник матриці?
12. Як обчислити у Matlab обернену матрицю?
13. Що таке "псевдообернення" матриці і як воно здійснюється у Matlab?
14. Що таке "власні числа" і "власні вектори" матриці і як вони визначаються у Matlab?
15. Що таке "сингулярні числа" матриці і як вони визначаються у MatLAB?
16. Які функції від матриці передбачені у Matlab?
17. Як можна визначити поняття "матрична експонента", "корінь квадратний з матриці", "логарифм від матриці"? Як їх відшукати у системі MatLAB?
18. Що таке матричний поліном? Чи збігається це поняття з поняттям поліному від аргумента-матриці?
19. Як можна розрахувати значення матричного поліному?
20. Як знайти характеристичний поліном матриці?
21. Що називають СЛАР? Що означає "розв'язати СЛАР"?
22. У який спосіб найпростіше розв'язати СЛАР у системі Matlab?
23. Коли раціонально для розв'язування СЛАР використовувати LU-розкладання матриці? обернену матрицю?
24. Які функції є у Matlab, що дозволяють знайти корені СЛАР методами ітерації? Чим вони відрізняються? Як ними користуватися?

5.5. Література

1. Барановская Г. Г., Любченко И. Н. Микрокалькуляторы в курсе высшей математики: Практикум. – К.: Вища шк., 1987. – 288 с.
6. Гультияев А. К. MatLAB 5.6. Имитационное моделирование в среде Windows: Практич. пособие. –СПб.: КОРОНА-принт, 1999. – 288 с.
3. Дьяконов В. П. Справочник по применению системы РС MatLAB. – М.: Физматлит, 1993. – 113с.
4. Дьяконов В. П., Абраменкова И. В. MATLAB 5.0/5.3. Система символьной математики. – М.: Нолидж, 1999. – 640с
5. Лазарев Ю. Ф. Початки програмування у середовищі MatLAB: Навч. посібник. – К.: "Корнійчук", 1999. – 160 с.
6. Лазарев Ю. Ф. MatLAB 5.x. – К.: Издательская группа BHV, 2000. – 384 с.
7. Лазарев Ю. Ф. Моделирование процессов и систем в MATLAB. Учебный курс. – СПб.: Питер; Киев: Издат. группа BHV, 2005. – 512 с.
8. Лазарев Ю. Ф. Моделювання на ЕОМ. Навч. посібник. – К.: Корнійчук, 2007. - 290 с.
9. Мартынов Г. Г., Иванов А. П. MATLAB 5.x, вычисления, визуализация, программирование. - М.: "Кудиц-образ", 2000. - 332 с.
10. Потемкин В. Г. Система MatLAB: Справ. пособие. - М.: ДИАЛОГ-МИФИ, 1997. - 350 с.
11. Потемкин В. Г. MatLAB 5 для студентов: Справ. пособие. - М.: ДИАЛОГ-МИФИ, 1998. - 314 с.
16. Потемкин В. Г., Рудаков П. И. MatLAB 5 для студентов. - 2-е изд., испр. и дополн. - М.: ДИАЛОГ-МИФИ, 1999. - 448 с.
13. Потемкин В. Г. Система инженерных и научных расчетов MatLAB 5.x: - В 2-х т. Том 1. - М.: ДИАЛОГ-МИФИ, 1999. - 366 с.
14. Потемкин В. Г. Система инженерных и научных расчетов MatLAB 5.x: - В 2-х т. Том 6. - М.: ДИАЛОГ-МИФИ, 1999. - 304 с.
15. Сулима И. М., Гавриленко С. И., Радчик И. А., Юдицкий Я. А. Основные численные методы и их реализация на микрокалькуляторах. - К.: Вища шк., 1987. - 312 с.

6. ПРОГРАМУВАННЯ У MATLAB

Робота в режимі калькулятора в середовищі Matlab, незважаючи на достатньо значні можливості, має істотні незручності. Неможливо повторити всі попередні обчислення й дії при нових значеннях початкових даних без повторного набирання всіх попередніх операторів. Не можна повернутися назад і повторити деякі дії, або за деякою умовою перейти до виконання іншої послідовності операторів. І взагалі, якщо кількість операторів є значною, стає проблемою налагодити правильну їхню роботу через неминучі помилки при набірні команд. Тому складні, із перериваннями, складними переходами по певних умовах, із часто повторюваними однотипними діями обчислення, які, до того ж, необхідно проводити неодноразово при змінених первинних даних, потребують їхнього спеціального оформлення у виді записаних на диску файлів, тобто у виді програм. Перевага програм у тому, що, унаслідок того, що вони зафіксовані у виді записаних файлів, стає можливим багаторазове звернення до тих самих операторів і до програми в цілому. Це дозволяє спростити процес налагоджування програми, зробити процес обчислень більш наочним і прозорим, а завдячуючи цьому - різко зменшити можливість появи принципових помилок при розробці програм. Крім того, у програмах виникає можливість автоматизувати також і процес змінювання значень первісних параметрів у діалоговому режимі.

Створення програми в середовищі Matlab здійснюється за допомогою вбудованого редактора. Вікно цього вбудованого редактора виникає на екрані, якщо перед цим використано команду "M-file" із поділу *New* або обрано назву одного з існуючих M-файлів при виклику команди *Open M-file* із меню **File** командного вікна. У першому випадку вікно текстового редактора є порожнім (рис. 6.1), у другому - у ньому міститься текст викликаного M-файлу. В обох випадках вікно текстового редактора готове для введення нового тексту або коригування існуючого.

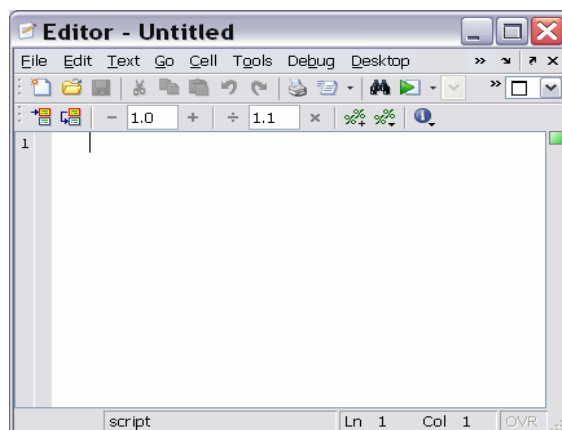


Рис. 6.1. Вікно вбудованого текстового редактора Matlab

Запис тексту програми (М-файлу) мовою Matlab має підпорядковуватися наступним правилам.

7. Зазвичай кожний оператор записується в окремому рядку тексту програми. Ознакою кінця оператора є символ (він не виникає у вікні) повернення каретки й переходу на наступний рядок, який вводиться в програму при натисканні клавіші <Enter>, тобто при переході при записі тексту програми на наступний рядок.

8. Можна розміщувати кілька операторів в одному рядку. Тоді попередній оператор у тому ж рядку має закінчуватися символом ';' або ','.

9. Можна довгий оператор записувати в декілька рядків. При цьому попередній рядок оператора має завершуватися трьома крапками ('...').

10. Якщо черговий оператор не закінчується символом ';', результат його дії при виконанні програми буде виведений у командне вікно. Щоб запобігти виведенню на екран результатів дії оператора програми, запис цього оператора в тексті програми має закінчуватися символом '!'.

11. Рядок програми, що починається із символу '%', не виконується. Цей рядок сприймається системою Matlab як *коментар*. Тому для введення коментарю в будь-яке місце тексту програми достатньо почати відповідний рядок із символу '%'.
12. Рядки коментаря, які передують першому виконуваному (тобто такому, що не є коментарем) оператору програми, сприймаються системою Matlab як опис програми. Саме ці рядки виводяться в командне вікно, якщо в ньому набрано команду

help <ім'я файла>

10. У програмах мовою Matlab *відсутній символ закінчення тексту програми*.

11. У мові Matlab *змінні не описуються і не оголошуються*. Будь-яке нове ім'я, що зустрічається в тексті програми при її виконанні, сприймається системою Matlab як ім'я матриці. Розмір цієї матриці встановлюється при введенні значень її елементів або визначається діями по встановленню значень її елементів, описаними у попередньому операторі або процедурі. Ця особливість робить мову Matlab дуже простою у вжитку і привабливою. У мові Matlab неможливо використання вхідної матриці або змінної, у якій попередньо не введені або обчислені значення її елементів (а значить - і визначені розміри цієї матриці). У протилежному випадку при виконанні програми Matlab з'явиться повідомлення про помилку - "Змінна не визначена".

12. Імена змінних можуть містити лише букви латинського алфавіту або цифри і мають починатися з букви. Загальна кількість символів в імені може сягати 30. В іменах змінних можуть використовуватися як великі, так і малі букви. Особливістю мови Matlab є те, що *великі й малі букви в іменах розрізняються системою*. Наприклад, символи "a" і "A" можуть використовуватися в одній програмі для позначення різних величин.

Програми мовою MatLAB мають два різновиди - так званий *Script-файл* (*файл-сценарій*, або *керуюча програма*) і *файл-функція* (*процедура*). Обидва різновиди матимуть розширення *.m* при записі тексту файлу на диск, тобто їх не можна розрізнити по виду ймення.

За допомогою Script-файлів оформлюють основні програми, що керують із початку до кінця організацією усього обчислювального процесу, і окремі частини основних програм (вони можуть бути записані у виді окремих Script-файлів). Як файл-функції оформляють окремі процедури й функції (тобто такі частини програми, які розраховані на неодноразове використання Script-файлами або другими процедурами при змінених значеннях вхідних параметрів і не можуть бути виконані без попереднього завдання значень деяких змінних, які називають *вхідними*).

Головною зовнішньою відмінністю цих двох видів файлів є те, що файл-функції мають перший рядок виду

function <ПКВ> = <ім'я процедури >(<ПВВ>),

де позначено ПКВ - Перелік Кінцевих Величин, ПВВ - Перелік Вхідних Величин.

Script-файли такого рядка не мають.

Принципова ж відмінність полягає в зовсім різному сприйнятті системою імен змінних у цих двох видах файлів. У файл-функціях усі імена змінних усередині файлу, а також імена змінних, зазначені в заголовку (ПКВ і ПВВ), сприймаються як *локальні*, тобто усі значення цих змінних після завершення роботи процедури зникають, і область оперативної пам'яті, що була відведена під запис значень цих змінних, звільняється для запису в її значень інших змінних.

У Script-файлах усі використовувані змінні утворюють так званий "*робочий простір*" (*Work Space*). Значення й зміст їх зберігаються не тільки протягом часу роботи програми, але й протягом усього сеансу роботи із системою, а, виходить, і при переході від виконання одного Script-файлу до іншого. Інакше кажучи, робочий простір є єдиним для всіх Script-файлів, що викликаються в поточному сеансі роботи із системою. Саме завдячуючи цьому будь-який довгий Script-файл можна розбити на декілька фрагментів, оформити кожний з них у виді окремого Script-файлу, а в головному Script-файлі замість відповідного фрагменту записати оператор виклику Script-файлу, який подає цей фрагмент. Цим забезпечується компактне й наочне подання навіть досить складної програми.

За винятком зазначених відмінностей, файл-функції і Script-файли оформляються однаково. Головні правила написання текстів M-файлів були наведені раніше (п. 1. 6. 8).

6.1. Оператори керування обчислювальним процесом

Загалом кажучи, оператори керування необхідні, головним чином, для організації обчислювального процесу, який записується у виді деякого тексту програми на мові програмування високого рівня. При цьому до операторів керування обчислювальним процесом звичайно відносять оператори безумовного переходу, умовних переходів (розгалуження обчислювального процесу) і оператори організації циклічних процесів. Проте система MatLAB побудована таким чином, що ці оператори можуть бути використані і при роботі MatLAB у режимі калькулятора.

У мові MatLAB відсутній оператор безумовного переходу і, відповідно до цього, немає поняття мітки. Ця обставина є хвибою мови MatLAB і утруднює організацію повернення обчислювального процесу до будь-якого попереднього або наступного оператора програми.

Усі оператори циклу й умовного переходу побудовані в MatLAB у виді складного оператора, що починається з одного зі службових слів *if*, *while*, *switch* або *for* і закінчується службовим словом *end*. Оператори усередині між цими словами сприймаються системою як частини одного складного оператора. Тому натискання клавіші <Enter> для переходу до наступного рядка не призводить у цьому випадку до виконання цих операторів. Виконання операторів починається лише тоді, коли введена "завершувальна дужка" складного оператора у виді *end*, а потім натиснуто клавішу <Enter>. Якщо декілька складних операторів такого типу вкладені один в інший, обчислення починаються лише тоді, коли записаний кінець *end* найбільш охоплюючого (зовнішнього) складного оператора. З цього випливає можливість здійснення навіть у режимі калькулятора досить складних і об'ємних (що складаються з багатьох рядків і операторів) обчислень, якщо вони охоплені складним оператором.

6.1.1. Оператор умовного переходу

Конструкція оператора переходу за умовою в загальному виді є такою:

```
if <умова>
    <оператори1>
else
    <оператори2>
end
```

Працює оператор у такий спосіб. Спочатку перевіряється, чи виконується зазначена умова. Якщо її виконано, програма виконує сукупність операторів, що записана в поділі <оператори1>. Якщо умову не виконано, виконується послідовність операторів поділу <оператори2>.

Скорочена форма умовного оператора має вид:

```
if <умова>
    <оператори>
end
```

Дія оператора в цьому випадку аналогічно, за винятком того, що при невиконанні заданої умови виконується оператор, наступний за оператором *end*.

Легко помітити хиби цього оператора, що впливають із відсутності оператора безумовного переходу: усі частини програми, що виконуються в залежності від умови, повинні розміщатися усередині операторних дужок *if* і *end*.

Як умова використовується вирази типу:

<ім'я змінної1> <операція порівнювання> <ім'я змінної2>

Операції порівнювання в мові *MatLAB* можуть бути такими:

<	- менше;
>	- більше;
<=	- менше або дорівнює;
>=	- більше або дорівнює;
= =	- дорівнює;
~ =	- не дорівнює.

Умова може бути складеною, тобто складатися з кількох простих умов, що об'єднуються знаками логічних операцій. Знаками логічних операцій у мові *MatLAB* є:

& – логічна операція “АБО” (“OR”);

~ - логічна операція “НІ” (“NOT”).

Логічна операція “Виняткове АБО” може бути реалізована за допомогою функції *xor*(A,B), де A і B - деякі умови.

Є припустимою ще одна конструкція оператора умовного переходу:

```

if <умова1>
    <оператори1>
elseif <умова2>
    <оператори2>
elseif <умова3>
    <оператори3>
...
else
    <оператори>
end

```

Оператор *elseif* виконується тоді, коли <умова1> не виконана. При цьому спочатку перевіряється <умова2>. Якщо її виконано, виконуються <оператори2>, якщо ж ні, <оператори2> ігноруються, і відбувається перехід до наступного оператора *elseif*, тобто до перевірки виконання <умови3>. Аналогічно, при виконанні її виконуються <оператори3>, у протилежному випадку відбувається перехід до наступного оператора *elseif*. Якщо жодну з умов в операторах *elseif* не виконано, виконуються <оператори>, що містяться за оператором *else*. У такий спосіб може бути забезпечене розгалуження програми по кількох напрямках.

6.1.2. Оператор переключення

Оператор переключення має таку структуру:

```

switch <вираз, скаляр або рядок символів>
case <значення1>

```

```

        <оператори1>
    case <значення2>
        <оператори2>
    ...
    otherwise
        <оператори>
end

```

Він здійснює розгалужування обчислень у залежності від значень деякої змінної або виразу, порівнюючи значення, отримане в результаті обчислення виразу в рядку *switch*, із значеннями, зазначеними в рядках із словом *case*. Відповідна група операторів *case* виконується, якщо значення виразу збігається зі значенням, зазначеним у відповідному рядку *case*. Якщо значення виразу не збігається з жодним із значень у групах *case*, виконуються оператори, що наслідують *otherwise*.

6.1.3. Оператори циклу

У мові MatLAB є два різновиди операторів циклу – умовний і арифметичний.

Оператор циклу з передумовою має вид:

```

while <умова>
    <оператори>
end

```

Оператори усередині циклу виконуються лише в тому випадку, якщо є виконаною умова, записана після слова *while*. При цьому серед операторів усередині циклу обов'язково повинні бути такі, що змінюють значення однієї зі змінних, зазначених в умові циклу.

Наведемо приклад обчислення значення синуса при 21 значенні аргументу від 0.2 до 4 із кроком 0.2:

```

» i = 1;
» while i <= 20
    x = i/5;
    si = sin(x);
    disp([x,si])
    i = i+1;
end
0.2000    0.1987
0.4000    0.3894
0.6000    0.5646
0.8000    0.7174
1.0000    0.8415
1.2000    0.9320
1.4000    0.9854
1.6000    0.9996
1.8000    0.9738
2.0000    0.9093
2.2000    0.8085
2.4000    0.6755
2.6000    0.5155
2.8000    0.3350
3.0000    0.1411
3.2000   -0.0584
3.4000   -0.2555
3.6000   -0.4425

```



```
3.8000    -0.6119
4.0000    -0.7568
```

Примітка. Зверніть увагу на те, якими засобами в зазначеному прикладі забезпечено виведення на екран значень кількох змінних одним рядком.

Для цього використовується оператор *disp*, який раніше застосовувався. Але, відповідно до правил застосування цього оператора, у ньому має бути тільки один аргумент (текст, змінна або матриця). Щоб обминути цю перешкоду, потрібно декілька числових змінних об'єднати в єдиний об'єкт - вектор-рядок, а це легко виконується за допомогою звичайної операції формування вектора-рядка з окремих елементів

```
[ x1, x2, ... , xN].
```

Таким чином, за допомогою оператора виду:

```
disp([x1, x2, ... , xN])
```

можна забезпечити виведення результатів обчислень у виді таблиці даних.

Арифметичний оператор циклу має вид:

```
for <ім'я> = <ПЗ> : <К> : <КЗ>
    <оператори>
```

```
end,
```

де <ім'я> – ім'я керуючої змінної циклу – "лічильника" циклу; <ПЗ> – задане початкове значення цієї змінної; <К> – значення кроку, із яким вона має змінюватися; <КЗ> – кінцеве значення змінної циклу. У цьому випадку <оператори> усередині циклу виконуються кілька разів (кожного разу при новому значенні керуючої змінної) доти, поки значення керуючої змінної не вийде за межі інтервалу між <ПЗ> і <КЗ>. Якщо параметр <К> не зазначений, за замовчуванням його значення приймається рівним одиниці.

Щоб достроково вийти з циклу (наприклад, при виконанні деякої умови) застосовують оператор *break*. Якщо програма стикається з цим оператором, виконання циклу достроково припиняється, і починає виконуватися оператор, наступний за словом *end* циклу.

Для прикладу використаємо попереднє завдання:

```
» a = [' i ' ; ' x ' ; ' sin(x) '];
» for i = 1:20
    x = i/5;
    si = sin(x);
    if i==1
        disp(a)
    end
    disp([i,x,si])
end
```

В результаті виходить

```
 i      x      sin(x)
1.0000 0.2000 0.1987
6.0000 0.4000 0.3894
3.0000 0.6000 0.5646
4.0000 0.8000 0.7174
5.0000 1.0000 0.8415
6.0000 1.2000 0.9320
7.0000 1.4000 0.9854
8.0000 1.6000 0.9996
9.0000 1.8000 0.9738
10.0000 6.0000 0.9093
11.0000 6.2000 0.8085
```

16.0000	6.4000	0.6755
13.0000	6.6000	0.5155
14.0000	6.8000	0.3350
15.0000	3.0000	0.1411
16.0000	3.2000	-0.0584
17.0000	3.4000	-0.2555
18.0000	3.6000	-0.4425
19.0000	3.8000	-0.6119
20.0000	4.0000	-0.7568

У такий спосіб можна забезпечити виведення інформації у вигляді таблиць.

6.2. Створення найпростіших файл-функцій (процедур)

6.2.1. Загальні вимоги до побудови

Як було зазначено раніше, файл-функція (процедура) має починатися з рядка заголовка виду

function [<ПКВ>] = <ім'я процедури>(<ПВВ>).

Якщо перелік кінцевих (вихідних) величин (ПКВ) містить тільки один об'єкт (у загальному випадку – матрицю), то файл-функція є звичайною функцією (однієї або кількох змінних). Фактично навіть у цьому найпростішому випадку файл-функція є вже процедурою у звичайному розумінні інших мов програмування, якщо вихідна величина є вектором або матрицею. Перший рядок у цьому випадку має вид:

function <ім'я змінної> = <ім'я процедури>(<ПВВ>).

Якщо ж у результаті виконання файл-функції мають бути визначені (обчислені) кілька об'єктів (типу матриць), така файл-функція є вже більш складним об'єктом, який у програмуванні зазвичай називається або процедурою (у мові Паскаль), або підпрограмою. Загальний вид першого рядка в цьому випадку стає таким:

function [y1, y2, ... , yN] = <ім'я процедури>(<ПВВ>),

тобто перелік вихідних величин y1, y2, ... , yN має бути поданий як вектор-рядок з елементами y1, y2, ... , yN (кожний з яких може бути матрицею).

У найпростішому випадку функції однієї змінної заголовков набуває вид:

function y = **func**(x),

де **func** – ім'я функції (M-файлу).

Як приклад розглянемо складання M-файлу для функції

$$y = f_1(x) = d^3 \cdot ctg(x) \cdot \sqrt{\sin^4(x) - \cos^4(x)} \quad .$$

Для цього треба активізувати поділ *File* головного меню командного вікна MatLAB, обрати в підменю, що виникло на екрані, розділ *New*, а потім – команду *M-file*. На екрані виникне вікно текстового редактора. У ньому потрібно набрати такий текст:

```
function y = F1(x,d)
% Процедура, яка обчислює значення функції
% y = (d^3)*ctg(x)*sqrt(sin(x)^4-cos(x)^4).
% Звернення y = F1(x,d).
y = (d^3)*cot(x). *sqrt(sin(x). ^4-cos(x). ^4);
```

Після цього необхідно зберегти цей текст у файлі під ім'ям **F1.m**. Необхідний М-файл створений. Тепер можна користуватися цією функцією при розрахунках. Так, якщо ввести команду

```
» y = F1(1, 0.1)
```

то одержимо результат

```
y = 4. 1421e-004.
```

Варто зауважити, що аналогічно можна одержати одразу вектор усіх значень зазначеної функції при різних значеннях аргументу, якщо останні зібрати в деякий вектор. Так, якщо сформулювати вектор

```
» zet= 0:0. 3:1. 8;
```

і звернутися до тієї ж процедури

```
» my = F1(zet,1),
```

те одержимо:

```
Warning: Divide by zero
my =
Columns 1 through 4
    Na + Inf  0 + 6. 9369i    0 + 0. 8799i    0. 3783
Columns 5 through 7
    0. 3339    0. 0706    -0. 2209
```

Примітки.

1. Можливість використання сформованої процедури як для окремих чисел, так і для векторів і матриць обумовлена застосуванням у запису відповідного М-файлу замість звичайних знаків арифметичних дій їхніх аналогів із попередньою крапкою.

6. Щоб уникнути виведення на екран небажаних проміжних результатів, необхідно в тексті процедури усі обчислювальні оператори завершувати символом " ; ".

3. Як показують наведені приклади, імена змінних, зазначені в заголовку файл-функції можуть бути будь-якими (збігатися чи ні з іменами, використовуваними при зверненні до цієї файл-функції), тобто мають формальний характер. Важливіше за все лише те, щоб структура звернення цілком відповідала структурі заголовка в запису тексту М-файлу і щоб змінні в цьому зверненні мали той тип і розмір, що й в заголовку М-файлу.

Щоб одержати інформацію про створену процедуру, достатньо набрати в командному вікні команду:

```
» help f1,
```

і в командному вікні з'явиться

```
Процедура, яка обчислює значення функції
y = (d^3)*ctg(x)*sqrt(sin(x)^4-cos(x)^4).
Звернення y = F1(x,d).
```

Інший приклад. Побудуємо графік двох функцій:

```
y1 = 200 sin(x)/x;          y2 = x6.
```

Для цього створимо М-файл, що обчислює значення цих функцій:

```
function y = myfun(x)
% Обчислення двох функцій
% y(1) = 200 sin(x)/x,          y(2) = x6.
```

```
y(:,1) = 200*sin(x) ./ x;
y(:,2) = x . ^ 2;
```

Тепер побудуємо графіки цих функцій:

```
» plot('myfun', [-20 20], 50, 2), grid
» set(gcf,'color','white'); title('Графік функції "MYFUN"')
```

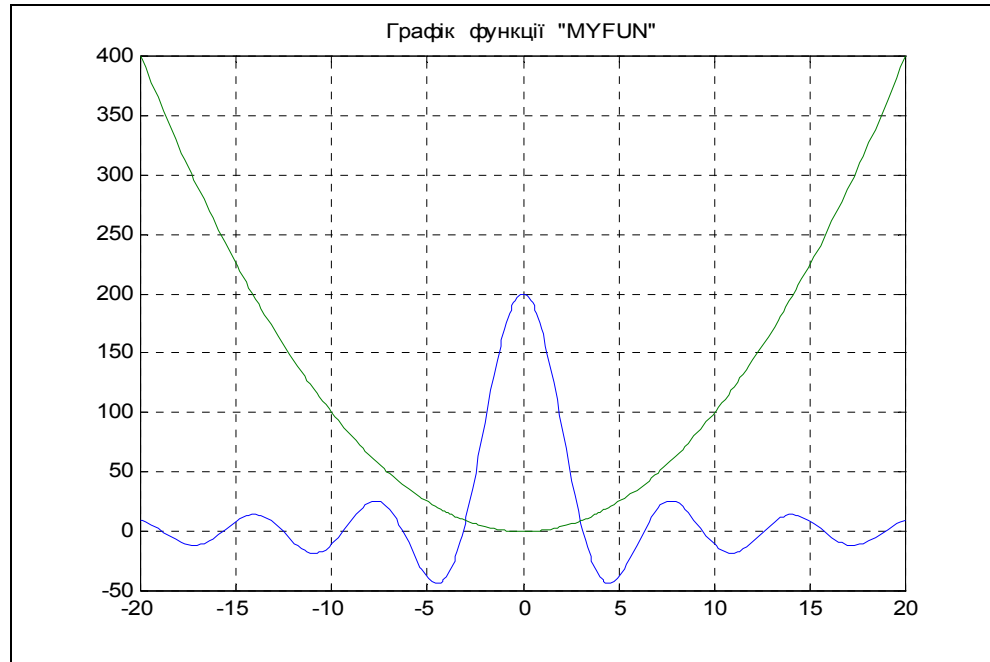


Рис. 6.6. Побудова графіків власних функцій

Результат зображений на рис. 6.6.

Третій приклад – створення файл-функції, що обчислює значення функції $y(t) = k_1 + k_2 \cdot t + k_3 \cdot \sin(k_4 \cdot t + k_5)$.

У цьому випадку зручно об'єднати сукупність коефіцієнтів k у єдиний вектор K :

```
K = [k1 k2 k3 k4 k5]
```

і створити такий М-файл:

```
function y = dvob(x, K)
% Обчислення функції
% y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5)),
% де K - вектор із п'яти елементів
%
% Використовується для визначення поточних значень
% параметрів руху рухомого об'єкта
y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5));
```

Тоді розрахунок, наприклад, 11-ти значень цієї функції можна здійснити

так

```
» K = ones(1,5);
» t = 0:1:10;
» fi = dvob(t, K)
```

```
fi = 1. 8415 6. 9093 3. 1411 3. 2432 4. 0411 5. 7206 7. 6570 8. 9894 9. 4560 10.
0000
```

6.2.2. Типове оформлення процедури-функції

Рекомендується оформляти М-файл процедури-функції за такою схемою:

```
function [<Вихід>] = <ім'я функції>(<Вхід>)
% <Стисле пояснення, що робить процедура>
% Вхідні змінні
% <Детальне пояснення про зміст, типі і розміри
% кожній із змінних, перерахованих у переліку <Вхід>
% Вихідні змінні
% <Детальне пояснення про зміст, типі і розміри
% кожної із змінних переліку <Вихід>
% і величини, що використані у процедурі як глобальні>
% Використання інших функцій і процедур
% <Розділ заповнюється, якщо процедура містить звернення
% до інших процедур, крім умонтованих>
% < Порожній рядок >
% Автор : <Вказується автор процедури, дата створення кінцевого варіанта
% процедури й організація, у якій створена програма>
< Текст процедури >
```

Тут позначено: <Вихід> - перелік вихідних змінних процедури, <Вхід> - перелік вхідних змінних, розділених комами.

Примітка. При використанні команди *help* <ім'я процедури> у командне вікно виводяться рядки коментарю до першого порожнього рядка.

6.2.3. Запитання для самоперевірки

1. Для чого створюються програми в середовищі MatLAB?
2. Чим відрізняються файли-функції від Script-файлів? Яка сфера застосування кожного з цих видів файлів?
3. Як створити М-файл процедури або функції?
4. Які основні правила написання текстів М-файлів у мові MatLAB?

6.3. Створення Script-файлів

6.3.1. Основні особливості Script-файлів

Перелікуємо головні особливості Script-файлів:

- Script-файли є блоками операторів і команд, що незалежно (самостійно) виконуються;

- усі змінні, які використовуються в них, утворюють так названий *робочий простір*, що є загальним для всіх Script-файлів, що виконуються; із цього випливає, що при виконанні кількох Script-файлів імена змінних у них мають бути узгоджені, щоб одне ім'я означало в кожному з них той самий об'єкт обчислень;

- в них немає заголовка, тобто першого рядка визначеного виду й призначення;

■ звернення до них не потребує вказівки ніяких імен змінних: усі змінні формуються в результаті виконання програми або сформовані раніше й існують у робочому просторі.

Необхідно відзначити, що робочий простір Script-файлів є недосяжним для файл-функцій, які використовуються в ньому. У файл-функціях неможливо використовувати значення, що одержують змінні в Script-файлі, обходячи заголовок файл-функції (через те, що всі змінні файл-функції є *локальними*). Єдиною можливістю зробити так, щоб усередині файл-функції деяка змінна робочого простору могла бути застосованою, зберігши своє значення й ім'я, є спеціальне оголошення цієї змінної у Script-файлі як глобальної за допомогою службового слова **global**. Крім того, аналогічний запис має бути розташований й у тексті М-файлу тієї файл-функції, яка буде використовувати значення відповідної змінної Script-файлу.

Наприклад, можна перебудувати файл-функції першого й третього прикладів із попереднього розділу, уводячи коефіцієнти відповідних функцій як глобальні змінні:

```
function y = dvob1(x)
% Обчислення функції
% y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5)),
% де ДО - глобальний вектор із п'ятьох елементів
%     Застосовується для визначення поточних значень
%     параметрів руху рухомого об'єкта
global K
y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5));
```

Щоб використати нову файл-функцію **dvob1** у Script-файлі, в останньому перед зверненням до цієї функції має бути записаний рядок

```
global K
```

і визначений вектор-рядок **K** із п'ятьох елементів (задані їхні значення).

Якщо в одному рядку оголошуються *кілька змінних* як глобальні, вони мають відокремлюватися пропусками (не комами!).

6.3.2. Введення й виведення інформації у діалоговому режимі

Для забезпечення взаємодії з користувачем у процесі виконання М-файлу в системі MatLAB призначені такі команди:

disp, sprintf, input, menu, keyboard, pause.

Команда **disp** здійснює виведення значень зазначеної змінної або зазначеного тексту в командне вікно. Звернення до неї має вид:

disp (<змінна або текст в апострофах>).

Особливістю цієї команди є те, що аргумент у неї може бути тільки один. Тому неможливо без спеціальних заходів здійснити виведення кількох змінних і, особливо, об'єднання тексту з чисельними значеннями деяких змінних, що часто є необхідним для зручного подання інформації.

Для усунення цієї хиби використовують декілька засобів.

Щоб вивести значення кількох змінних у єдиний рядок (це необхідно при створенні таблиць даних), потрібно створити єдиний об'єкт, що містив би всі ці значення. Це можна зробити, об'єднавши відповідні змінні у вектор, користуючись операцією створення вектора-рядка:

```
x = [x1 x2 ... xN].
```

Тоді виведення значень кількох перемінних в один рядок буде мати вид:

```
disp ([x1 x2 ... xN]).
```

Наведемо приклад:

```
» x1=1.24;    x2=-3.45;    x3=5.76;    x4=-8.07;
```

```
» disp([x1 x2 x3 x4])
```

```
1.2400 -3.4500 5.7600 -8.0700.
```

Аналогічно можна об'єднувати декілька текстових змінних, наприклад:

```
» x1='psi ';  x2='fi ';    x3='teta ';  x4=' w1 ';
```

```
» disp([x1 x2 x3 x4])
```

```
psi fi teta w1
```

Значно складніше об'єднати в один рядок текст і значення змінних, що також часто є необхідним. Труднощі виникають тому, що текстові і числові змінні не можуть бути об'єднані, бо є даними різних типів. Одним із шляхів подолання цієї перешкоди є переклад числового значення змінної у символічну (текстову) форму. Це можливо, якщо скористатися функцією *num2str*, яка здійснює таке перетворення. Запис

```
y = num2str(x)
```

перетворить числове значення змінної "x" на її текстове подання, наприклад:

```
» x = -9.30876e-15
```

```
x = -9.3088e-015
```

```
» y = num2str(x)
```

```
y = -9.309e-015
```

При цьому форма символічного подання визначається встановленим режимом виведення чисел на екран (Numeric Format).

Якщо T – текстова змінна, або деякий текст, а X – числова змінна, то виведення їх в одному рядку можна забезпечити зверненням

```
disp ([T num2str(X)]).
```

Розглянемо приклад:

```
x = -9.3088e-015
```

```
» T = 'Значення параметра дорівнює ';
```

```
» disp([T x])
```

```
Значення параметра дорівнює
```

```
» disp([T num2str(x)])
```

```
Значення параметра дорівнює -9.309e-015
```

Як впливає з цього приклада, "механічне" об'єднання текстової й числової змінних не приводить до бажаного результату, а використання функції *num2str* – приводить.

Іншим засобом досягнення того ж результату є використання функції *sprintf*. Звернення до неї має вид:

```
Y = sprintf ('<текст1> %g <текст2>', X).
```

У результаті утворюється текстовий рядок Y, що складається з тексту, зазначеного в <текст1>, і значення числової змінної X в відповідності з форматом

`%g`, причому текст із фрагмента `<текст2>` розташовується після значення змінної `X`. Цю функцію можна використовувати в команді `disp` у виді:

`disp(sprintf(' <текст> %g', X)).`

Приклад:

» `disp(sprintf('Параметр1 = %g ',x))`

Параметр1 = -9.30876e-015

Уведення інформації із клавіатури в діалоговому режимі можна здійснити за допомогою функції `input`. Звернення до неї виду:

`x = input(' <запрошення>')`

приводить до таких дій комп'ютера. Виконання операторів програми припиняється. Комп'ютер переходить у режим очікування введення інформації із клавіатури. Після закінчення введення із клавіатури (яке визначається натисканням клавіші `<Enter>`) уведена інформація запам'ятовується в програмі під ім'ям `"x"`, і виконання програми продовжується.

Зручним інструментом вибору деякої з альтернатив майбутніх обчислювальних дій є функція `menu` MatLAB, яка утворює вікно меню користувача. Функція `menu` має такий формат звернення:

`k=menu('Заголовок меню','Альтернатива1','Альтернатива2','Альтернатива n').`

Таке звернення приводить до появи на екрані дисплея меню, зображеного на рис. 6.3.

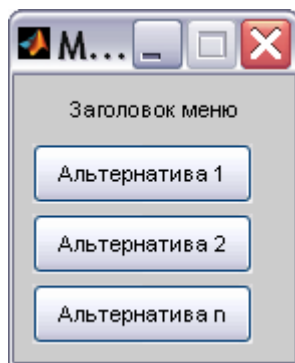


Рис. 6.3. Вид меню

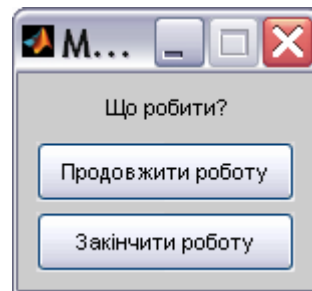


Рис. 6.4. Меню продовження роботи

Виконання програми тимчасово припиняється. Система очікує відповіді у виді натискання лівої клавіші миші після встановлення курсору на одну із зображених "кнопок" меню з альтернативами. Після правильної відповіді вихідному параметрові `"k"` присвоюється значення номера обраної альтернативи (1, 2 або 3). У загальному випадку число альтернатив може бути до 36.

Тепер, у залежності від отриманого значення цього параметра, можна побудувати процес розгалуження обчислень, наприклад, вибору параметра, значення якого потрібно змінити.

Команда `pause` тимчасово припиняє виконання програми доти, поки користувач не натисне будь-яку клавішу клавіатури. Якщо після назви команди зазначити в скобках деяке додатне ціле число `n`, то затримка виконання програми буде здійснена протягом `n` секунд.

Якщо в тексті М-файлу зустрічається команда `keyboard`, то при виконанні програми виконання М-файлу припиняється, і керування передається клавіатурі

дисплея. Цей спеціальний режим роботи супроводжується появою у командному вікні MatLAB нового виду запрошення до дій

k>>.

У цьому режимі користувач може здійснити будь-які дії, перевірити або змінити дані. При цьому йому доступні всі команди й процедури системи MatLAB. Для завершення роботи в цьому режимі необхідно набрати команду *return*. Тоді система продовжить роботу програми з оператора, що є наступним після оператора *keyboard*.

6.3.3. Організація повторювання дій

Однією з основних задач створення самостійної програми є забезпечення повертання до початку програми з метою продовження її виконання при нових значеннях первісних даних.

Нехай основні оператори створеної програми розташовані в Script-файлі з ім'ям *ScrFil_yadro.m*. Тоді схема забезпечення повертання до початку виконання цього Script-файлу може бути, приміром, такою:

```
flag =0;
while flag == 0
    ScrFil_yadro
    kon=0;
    kon=input("Закінчити роботу-<3>, продовжити - <Enter>");
    if kon==3,
        flag=3;
    end
end
```

У цьому випадку Script-файл *ScrFil_yadro* буде повторно виконуватися доти, поки на питання 'Закінчити роботу-<3>, продовжити - <Enter>' не буде введено із клавіатури відповідь "3". Якщо ж відповідь буде саме такою, цикл закінчиться й будуть виконуватися наступні за цим циклом оператори. Природно, що змінна *flag* не повинна змінювати своє значення в Script-файлі *ScrFil_yadro*.

Можна також із тією ж метою використовувати механізм створення меню. У цьому випадку програму можна подати, приміром, так:

```
k=1;
while k==1
    ScrFile_Yadro
    k = menu(' Що робити ? ',' Продовжити роботу ', ' Закінчити роботу ');
end
```

Тоді, після першого виконання Script-файлу *ScrFil_Yadro* на екрані виникне вікно меню, зображене на рис. 6.4, і, при натисканні кнопки першої альтернативи значення *k* залишиться рівним одиниці, цикл повториться, а при натисканні другої кнопки *k* стане рівним 2, цикл закінчиться і програма перейде до закінчення роботи.

6.3.4. Організація змінювання даних у діалоговому режимі

Повторення дій, що містяться в ядрі *ScrFil_Yadro*, має сенс тільки у випадку, коли на початку цього ядра забезпечене виконання дій по змінюванню деяких із первинних величин. MatLAB містить ряд зручних засобів, які дозволяють здійснювати змінювання даних у діалоговому режимі з використанням стандартних меню-вікон користувача.

Організацію діалогового змінювання даних розглянемо на прикладі деяких 5 параметрів, які назвемо *Параметр1*, *Параметр2*, ..., *Параметр5*. Нехай їхні позначення як змінних у програмі такі: x_1 , x_2 , ..., x_5 . Тоді меню вибору параметра для змінювання його значення повинно містити 6 альтернатив: 5 із них призначені для вибору одного із зазначених параметрів, а остання альтернатива має надати можливість виходу з меню, коли значення всіх параметрів встановлені.

Тому варіант оформлення такого меню може бути, наприклад, наступним:

```
k = menu(' Що змінити ? ', ' Параметр1 ', ' Параметр2 ', ...
        ' Параметр3 ', ' Параметр4 ', ' Параметр5 ', ' Нічого не змінювати '),
```

що приведе до появи вікна, поданого на рис. 6.5.

Легко помітити хиби такого оформлення вікна меню. Щоб прийняти рішення, значення якого саме параметра варто змінити і як, користувач має мати перед очима не тільки перелік параметрів, які можна змінити, але й поточні значення цих параметрів. Тому на кожній кнопці меню має міститися також інформація про поточне значення відповідного параметра. Це можна зробити, використовуючи раніше згадану функцію *sprintf*, наприклад, у такий спосіб:

```
x1=-1.89;    x2=239.78;    x3=-6.56e-3;    x4=7.28e-15;    x5=1.023e-32;
k = menu(' Що змінювати ? ', ...
        sprintf(' Параметр1  x1 = %g', x1),...
        sprintf(' Параметр2  x2 = %g', x2),...
        sprintf(' Параметр3  x3 = %g', x3),...
        sprintf(' Параметр4  x4 = %g', x4),...
        sprintf(' Параметр5  x5 = %g', x5),...
        ' Нічого не змінювати ')
```

Результат наведено на рис. 6.6.

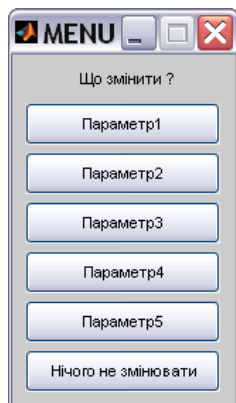


Рис. 6.5. Варіант 1 меню

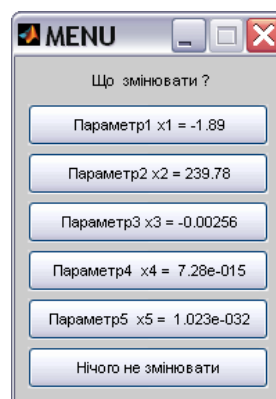


Рис. 6.6. Варіант 2 меню

Меню дозволяє обрати параметр, який потрібно змінити, проте не забезпечує самого змінювання обраного параметра. Це змінювання має бути здійснено за допомогою введення нового значення з клавіатури, приміром, так:

```
x = input( [sprintf('Поточне значення x = %g',x), 'Нове значення x = ']).
```

Наприклад, якщо ввести команди

```
» x = 3. 02e-2;
```

```
» x=input( [sprintf('Поточне значення x = %g',x), ' Нове значення x = ']
```

то в командному вікні виникне напис:

```
Поточне значення x = 0. 0302 Нове значення x =
```

Виконання програми припиниться. ПК буде очікувати закінчення введення інформації із клавіатури. Якщо тепер набрати на клавіатурі "0.073" і натиснути клавішу <Enter>, то в командному вікні з'явиться напис:

```
Поточне значення x = 0. 0302 Нове значення x = 0. 073
```

```
x = 0. 0730
```

Щоб запобігти повторному виведенню на екран уведеного значення, необхідно рядок із функцією **input** завершити символом " ; ".

Тепер слід організувати вибір різних видів такого типу операторів відповідно до окремих обраних параметрів. Для цього можна використовувати оператор умовного переходу, наприклад, так:

```
if k==1,
```

```
    x1 = input( [sprintf('Поточне значення x1 = %g', x1) ...
                ' Нове значення x1= ']);
```

```
elseif k==2,
```

```
    x2 = input( [sprintf('Поточне значення x2 = %g', x2) ...
                ' Нове значення x2= ']);
```

```
elseif k==3,
```

```
    x3 = input( [sprintf('Поточне значення x3 = %g', x3) ...
                ' Нове значення x3= ']);
```

```
elseif k==4
```

```
    x4 = input( [sprintf('Поточне значення x4 = %g', x4) ...
                ' Нове значення x4= ']);
```

```
elseif k==5
```

```
    x5 = input( [sprintf('Поточне значення x5 = %g', x5) ...
                ' Нове значення x5= ']);
```

```
end
```

Щоб можна було проконтролювати правильність введення нових значень, забезпечити можливість їхнього коригування і послідовного змінювання всіх бажаних параметрів, потрібно, щоб після введення нового значення будь-якого параметра на екрані знову виникало те саме вікно меню, але вже зі скоригованими значеннями. При цьому кінець роботи з меню має настати тільки за умови обрання останньої альтернативи меню "Нічого не змінювати", що відповідає значенню k, рівному 6.

Тому попередні оператори варто замкнути в цикл:

```
k=1;
```

```
while k<6
```

```
k = menu(' Що змінити ? ', ...
```

```
sprintf(' Параметр1 x1 = %g', x1),...
```

```
sprintf(' Параметр2 x2 = %g', x2),...
```

```
sprintf(' Параметр3 x3 = %g', x3),...
```

```
sprintf(' Параметр4 x4 = %g', x4),...
```

```
sprintf(' Параметр5 x5 = %g', x5), ' Нічого не змінювати ');
```

```
if k==1,
```

```
    x1 = input( [sprintf('Поточне значення x1 = %g', x1) ...
```

```

elseif k==2,      ' Нове значення x1= ');
                  x2 = input( [sprintf('Поточне значення x2 = %g', x2) ...
                              ' Нове значення x2= ']);
elseif k==3,      x3 = input( [sprintf('Поточне значення x3 = %g', x3) ...
                              ' Нове значення x3= ']);
elseif k==4,      x4 = input( [sprintf('Поточне значення x4 = %g', x4) ...
                              ' Нове значення x4= ']);
elseif k==5,      x5 = input( [sprintf('Поточне значення x5 = %g', x5) ...
                              ' Нове значення x5= ']);
end
end

```

У такий спосіб організується досить зручне діалогове змінювання значень параметрів.

Якщо вхідних параметрів, значення яких потрібно змінювати, досить багато, варто об'єднати такі параметри в компактні групи (бажано – за якоюсь загальною властивістю, яка відрізняє визначену групу від інших) і аналогічним чином забезпечити діалогове змінювання, використовуючи окреме меню для кожної групи. Очевидно, при цьому необхідно попередньо забезпечити вибір однієї з цих груп параметрів через додаткове вікно меню.

6.3.5. Типова структура й оформлення Script-файлу

При написанні тексту програми у виді Script-файлу необхідно брати до уваги наступне.

1. Зручніше оформляти весь процес діалогового змінювання значень параметрів у виді окремого Script-файлу, приміром, за ім'ям *ScrFil_Menu*, де під скороченням "ScrFil" розуміється ім'я основного (збірного) Script-файлу.

2. Через те що вже на самому початку роботи із програмою в меню вибору змінюваного параметра, яке з'являється на екрані, мають виводитися деякі значення параметрів, перед головним циклом програми, який забезпечує повертання до початку обчислень, необхідно розмістити частину програми, яка б задавала первісні значення всіх параметрів. Крім того, на початку роботи програми дуже зручно вивести на екран стисло інформацію про призначення програми, більш детальну інформацію про досліджувану математичну модель з указівкою місця в ній і змісту усіх первинних параметрів, а також первісні ("вшиті") значення всіх параметрів цієї моделі. Це бажано також оформити у виді окремого Script-файлу, наприклад, під ім'ям *ScrFil_Zastavka*.

3. При завершенні роботи програми зазвичай виникає потреба дещо упорядкувати робочий простір, наприклад, очистити його від уведених глобальних змінних (вони, залишаючись у робочому просторі, перешкоджають правильній роботі іншої, наступної програми, яка може мати зовсім інші глобальні змінні, або такі ж за ім'ям, але зовсім інші за типом, змістом і значенням), закрити відкриті програмою графічні вікна (фігури) і т.д. Цю завершальну частину теж можна оформити як окремий Script-файл, наприклад, назвавши його *ScrFil_Kin*.

У цілому типова схема оформлення Script-файлу керувальної програми може бути подана в такому виді:

```
%      <Позначення Script-файлу (ScrFil.m)>
% <Текст коментарю з описом призначення програми>
%      < Порожній рядок >
%      Автор < Прізвище І. Б., дата створення, організація>

ScrFil_Zastavka
k = menu(' Що робити ? ','Продовжити роботу ',' Закінчити роботу ');
if k==1,
    while k==1
        ScrFil_Menu
        ScrFile_Yadro
        k = menu(' Що робити ? ',' Продовжити роботу ',' Закінчити роботу ');
    end
end
ScrFil_Kin
```

6.4. Графічне оформлення результатів

6.4.1. Загальні вимоги до подання графічної інформації

Обчислювальна програма, утворювана інженером-розробником, призначена, по-більшості, для дослідження поведінки розроблювального устрою за різноманітних умов його експлуатації, при різних значеннях його конструктивних параметрів або для розрахунку певних параметрів його поведінки. Інформація, одержувана в результаті виконання обчислювальної інженерної програми, як правило, має форму деякого ряду чисел, кожний з яких відповідає певному значенню деякого параметра (аргументу). Таку інформацію зручніше усього узагальнювати й подавати в графічній формі.

Вимоги до оформлення інженерної графічної інформації відрізняються від вимог до звичайних графіків у математиці. Зазвичай на основі поданої графічної інформації користувач-інженер має мати змогу прийняти якесь інженерне рішення про вибір значень певних конструктивних параметрів, що характеризують досліджуваний процес або технічний устрій, із метою, щоб прогнозоване поведінка технічного устрою задовольняло певні задані умови. Тому інженерні графіки мають бути, як говорять, “читабельними”, тобто мати такий вид, щоб з них легко було “відлічувати” значення функції при будь-яких значеннях аргументу і навпаки з відносною похибкою в декілька відсотків. Це стає можливим, якщо координатна сітка графіків відповідає певним цілим числам якогось десяткового розряду. Як уже раніше відзначалося, *графіки, побудовані системою MatLAB, цілком відповідають цим вимогам.*

Крім цього, інженерна графічна інформація має супроводжуватися достатньо детальним описом, із якого має стати зрозуміло, який об'єкт і за якою математичною моделлю досліджується, наведені числові значення параметрів досліджуваного об'єкта і математичної моделі. Не є зайвим і вказівка імені програми, за допомогою якої отримана ця графічна інформація, а також зведень

про автора програми й дослідника, щоб користувачеві було зрозуміло, до кого і куди треба звертатися для наведення довідок про отриману інформацію.

Задачею інженерної програми часто є порівняння декількох функцій, отриманих при різних сполученнях конструктивних параметрів або параметрів зовнішніх дій. Таке порівняння зручніше і наочніше проводити, якщо згадані функції подані у виді графіків.

При цьому потрібно звернути увагу на наступне:

1) якщо потрібно порівнювати графіки функцій одного аргументу, діапазони змінювання яких не надто відрізняються один від одного (не більш ніж на десятковий порядок, тобто не більш ніж у десять разів), порівняння зручніше усього здійснювати по графіках цих функцій, побудованих в одному графічному полі (тобто у загальних координатних осях); у цьому випадку *варто виводити графіки за допомогою однієї функції **plot***;

2) якщо за тих самих умов діапазони змінювання функцій значно різняться, можна запропонувати два підходи:

- якщо всі порівнювані функції є значеннями величин однакової фізичної природи і ці значення усі додатні, графіки варто виводити також в одне графічне поле, але *в логарифмічному масштабі* (тобто використовувати процедуру *semilogy*);

- за умови, що усі функції мають різну фізичну природу, але аргумент у них загальний і змінюється в однім діапазоні, графіки *потрібно будувати в одному графічному вікні (фігурі), але в різних графічних полях* (користуючись для цього декількома окремими зверненнями до функції *plot* у різних підвікнах графічного вікна, що забезпечується застосуванням процедури *subplot*); при цьому зручно розміщувати окремі графіки один під одним таким чином, щоб однакові значення аргументу у всіх графіках розташовувалися на одній вертикалі;

3) крім згаданих раніше написів у кожному графічному полі, закінчене графічне оформлення будь-якого графічного вікна (фігури) обов'язково повинно містити *додаткову текстову інформацію* такого вмісту:

- стисле повідомлення про об'єкт дослідження;
- математична модель, закладена основу здійснених у програмі обчислень з указівкою імен параметрів і змінних;
- інформація про використані значення параметрів;
- інформація про отримані значення деяких обчислених інтегральних параметрів;
- інформація про ім'я М-файлу використаної програми, виконавця роботи й дату проведення обчислювального експерименту;
- інформація про автора використаної програми й організації, де він працює.

Остання вимога ставить перед необхідністю відокремлювати (за допомогою тієї ж процедури *subplot*) у кожній фігурі місце для виведення зазначеної текстової інформації.

6.4.2. Розбиття графічного вікна на підвікна

Як впливає зі сказаного, при створенні закінченого графічного інженерного документа в системі MatLAB необхідно використовувати процедуру *subplot*. Загальне призначення й застосування функції *subplot* описані в поділі. 1.5.3.

Розглянемо, як забезпечити бажане розбиття всього графічного вікна на окремі поля графіків і текстове підвікно.

Нехай потрібно розбити усе поле графічного вікна так, щоб верхня третина вікна утворила поле виведення тексту, а нижні дві третини утворили єдине поле виведення графіків. Це можна здійснити таким чином:

- перед виведенням текстової інформації у графічне вікно треба встановити команду *subplot(3,1,1)*, яка показує, що весь графічний екран, розділений на три однакові частини по вертикалі, а для наступного виведення буде використана верхня з цих трьох частин (рис. 6.7);
- виведенню графіків у графічне вікно має передувати команда *subplot(3,1,[2 3])*, відповідно до якої графічне вікно розділяється, як і раніше, на три частини по вертикалі, але тепер для виведення графічної інформації буде використаний простір, що об'єднує друге й третє зі створених підвікон (полів) (зверніть увагу, що *підвікна об'єднуються таким самим чином, як елементи вектора у вектор-рядок*).

Щоб створити три окремих поля графіків один під іншим на трьох чвертях екрана по горизонталі, а текстову інформацію розмістити в останній чверті по горизонталі, то це можна зробити (рис. 6.8) так:

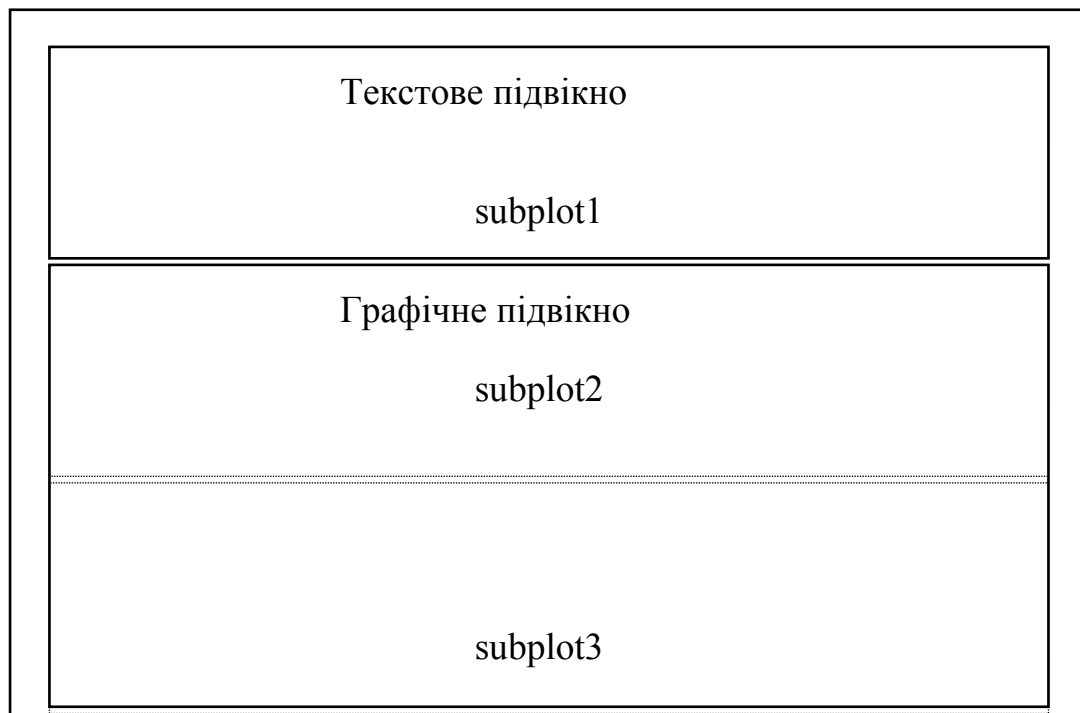


Рис. 6.7. Схема 1 розташування підвікон у графічному вікні

- 1) поділити весь простір фігури на 12 частин - на 3 частини по вертикалі і на 4 частини по горизонталі; при цьому підвікна будуть розташовані так, як показано на рис. 6.7;
- 2) щоб організувати виведення графіків у перше графічне підвікно, треба попередньо ввести команду **subplot(3,4,[1 2 3])**, що об'єднує підвікна sp1, sp2 і sp3 у єдине графічне підвікно;

Графічне підвікно 1			Текстове підвікно
sp1	sp2	sp3	sp4
Графічне підвікно 2			sp8
sp5	sp6	sp7	
Графічне підвікно 3			sp12
sp9	sp10	sp11	

Рис. 6.8. Схема 2 розташування підвікон у графічному вікні

- 3) аналогічно, виведенню графіків у друге графічне підвікно повинно передувати звернення до команди **subplot(3,4,[5 6 7])**, а виведенню графіків у третє графічне підвікно - **subplot(3,4,[9 10 11])**;
- 4) нарешті, до оформлення тексту можна приступити після звернення **subplot(3,4,[4 8 12])**.

6.4.3. Виведення тексту в графічне вікно (підвікно)

Якщо по черзі сформувати підвікна, приміром, відповідно до останньої схеми, не здійснюючи ніяких операцій по виведенню графіків або тексту:

- » **subplot(3,4,[5 6 7])**
- » **subplot(3,4,1:3)**
- » **subplot(3,4,9:11)**
- » **subplot(3,4,[4 8 12])**,

у вікні фігури з'явиться зображення, подане на рис. 6.9.

З його розгляду випливає:

- 1) після звернення до процедури **subplot** у відповідному підвікні з'являється зображення осей координат із позначенням поділок по осях;
- 2) початковий діапазон змінювання координат по обох осях підвікна завжди встановлюється за замовчуванням від 0 до 1;
- 3) поле виведення графіків займає не весь простір відповідного підвікна, - залишається деяке місце навколо поля графіка для виведення заголовка графіка, написів по осях координат та ін.

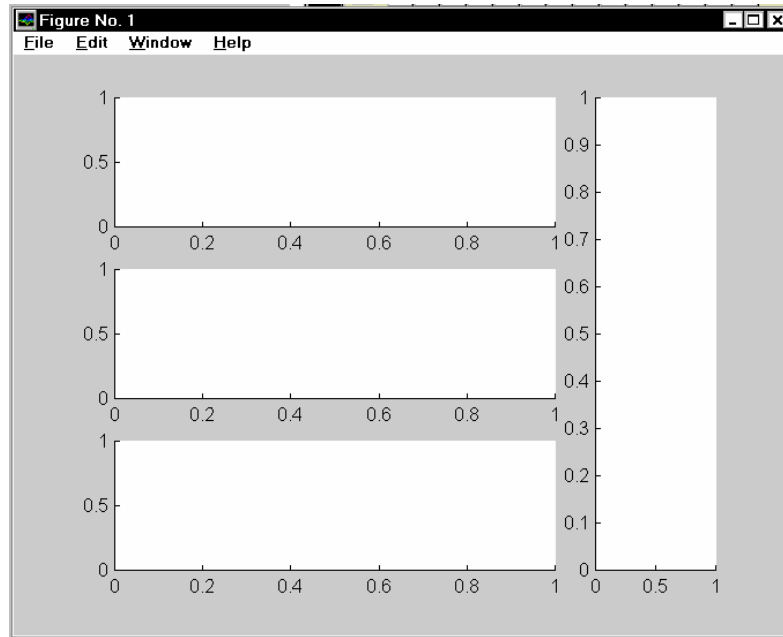


Рис. 6.9. Вид графічного вікна після розбиття його на підвікна

Тому для виведення тексту в одне з підвікон потрібно спочатку очистити це підвікно від зображення осей координат і написів на них. Це робиться за допомогою команди

axis('off').

Наприклад, якщо цю команду ввести після попередніх команд, у вікні фігури зникне зображення координатних осей останнього підвікна (рис. 6.10).

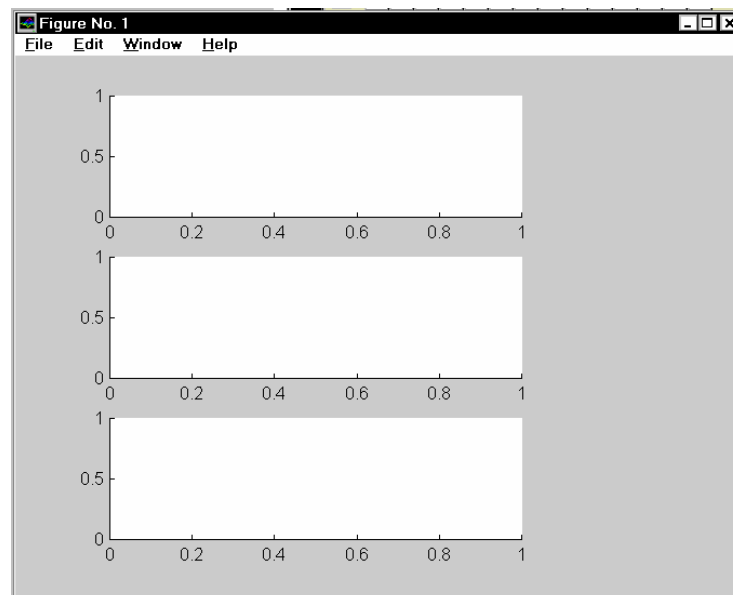


Рис. 6.10. Вид графічного вікна після команди **axis('off')**.

Тепер можна починати виведення тексту в це підвікно.

Основною функцією, яка забезпечує виведення тексту в графічне вікно, є функція **text**. Узагальнена форма звернення до неї має вид:

$h = \text{text}(x, y, \text{'<текст>'}, \text{'FontName'}, \text{'<назва шрифту>'}, \text{'FontSize'}, \text{'<розмір шрифту в пікселях>'}$).

Вона здійснює виведення зазначеного тексту зазначеним шрифтом зазначеного розміру, починаючи з точки підвікна з координатами x і y відповідного поля графіка підвікна. При цьому координати x і y вимірюються в одиницях величин, які відкладаються уздовж відповідних осей графіка підвікна. Через те що, як ми переконалися, діапазон змінювання цих координат дорівнює $[0...1]$, то для того, щоб помістити початок тексту в точку усередині поля графіка, необхідно, щоб його координати x і y були в цьому діапазоні.

Проте можна використовувати і дещо ширшій діапазон, враховуючи те, що поле підвікна більше поля його графіка.

Розглянемо приклад текстового оформлення. Нехай частина програми має вид:

```
subplot(3,4,1:3);      subplot(3,4,5:7);      subplot(3,4,9:11);      subplot(3,4,[4;8;12]);
axis('off');
% Процедура виведення даних в текстове поле графічного вікна
D1 =[2 1 300 1 50];
D2 = [ 0.1 0.02 -0.03 0 1 4 -1.5 2 0.1 -0.15 0 0];
D5 = [0.001 0.01 15 16];
sprogram = 'vsp1'; sname = 'Лазарев Ю.Ф.';
h1=text(-0.2,1,'Исходные параметры:', 'FontSize',12);
h1=text(0,0.95,'Гиротакметров', 'FontSize',10);
h1=text(0.2,0.9,sprintf(' = %g ',D1(3)), 'FontSize',10);
h1=text(-0.2,0.85,sprintf(' = %g ',D1(4)), 'FontSize',10);
h1=text(0.6,0.85,sprintf(' = %g ',D1(5)), 'FontSize',10);
h1=text(-0.2,0.8,sprintf(' = %g ',D1(1)), 'FontSize',10);
h1=text(0.6,0.8,sprintf('J2 = %g ',D1(2)), 'FontSize',10);
h1=text(0,0.75,'Внешних воздействий', 'FontSize',10);
h1=text(-0.2,0.7,sprintf('pst0 = %g ',D2(1)), 'FontSize',10);
h1=text(0.6,0.7,sprintf(' tet0 = %g ',D2(2)), 'FontSize',10);
h1=text(0.2,0.66,sprintf(' fit0 = %g ',D2(3)), 'FontSize',10);
h1=text(-0.2,0.62,sprintf(' psm = %g ',D2(4)), 'FontSize',10);
h1=text(0.6,0.62,sprintf(' tem = %g ',D2(5)), 'FontSize',10);
h1=text(0.2,0.58,sprintf(' fim = %g ',D2(6)), 'FontSize',10);
h1=text(-0.2,0.54,sprintf(' omps = %g ',D2(7)), 'FontSize',10);
h1=text(0.6,0.54,sprintf(' omte = %g ',D2(8)), 'FontSize',10);
h1=text(0.2,0.5,sprintf(' omfi = %g ',D2(9)), 'FontSize',10);
h1=text(-0.2,0.46,sprintf(' eps = %g ',D2(10)), 'FontSize',10);
h1=text(0.6,0.46,sprintf(' ete = %g ',D2(11)), 'FontSize',10);
h1=text(0.2,0.42,sprintf(' efi=%g ',D2(12)), 'FontSize',10);
h1=text(0,0.35,'Интегрирования', 'FontSize',10, 'FontUnderline','on');
h1=text(0,0.3,sprintf('h = %g ',D5(1)), 'FontSize',10);
h1=text(0,0.25,sprintf('hpr = %g ',D5(2)), 'FontSize',10);
h1=text(0,0.2,sprintf('t = %g ',D5(3)), 'FontSize',10);
h1=text(0,0.15,sprintf('tfinal = %g ',D5(4)), 'FontSize',10);
h1=text(-0.3,0.12,'-----', 'FontSize',10);
tm=fix(clock); Tv=tm(4:6);
h1=text(-0.2,0.08,['Программа ' sprogram], 'FontSize',10);
h1=text(-0.3,0.04,['Расчеты провел ' sname], 'FontSize',10);
h1=text(-0.3,0,[sprintf(' %g :',Tv) ' ' date], 'FontSize',10);
h1=text(-0.3,-0.04,'-----', 'FontSize',10);
h1=text(-0.3,-0.08,'Ukraine, KPI, cath. PSON', 'FontSize',10);
```

Виконання цієї програми приводить до появи у вікні фігури зображення, поданого на рис. 6.11.

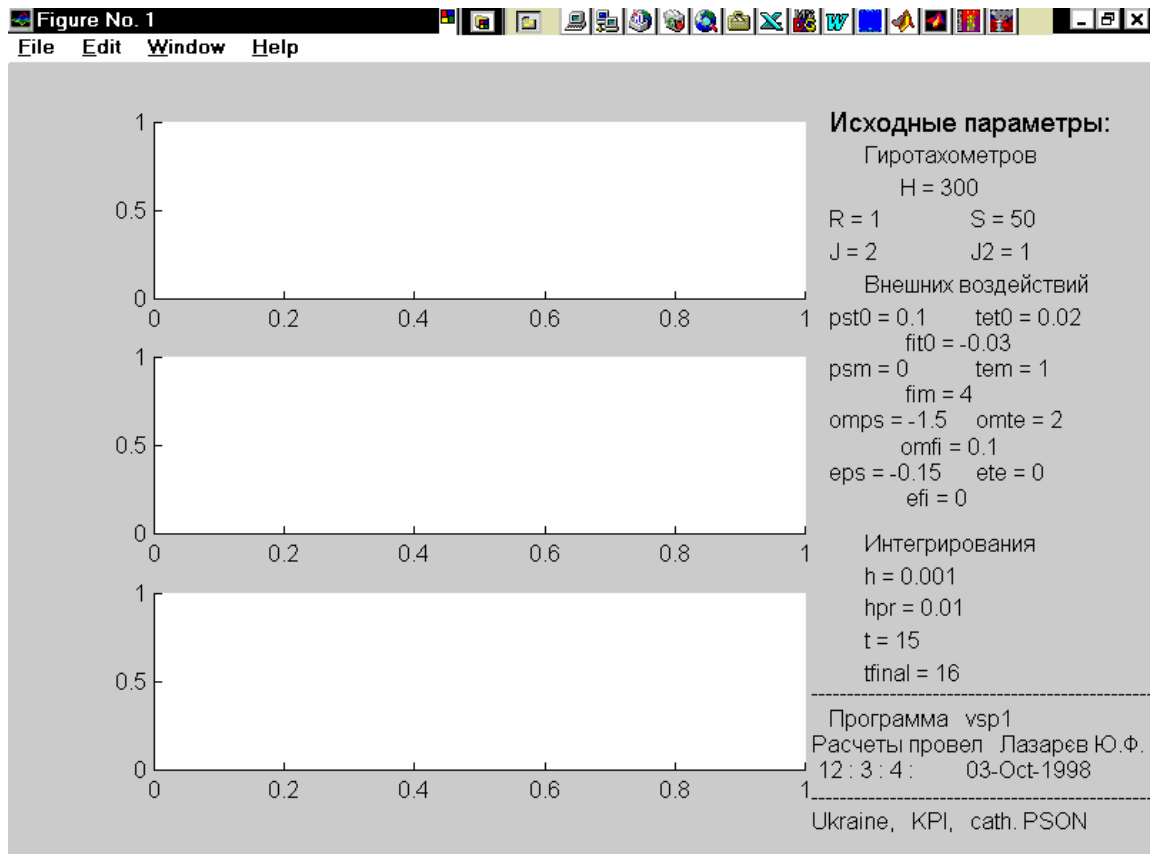


Рис. 6.11. Вид графічного вікна з оформленим текстовим підвіконням

6.5. Функції функцій

Деякі важливі універсальні процедури в MatLAB використовують як змінний параметр ім'я функції, із яким вони оперують, і тому потребують при зверненні до них указівки імені М-файлу, у якому записаний текст програми обчислення деякої іншої процедури (функції). Такі процедури називають **функціями функцій**.

Щоб скористатися такою функцією від функції, необхідно, щоб користувач попередньо створив М-файл, у якому обчислювалося б значення потрібної функції за відомим значенням її аргументу.

6.5.1. Стандартні функції від функцій Matlab

Перелікуємо деякі зі стандартних функцій від функцій, передбачених у MatLAB.

Обчислення інтеграла методом квадратур здійснюється процедурою

$$[I, cnt] = quad(\langle \text{ім'я функції} \rangle, a, b).$$

Тут a і b – нижня й верхня межа змінювання аргументу функції; I – отримане значення інтеграла; cnt – кількість звернень до обчислення функції, поданої М-файлом із назвою, указаним у $\langle \text{ім'я функції} \rangle$. Функція **quad** використовує квадратурні формули Ньютона-Котеса четвертого порядку.

Аналогічна процедура **quad8** використовує більш точні формули 8-го порядку.

Інтегрування звичайних диференціальних рівнянь здійснюють функції *ode23* і *ode45*. Вони можуть застосовуватися як для розв'язування простих диференціальних рівнянь, так і для моделювання складних динамічних систем, тобто систем, поведінка яких можна описати сукупністю звичайних диференціальних рівнянь. При цьому попередньо система звичайних диференціальних рівнянь (ЗДР) має бути подана як система рівнянь 1-го порядку у формі Коші:

$$\frac{dy}{dt} = f(y, t),$$

де y – вектор змінних стану (фазових змінних системи); t – аргумент (зазвичай – час); f – нелінійна вектор-функція від змінних стану y і аргументу t .

Звернення до процедур чисельного інтегрування ЗДР має вид:

$[t, y] = \text{ode23}(\text{'<ім'я функції>'}, \text{tspan}, y_0, \text{options})$

$[t, y] = \text{ode45}(\text{'<ім'я функції>'}, \text{tspan}, y_0, \text{options})$,

де використані параметри мають такий зміст: '<ім'я функції>' – рядок символів, що є ім'ям М-файла, у якому обчислюється вектор-функція $f(y, t)$, тобто *праві частини системи ЗДР*; y_0 – вектор початкових значень змінних стану; t – масив кінцевих значень аргументу, що відповідають крокам інтегрування; y – матриця проінтегрованих значень фазових змінних, в якій кожний стовпчик відповідає одній зі змінних стану, а рядок містить значення змінних стану, що відповідають певному кроку інтегрування; tspan – вектор-рядок $[t_0 \text{ tfinal}]$, що містить два значення: t_0 – початкове значення аргументу t ; tfinal – кінцеве значення аргументу; options – рядок із параметрів, що визначають значення припустимої відносно й абсолютної похибки інтегрування.

Параметр options можна не вказувати. Тоді за замовчуванням припустима відносна похибка інтегрування приймається рівною $1 \cdot 10^{-3}$, абсолютна (по кожній із змінних стану) – $1 \cdot 10^{-6}$. Якщо ж по якихось параметрах ці значення не влаштовують користувача, треба перед зверненням до процедури чисельного інтегрування встановити нові значення припустимих похибок за допомогою процедури *odeset* у такий спосіб:

$\text{options} = \text{odeset}(\text{'RelTol'}, 1e-4, \text{'AbsTol'}, [1e-4 \ 1e-4 \ 1e-5])$.

Параметр RelTol визначає відносну похибку чисельного інтегрування по усіх фазових змінних одночасно, а AbsTol є вектором-рядком, що складається з абсолютних припустимих похибок чисельного інтегрування по кожній з фазових змінних.

Функція *ode23* здійснює інтегрування чисельним методом Рунге-Кутта 2-го порядку, а за допомогою методу 3-го порядку контролює відносні й абсолютні похибки інтегрування на кожному кроці і змінює величину кроку інтегрування так, щоб забезпечити задані межі похибок інтегрування.

Для функції *ode45* основним методом інтегрування є метод Рунге-Кутта 4-го порядку, а величина кроку контролюється методом 5-го порядку.

Обчислення мінімумів і коренів функції здійснюється такими функціями MatLAB:

fmin – відшукування мінімуму функції одного аргументу;

fmins – відшукування мінімуму функції кількох аргументів;

fzero – відшукування нулів (коренів) функції одного аргументу.

Звернення до першої з них у загальному випадку має такий вид:

$$X_{\min} = \mathit{fmin}(\text{'<ім'я функції>'}, X1, X2).$$

Результатом цього звернення буде значення X_{\min} аргументу функції, яке відповідає локальному мінімуму в інтервалі $X1 < X < X2$ функції, заданої М-файлом із зазначеним ім'ям.

Як приклад розглянемо відшукування значення числа π як значення локального мінімуму функції $y = \cos(x)$ на відрізку $[3, 4]$:

» $X_{\min} = \mathit{fmin}(\text{'cos'}, 3, 4)$

$X_{\min} = 3.1416e+000$

Звернення до другої процедури повинно мати форму:

$X_{\min} = \mathit{fmins}(\text{'<ім'я функції>'}, X0),$

при цьому X є вектором аргументів, а $X0$ означає початкове (первинне) значення цього вектора, в околі якого відшукується найближчий локальний мінімум функції, заданої М-файлом із зазначеним ім'ям. Функція *fmins* знаходить вектор аргументів X_{\min} , який відповідає знайденому локальному мінімуму.

Звернення до функції *fzero* повинно мати вид:

$$z = \mathit{fzero}(\text{'<ім'я функції>'}, x0, \text{tol}, \text{trace}).$$

Тут позначено: $x0$ – початкове значення аргументу, в околі якого відшукується дійсний корінь функції, значення якої обчислюються в М-файлі із заданим ім'ям; *tol* – задана відносна похибка обчислення кореня; *trace* – позначення необхідності виводити на екран проміжні результати; z – значення шуканого кореня.

Побудова графіків функції однієї змінної може бути здійснена за допомогою процедури *fplot*. Відмінність її від процедури *plot* у тому, що для побудови графіка функції немає необхідності в попередньому обчисленні значення функції й аргументу. Звернення до неї має вид:

$$\mathit{fplot}(\text{'<ім'я функції>'}, [\text{<інтервал>}], n),$$

де <інтервал> – це вектор-рядок із двох чисел, що задають, відповідно, нижню й верхню межі змінювання аргументу; <ім'я функції> – ім'я М-файла з текстом процедури обчислення значення бажаної функції за заданим значенням її аргументу; n – бажана кількість частин розбиття зазначеного інтервалу. Якщо останню величину не задати, за замовчуванням інтервал розбивається на 25 частин. Хоча кількість частин " n ", на які розбито інтервал змінювання аргументу, визначено, проте насправді кількість значень вектора " x " може бути значно більшою за рахунок того, що функція *fplot* проводить обчислення з додатковим обмеженням, щоб збільшення кута нахилу графіка функції на кожному кроці не перевищувало 10 градусів. Якщо ж воно виявилось більшим, здійснюється роздрібнення кроку змінювання аргументу, але не більше ніж у 20 разів. Останні два числа (10 і 20) можуть бути змінені користувачем, для цього при зверненні слід додати ці нові значення в заголовок процедури в зазначеному порядку.

Якщо звернутися до цієї процедури так:

$[x, Y] = \mathit{fplot}(\text{'<ім'я функції>'}, [\text{<інтервал>}], n)$,

то графік зазначеної функції не відображується на екрані (у графічному вікні). Замість цього обчислюється вектор "x" аргументів і вектор (або матриця) Y відповідних значень зазначеної функції. Щоб при зверненні останнього виду побудувати графік, необхідно зробити це у подальшому за допомогою процедури $\mathit{plot}(x, Y)$.

6.5.3. Запитання для самоперевірки

1. Що розуміється під поняттям "функція функцій"?
6. Які найбільш вживані стандартні функції функцій є в MatLAB?

6.6. Створення функцій від функцій

Деякі алгоритми є загальними для усіх функцій певного типу. Тому їхня програмна реалізація є єдиною для усіх функцій цього типу, але потребує використання алгоритму обчислення конкретної функції. Останній алгоритм може бути зафіксований у виді певного файлу-функції. Щоб перший, більш загальний алгоритм був пристосований для будь-якої функції, потрібно, щоб ім'я цієї функції було деякої змінною, яка могла б набувати визначеного значення (текстового імені файл-функції) тільки при зверненні до основного алгоритму. Такі функції від функцій уже розглядалися в розділі 6.1. До них належать процедури:

- обчислень інтеграла від функції, що потребують вказівки імені M-файлу, що містить обчислення значення подинтегральної функції;
- чисельного інтегрування диференціальних рівнянь, використання яких потребує вказівки імені M-файлу, у якому обчислюються праві частини рівнянь у формі Коші;
- алгоритмів чисельного обчислення коренів нелінійних алгебричних рівнянь (нулів функцій), які потребують вказівки файл-функції, нуль якого відшукується;
- алгоритмів пошуку мінімуму функції, яку, у свою чергу, треба задавати відповідним M-файлом і т.п.

На практиці досить часто виникає необхідність створювати власні процедури такого типу. Matlab надає такі можливості.

6.6.1. Процедура *feval*

У MatLAB будь-яка функція (процедура), наприклад, за ім'ям FUN1, може бути виконана не тільки за допомогою звичайного звернення виду:

$[y_1, y_2, \dots, y_k] = \text{FUN1}(x_1, x_2, \dots, x_n)$,

а і за допомогою спеціальної процедури *feval* у такий спосіб:

$[y_1, y_2, \dots, y_k] = \mathit{feval}(\text{'FUN1'}, x_1, x_2, \dots, x_n)$,

де ім'я функції FUN1 є вже однією із вхідних змінних – символічною змінною — і тому поміщається в апострофи.

Перевагою виклику функції у другій формі є те, що таке звернення не змінює своєї форми при змінюванні ймення функції, наприклад, на FUN6. Це дозволяє уніфікувати звернення до усіх функцій певного типу, тобто таких, що мають однакову кількість вхідних і вихідних параметрів визначеного типу. При цьому ім'я функції (а, виходить, і сама функція, що використовується) може бути довільним і змінюватися при повторних зверненнях.

Через те, що при виклику функції за допомогою процедури feval ім'я функції розглядається як один із вхідних параметрів процедури, останнє (ім'я функції) можна використовувати як змінну й оформляти в М-файлі звернення до неї, не знаючи ще конкретного імені функції.

6.6.2. Приклади створення процедур від функцій

6.6.2.1. Процедура методу Рунге-Кутта 4-го порядку

Нехай задано систему звичайних диференціальних рівнянь (ЗДР) у формі Коші:

$$\frac{dy}{dt} = \mathbf{Z}(y, t),$$

де y – вектор змінних стану системи; t – аргумент (час); \mathbf{Z} – вектор заданих нелінійних функцій, який, власне, і визначає конкретну систему ЗДР.

Якщо значення вектора змінних y стану в момент часу t є відомим, то загальна формула, за якою може бути знайдений вектор y_{out} значень змінних стану системи в момент часу $t_{out} = t + h$ (h - крок інтегрування), має вид:

$$y_{out} = y + h * F(y, t).$$

Векторна функція $F(y, t)$ пов'язана з вектором \mathbf{Z} і може набувати різного вигляду в залежності від обраного методу чисельного інтегрування. Для методу Рунге-Кутта 4-го порядку оберемо таку її форму:

$$\mathbf{F} = (\mathbf{k}_1 + 3 \cdot \mathbf{k}_2 + 3 \cdot \mathbf{k}_3 + \mathbf{k}_4) / 8,$$

$$\text{де } \mathbf{k}_1 = \mathbf{Z}(y, t);$$

$$\mathbf{k}_2 = \mathbf{Z}(y + h \cdot \mathbf{k}_1 / 3, \quad t + h / 3);$$

$$\mathbf{k}_3 = \mathbf{Z}(y + h \cdot \mathbf{k}_2 - h \cdot \mathbf{k}_1 / 3, \quad t + 2h / 3);$$

$$\mathbf{k}_4 = \mathbf{Z}(y + h \cdot \mathbf{k}_3 - h \cdot \mathbf{k}_2 - h \cdot \mathbf{k}_1, \quad t + h).$$

Створимо М-файл процедури, яка здійснює ці обчислення, назвавши його "rko43":

```
function [tout,yout] = rko43(Zpfun,h,t,y)
%RKO43    Інтегрування ЗДР методом Рунге-Кутта 4-го порядку,
%         праві частини яких задані процедурою Zpfun.
%
%   Вхідні змінні:
%   Zpfun - рядок символів, що містить ім'я процедури
%         обчислення правих частин ЗДР
%
%   Звернення: z = fun(t,y), де Zpfun = 'fun'
%         де t - поточний момент часу
%         y   - вектор поточних значень змінних стану
```

```

%          z  - обчислені значення похідних z(i) = dy(i)/dt.
%  h  - крок інтегрування
%  t  - попередній момент часу
%  y  - попереднє значення вектора змінних стану.
%  Вихідні змінні:
%  tout - новий момент часу
%  yout - обчислене значення вектора y через крок інтегрування

% Ю.Ф.Лазарєв, каф. ПСОИ, К П І, Україна
%          Розрахунок проміжних значень похідних
k1 = feval(Zpfun, t, y);
k2 = feval(Zpfun, t+h/3, y+h/3*k1);
k3 = feval(Zpfun, t+2*h/3, y+h*k2-h/3*k1);
k4 = feval(Zpfun, t+h, y+h*(k3+k1-k2));
%          Розрахунок нових значень вектора змінних стану
tout = t + h;
yout = y + h*(k1 + 3*k2 + 3*k3 + k4)/8;
%          Кінець процедури RK043

```

Зверніть увагу на такі обставини:

- звернення до процедури обчислення правих частин не є конкретизованим; ім'я цієї процедури належить до вхідних змінних процедури інтегрування і має бути конкретизовано лише при зверненні до останньої;
- проміжні змінні k є векторами-рядками (те ж стосується і змінних 'y', а також змінних 'z', які обчислюються в процедурі правих частин).

6.6.2.2. Процедура правих частин рівняння маятника

Розглянемо процес створення процедури обчислення правих частин ЗДР на прикладі рівняння маятника, точка підвісу якого поступально переміщується з часом за гармонічним законом:

$$J\ddot{\varphi} + R\dot{\varphi} + mgl[1 + n_{my} \sin(\omega t + \varepsilon_y)] \sin \varphi = -mgl n_{mx} \sin(\omega t + \varepsilon_x) \cos \varphi,$$

де: J – момент інерції маятника; R – коефіцієнт демпфірування; mgl – опорний маятниковий момент маятника; n_{my} – амплітуда віброперевантаження точки підвісу маятника у вертикальному напрямку; n_{mx} – амплітуда віброперевантаження в горизонтальному напрямку; φ – кут відхилення маятника від вертикалі; ω – частота коливань точки підвісу; ε_x , ε_y – початкові фази коливань (з прискорення) точки підвісу в горизонтальному й вертикальному напрямках.

Щоб скласти М-файл процедури обчислення правих частин заданої системи ЗДР, насамперед треба привести це рівняння до форми Коші. Для цього введемо позначення:

$$y_1 = \varphi; \quad y_2 = \dot{\varphi} = \frac{d\varphi}{dt}.$$

Тоді рівняння маятника можна подати у виді сукупності двох диференціальних рівнянь 1-го порядку:

$$\begin{cases} \frac{dy_1}{dt} = y_2 \\ \frac{dy_2}{dt} = -[Ry_2 + mgl(1 + n_{my} \sin(\omega t + \varepsilon_y)) \sin y_1 + mgl n_{mx} \sin(\omega t + \varepsilon_x) \cos y_1] / J \end{cases}$$

Порівнюючи отриману систему з загальною формою рівнянь Коші, можна зробити висновок, що

$$Z_1(y_1, y_2, t) = y_2;$$

$$Z_2(y_1, y_2, t) = -[Ry_2 + mgl(1 + n_{my} \sin(\omega t + \varepsilon_y)) \sin y_1 + mgl n_{mx} \sin(\omega t + \varepsilon_x) \cos y_1] / J.$$

Саме обчислення цих двох функцій і має відбуватися в процедурі правих частин. Назвемо майбутню процедуру `fm0`. Вихідною змінною у ній буде вектор $z = [z_1 \ z_2]$, а вхідними – момент часу t і вектор $y = [y_1 \ y_2]$.

Деякою складністю є те, що постійні коефіцієнти в правих частинах не можна передати в процедуру через її заголовок. Тому об'єднаємо їх у вектор коефіцієнтів $K = [J, R, mgl, n_{my}, n_{mx}, \omega t, \varepsilon_y, \varepsilon_x]$ і віднесемо цей вектор до категорії глобальних: `global K`. Тоді М-файл буде мати вигляд:

```
function z = FM0(t,y);
% Процедура правих частин рівняння Фізичного Маятника.
% Здійснює розрахунок вектора "z" похідних
% від вектора "y" змінних стану по формулах:
%           z(1)=y(2);
%           z(2)=(-mgl*nmx*sin(om*t+ex)*cos(y(1))-R*y(2)-...
%           mgl*(1+nmy*sin(om*t+ey))*sin(y(1)))/J,
% Коефіцієнти передаються в процедуру через глобальний вектор
%           K=[J,R,mgl,nmy,nmx,om,ey,ex]
% Ю.Ф.Лазарєв 5-10-1998р.
global K
z(1) = y(2);
z(2) = (-K(3)*K(5)*sin(K(6)*t+K(8))*cos(y(1)) - K(2)*y(2) - ...
        K(3)*(1+K(4)*sin(K(6)*t+K(7)))*sin(y(1)))/K(1);
% Кінець процедури FM0
```

При використанні цієї процедури слід пам'ятати, що в тексті керувальної програми попередньо має бути оголошений глобальний вектор K за допомогою службового слова **global**, а потім визначені усі вісім його елементів.

Цю процедуру можна дещо ускладнити, групуючи разом обчислення всіх зовнішніх моментів сил, крім моменту сил тяжіння, і оформлюючи їх як окрему процедуру. Для цього спочатку перетворимо початкове рівняння, записуючи його у виді:

$$\varphi'' + \sin \varphi = S(\tau, \varphi, \varphi'),$$

де штрих – позначення похідної за безрозмірним часом $\tau = \omega_0 t$,

$$\omega_0 = \sqrt{\frac{mgl}{J}},$$

а через $S(\tau, \varphi, \varphi')$ позначено деяку задану функцію безрозмірного часу, кута повороту маятника і його безрозмірної швидкості

$$\varphi' = \frac{d\varphi}{d\tau}.$$

В аналізованому випадку ця функція набуває такого виду:

$$S(\tau, \varphi, \varphi') = -2 \cdot \zeta \cdot \varphi' - [n_{mx} \cdot \sin(\nu \cdot \tau + \varepsilon_x) \cdot \cos \varphi + n_{my} \cdot \sin(\nu \cdot \tau + \varepsilon_y) \cdot \sin \varphi],$$

причому безрозмірні величини ζ і ν визначаються виразами:

$$\zeta = \frac{R}{2 \cdot \sqrt{mgl \cdot J}}; \quad \nu = \frac{\omega}{\omega_0}.$$

Така безрозмірна форма подання рівнянь є кращою і зручнішою, тому що дозволяє скоротити кількість параметрів (у нашому випадку замість трьох розмірних параметрів J , R і mgl залишився один – ζ), а також подавати розв'язки рівняння в більш загальній формі.

Винесення обчислення моментів зовнішніх дій в окрему обчислювальну процедуру дозволяє також зробити процедуру правих частин рівняння маятника більш загальною, якщо звернення до процедури обчислення моментів здійснювати теж через функцію *feval*.

Створимо процедуру *MomFM1*, яка буде обчислювати моменти сил, що діють на маятник:

```
function m = MomFM1(t,y);
% Обчислення Моментів сил, що діють на Фізичний Маятник.
% Здійснює розрахунок моменту "m" сил
% за формулою:
%           m = -2*dz*y(2) - (nmx*sin(nu*t+ex)*cos(y(1)) + ...
%                       + nmy*sin(nu*t+ey)*sin(y(1)),
% Коефіцієнти передаються в процедуру через глобальний вектор
%           KM1=[dz,nmy,nmx,nu,ey,ex]
% Ю.Ф.Лазарєв 5-10-1998р.
global KM1
m= -2*KM1(1)*y(2)- (KM1(3)*sin(KM1(4)*t+KM1(6))*cos(y(1)) +...
    KM1(2)*sin(KM1(4)*t+KM1(5))*sin(y(1)));
% Кінець процедури MomFM1
```

Тепер слід перебудувати процедуру правих частин. Назвемо цей варіант FM1:

```
function z = FM1(mpfun,t,y);
% Процедура правих частин рівняння Фізичного Маятника.
% Здійснює розрахунок вектора "z" похідних
% векторів "y" змінних стану по формулах:
%           z(1)=y(2);
%           z(2)= - sin(y(1)) +S(t,y),
% де
% вхідні параметри:
%           mpfun - ім'я процедури S(t,y)
%                               mpfun = 'S';
%           t - поточний момент часу;
%           y - поточне значення вектора змінних стану;
% вихідні параметри:
%           z - вектор значень похідних від змінних стана.
```

```
% Ю.Ф.Лазарєв 5-10-1998р.
z(1) = y(2);
z(2) = - sin(y(1)) + feval(mpfun,t,y);
% Кінець процедури FM1
```

Через те що вид звернення до процедури правих частин змінився (додано нову вхідну символічну змінну `mpfun` – ім'я процедури обчислення моментів), необхідно також перебудувати і процедуру чисельного методу. Назвемо її `RKO43m`:

```
function [tout,yout] = rko43m(Zpfun,Mpfun,h,t,y)
%RKO43m Інтегрування ОДУ методом Рунге-Кутта 4-го порядку,
% праві частини яких задані процедурами Zpfun і Mpfun.
% Вхідні параметри:
%           Zpfun - рядок символів, що містить ім'я процедури
%                   обчислення правих частин ЗДР.
%           Звернення: z = fun(Mpfun,t,y),
%                   де Zpfun = 'fun',
%           Mpfun - рядок з ім'ям процедури, до якого
%                   звертається процедура fun;
%           t - поточний момент часу
%           y - вектор поточних значень змінних стану
%           z - обчислені значення похідних z(i) = dy(i)/dt.
%           h - крок інтегрування
%           t - попередній момент часу
%           y - попереднє значення вектори змінних стану.
% Вихідні параметри:
%           tout - новий момент часу
%           yout - нове значення вектора змінних стану
%                   через крок інтегрування
% Розрахунок проміжних значень похідних
k1 = feval(Zpfun, Mpfun,t, y);
k2 = feval(Zpfun, Mpfun, t+h/3, y+h/3*k1);
k3 = feval(Zpfun, Mpfun, t+2*h/3, y+h*k2-h/3*k1);
k4 = feval(Zpfun, Mpfun, t+h, y+h*(k3+k1-k2));
%           Розрахунок нових значень вектора змінних стану
tout = t + h;
yout = y + h*(k1 + 3*k2 + 3*k3 + k4)/8;
% Кінець процедури RKO43m
```

Така форма подання процедури обчислення правих частин диференціальних рівнянь є незручною, бо, по-перше, процедуру виду FM1 не можна використовувати при інтегруванні процедурами MatLAB `ode23` і `ode45` (останні потребують, щоб у процедурі правих частин було тільки два вхідні параметри, а в процедурі FM1 їх три), а, по-друге, така форма вимагає необхідність створення нових М-файлів методів чисельного інтегрування.

Можна цього уникнути, переробляючи ім'я додаткової функції `Mpfun` на глобальну змінну. Тоді процедура правих частин може бути записана так:

```
function z = FM2(t,y);
% Процедура правих частин рівняння Фізичного Маятника.
% Здійснює розрахунок вектора "z"
% похідних вектору "y" змінних стану по формулах:
%           z(1)=y(2);
%           z(2)= - sin(y(1)) +S(t,y),
```

```

% Вхідні параметри:
%   mpfun - ім'я процедури S(t,y) - передається як глобальна змінна
%   mpfun = 'S';
%   t - поточний момент часу;
%   y - поточне значення вектора змінних стану;
% Вихідні параметри:
%   z - вектор значень похідних від змінних стану.

% Ю.Ф.Лазарєв 7-10-1998р.
%
global MPFUN
z(1) = y(2);
z(2) = - sin(y(1)) + feval(MPFUN,t,y);
% Кінець процедури FM2

```

Тепер процедура FM2 має тільки два вхідні параметри, що передаються через заголовок, і може бути використана будь-якою процедурою чисельного методу інтегрування, у тому числі – процедурами *ode23* і *ode45*. Необхідно лише пам'ятати, що в основній програмі змінній MPFUN треба присвоїти деяке символічне значення (ім'я функції, яка буде використана в процедурі правих частин), і вона має бути оголошеною як глобальна. Наприклад, якщо буде використана раніше створена процедура MomFun1, у Script-файлі має бути присутнім рядки

```

global MPFUN
MPFUN = 'MomFm1';

```

6.7. Програма моделювання руху маятника

Раніше були розглянуті основні перешкоди, що стоять на шляху створення складних програм, і засоби їхнього подолання. Тепер, з огляду на це, спробуємо скласти й випробувати в роботі одну з досить складних комплексних програм.

Постановка задачі. Нехай потрібно створити програму, що дозволила б моделювати рух фізичного маятника з точкою підвісу, що вібрує, шляхом чисельного інтегрування диференціального рівняння цього руху.

Диференціальне рівняння руху маятника для цієї задачі можна прийняти таким:

$$J\ddot{\phi} + R\dot{\phi} + mgl[1 + n_{my} \sin(\omega \cdot t + \varepsilon_y)] \sin \phi = -mgl n_{mx} \sin(\omega t + \varepsilon_x) \cos \phi,$$

де J – момент інерції маятника; R – коефіцієнт демпфірування; mgl – опорний маятниковий момент маятника; n_{my} – амплітуда віброперевантаження точки підвісу маятника у вертикальному напрямку; n_{mx} – амплітуда віброперевантаження в горизонтальному напрямку; ϕ – кут відхилення маятника від вертикалі; ω – частота коливань точки підвісу; ε_x , ε_y – початкові фази коливань (з прискорень) точки підвісу в горизонтальному і вертикальному напрямках.

Потрібно створити таку програму, яка дозволяла б обчислювати закон змінювання кута відхилення маятника від вертикалі з часом за довільних, встанов-

люваних користувачем значень усіх вищезазначених параметрів маятника і поступального руху основи, а також за довільних початкових умов.

Обчислення будемо здійснювати шляхом чисельного інтегрування за допомогою стандартної процедури `ode45`.

Перетворення рівняння. Для підготування диференціальних рівнянь до чисельного інтегрування насамперед необхідно *привести ці рівняння до нормальної форми Коші*. Бажано також подати їх у безрозмірній формі.

Для поданого рівняння це було зроблено в попередньому розділі.

Запис М-файлу процедури правих частин. Наступним кроком підготовки програми є складання і запис на диск тексту процедури обчислення правих частин отриманої системи ДР у формі Коші.

Ця процедура була створена в попередньому розділі й остаточно вона має вид:

Файл FM2 . m

```
function z = FM2(t,y);
% Процедура правих частин рівняння Фізичного Маятника.
% Здійснює розрахунок вектора "z" похідних вектора
% "y" змінних стану по формулах:
%           z(1)=y(2);
%           z(2)=-sin(y(1)) +S(t,y),
% де
% Вхідні параметри:
%           t - поточний момент часу;
%           y - поточне значення вектора змінних стану;
%           MPFUN - ім'я процедури S(t,y) - глобальна змінна
%                   MPFUN = 'S';
% Вихідні параметри:
%           z - вектор значень похідних від змінних стану
.
% Ю.Ф.Лазарєв 7-10-1998р.
%
global MPFUN
z(1) = y(2);
z(2) = - sin(y(1))+ feval(MPFUN,t,y);
z =z';
% Кінець процедури FM2
```

Примітка. Зверніть увагу на незначну, але істотну відмінність наведеної процедури від аналогічної процедури попереднього розділу – наявність наприкінці процедури операції транспонування вектора похідних. Це обумовлено тим, що *процедура `ode45` потребує, щоб вектор похідних був обов'язково стовпцем*.

Як додаткову процедуру, що використовується в процедурі FM2, оберемо раніше створену процедуру MomFM1, яку запишемо у файл MomFM1.

Файл MomFM1 . m

```
function m = MomFM1(t,y);
% Обчислення Моментів Сил, що діють на Фізичний Маятник.
% Здійснює розрахунок моменту "m" сил
% по формулі:
% m =-2*dz*y(2) - (nmх*sin(nu*t+ex)*cos(y(1)) +...
% + nму*sin(nu*t+ey)*sin(y(1)),
% Коефіцієнти передаються в процедуру через
```

```
% глобальний вектор KM1=[dz,nmy,nmx,nu,ey,ex]
% Ю.Ф.Лазарєв 5-10-1998р.
global KM1
m = -2*KM1(1)*y(2)- KM1(3)*sin(KM1(4)*t+KM1(6))*cos(y(1)) -...
      KM1(2)*sin(KM1(4)*t+KM1(5))*sin(y(1));
% Кінець процедури MomFM1
```

Очевидно, що у викличному Script-файлі треба передбачити оголошення імені додаткового файлу MomFM1 як глобальної змінної MPFUN, а також забезпечити оголошення глобальної змінної по імені KM1 і завдання значень цього числового масиву з п'ятьох елементів.

Створення керуючого (головного) Script-файлу. Головний файл створимо відповідно до рекомендацій поділу 6.4.5 :

Файл FizMayatn2 . m

```
% FizMayatn2
% Керуюча програма дослідження руху фізичного маятника,
% встановленого на поступально віброуючій основі

% Лазарєв Ю.Ф., кафедра ПСОН, КПІ, Україна, 7-10-1998р.
%
FizMayatn2_Zastavka
k = menu(' Що робити ? ', ' Продовжити роботу ', ' Закінчити роботу ');
if k==1,
    while k==1
        FizMayatn2_Menu
        FizMayatn2_Yadro
        k = menu(' Що робити ? ', ' Продовжити роботу ', ' Закінчити роботу ');
    end
end
clear global
clear
% Кінець FizMayatn2
```

Як бачимо, програма викликає три додаткових Script-файли - FizMayatn2_Zastavka, FizMayatn2_Menu і FizMayatn2_Yadro. Тому потрібно створити ще ці три М-файли.

Створення Script-файла заставки. Як зазначалося, цей файл має містити оператори виведення на екран інформації про основні особливості математичної моделі, реалізованої у програмі, і введення первинних значень параметрів цієї моделі. Нижче приведений текст М-файла FizMayatn2_Zastavka.

Файл FizMayatn2 Zastavka . m

```
% FizMayatn2_Zastavka
% Частина (виведення заставки на екран)
% програми FizMayatn2
% Ю.Ф.Лазарєв, кафедра ПСОН, КПІ, Україна, 24-09-1998р.
%
% Введення "вшитих" значень
sprogram = 'FizMayatn6.m';
sname = 'Лазарєв Ю.Ф.';
KM1 = [0 0 0 0 0];
MPFUN = 'MomFm1';
global KM1 MPFUN
```

```

tfinal =2*pi*5;
fi0 =pi/180; fit0 = 0;
clc
disp( [' Це програма, що здійснює інтегрування рівняння ';...
' Фізичного Маятника при поступальній вібрації точки підвісу ';...
' у формі ';...
' fi'' + sin(fi) = - 2*dz*fi'' - ';...
' - nmy*sin(nu*t+ey)*sin(fi) -nmх*sin(nu*t+ex)*cos(fi)';...
' де fi - кут відхилення маятника від вертикалі, ';...
' dz - відносний коефіцієнт загасання, ';...
' nu - відносна частота вібрації точки підвісу, ';...
' nmy,nmх - амплітуди віброперевантаження у вертикальному ';...
' і горизонтальному напрямках відповідно, ';...
' ey,ex - початкові фази коливань у вертикальному ';...
' і горизонтальному напрямках відповідно, ';...
' KM1 = [dz,nmy,nmх,nu,ey,ex] - матриця коефіцієнтів '])
% Кінець FizMayatn2_Zastavka

```

У ньому здійснюється присвоювання первісних ("вшитих") значень усім параметрам заданого диференційного рівняння, а також параметрам чисельного інтегрування – початковим умовам руху маятника й тривалості процесу інтегрування. Частина цих параметрів об'єднується в єдиний глобальний вектор KM1. Одночасно змінній MPFUN, що буде використовуватися при інтегруванні, присвоюється значення 'MomFm1'.

Створення файлу меню. Зміст файлу меню FizMayatn2_Menu наведено нижче.

Файл FizMayatn2_Menu . m

```

% FizMayatn2_Menu
% Частина (здійснююча діалогове змінювання даних)
% програми FizMayatn2
% Ю.Ф.Лазарєв, кафедра ПСОН, КПІ, Україна, 24-09-1998р.
k=1;
while k<10
    disp(' ')
    disp(' Зараз встановлено ')
    disp([sprintf(' Початковий піг (градуси) = %g', fi0*180/pi),...
    sprintf(' Початкова швидкість = %g', fit0)])
    disp(sprintf(' Число періодів = %g', tfinal/2/pi))
    KM1
% KM1=[dz,nmy,nmх,nu,ey,ex]
    k = menu(' Що змінювати ? ', ...
    sprintf(' Відносний к-нт згасання = %g', KM1(1)),...
    sprintf(' Перевантаж.(вертикаль) = %g', KM1(2)),...
    sprintf(' Перевантаж.(горизонталь) = %g', KM1(3)),...
    sprintf(' Відносну частоту = %g', KM1(4)),...
    sprintf(' Фазу (вертикаль) = %g', KM1(5)),...
    sprintf(' Фазу (горизонталь) = %g', KM1(6)),...
    sprintf(' Початковий кут (градуси) = %g', fi0*180/pi),...
    sprintf(' Початкову швидкість = %g', fit0),...
    sprintf(' Кількість періодів = %g', tfinal/2/pi),...
    ' Нічого не змінювати ');
    disp(' ')
    if k<7, KM1(k) = input(['Зараз KM1(',num2str(k),sprintf(') = %g', KM1(k)),...
    ' Введіть нове значення = ']);
    elseif k==7, fi0 = input([sprintf('Зараз fi0 = %g градусів', fi0*180/pi),...
    ' Введіть нове значення = ']);

```

```

        fi0 = fi0*pi/180;
    elseif k==8,    fit0 = input(sprintf('Зараз fit0 = %g', fit0),...
        ' Введіть нове значення = ');
    elseif k==9,tfinal=input(sprintf('Зараз кількість періодів = %g', tfinal/2/pi),...
        ' Введіть нове значення = ');
        tfinal = tfinal*2*pi;
    end
end % FizMayatn2_Menu

```

Файл здійснює організацію діалогового введення-змінювання значень параметрів фізичного маятника, руху основи й параметрів чисельного інтегрування у відповідності зі схемою, описаною в поділі 6.4.4.

Створення файлу ядра програми. Основні дії по організації процесу чисельного інтегрування й виведенню графіків зосереджені у файлі по імені FizMayatn2_Yadro:

Файл FizMayatn2_Yadro . m

```

% FizMayatn2_Yadro
% Частина (здійснююча основні обчислення)
% програми FizMayatn2
% Ю.Ф.Лазарєв, кафедра ПСОН, КПІ, Україна, 7-10-1998р.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1. Підготовка початкових умов
%-----
t = 0; tf = tfinal;    y0 =[fi0 fit0];
options = odeset('RelTol',1e-8,'AbsTol',[1e-10 1e-10]);
%-----
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 6. Організація циклу інтегрування
%-----
[t,y] = ode45('FM2',[0 tf],y0,options);
%-----
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3. Виведення графіків
subplot(2,1,2);
plot(t/2/pi,y(:,1)*180/pi);grid;
title('Відхилення від вертикалі','FontSize',14);
xlabel('Час (у періодах малих власних коливань)','FontSize',12);
ylabel('Кут у градусах','FontSize',12);
subplot(2,4,1:2);
plot(y(:,1)*180/pi,y(:,2));grid;
title('Фазова траєкторія','FontSize',14);
xlabel('Кут у градусах','FontSize',12);
ylabel('Швидкість','FontSize',12);
%-----
% Виведення текстової інформації у графічне вікно
subplot(2,4,3:4);    axis('off');
h1=text(0,1.1,'Моделювання руху фізичного маятника', 'FontSize', 14, 'FontWeight',
'Bold');
h1=text(0.4, 1,'за рівнянням','FontSize',12);
h1=text(0,0.9,'fi" + 2*dz*fi" + [1+nmy*sin(nu*t+ey)]*sin(fi) =','FontSize',14);
h1=text(0.55,0.8,' = - nmх*sin(nu*t+ex)*cos(fi)','FontSize',14);
h1=text(0,0.7,'за таких значень параметрів:', 'FontSize',12);
h1=text(0.45,0.6,sprintf('dz = %g',KM1(1)), 'FontSize',12);
h1=text(0,0.5,sprintf('nmy = %g',KM1(2)), 'FontSize',12);
h1=text(0.7,0.5,sprintf('nmх = %g',KM1(3)), 'FontSize',12);
h1=text(0,0.4,sprintf('ey = %g',KM1(5)), 'FontSize',12);
h1=text(0.7,0.4,sprintf('ex = %g',KM1(6)), 'FontSize',12);

```



```

h1=text(0.45,0.3,sprintf('nu = %g',KM1(4)), 'FontSize', 12);
h1=text(0,0.2,'i початкових розумів:', 'FontSize', 12);
h1=text(0,0.1,[sprintf('fi(0) = %g',fi0*180/pi),' градусів'], 'FontSize', 12);
h1=text(0.7,0.1,sprintf('fi''(0) = %g',fit0), 'FontSize', 12);
h1=text(0,0.05,);-----');
h1=text(0,-0.2,);-----');
h1=text(-0.05,-0.05,['Програма ',sprogram]);
h1=text(0.55,-0.05,'Автор - Лазарєв Ю.Ф., каф. ПСОН');
h1=text(0,-0.15,['Виконав ',sname]);
tm=fix(clock); Tv=tm(4:5);
h1=text(0.65,-0.15,[sprintf(' %g:',Tv),' ',date]);
% Кінець файла FizMayatn2_Yadro

```

Як бачимо, основні операції включають три головних поділи – уведення початкових умов, організації циклу інтегрування й організації оформлення графічного вікна виведення.

Налагоджування програми. Налагодження програми складається із запуску головного М-файлу FizMayatn2, перевірки правильності функціонування всіх частин програми, внесення коректив у тексти використовуваних М-файлів доти, поки всі запрограмовані дії не будуть задовольняти задані вимоги.

Крім того, сюди відносяться і дії по перевірці "адекватності", тобто відповідності одержуваних програмою результатів окремим апріорно відомим випадкам поведження досліджуваної системи. Очевидно, для такої перевірки потрібно підібрати декілька сукупностей значень параметрів системи, при яких поведження системи є цілком відомим із попередніх теоретичних або експериментальних досліджень. Якщо отримані програмою результати цілком узгодяться з відомими, програма є адекватною прийнятій математичній моделі.

У приведеному тексті програми "вшиті" початкові значення параметрів відповідають вільному рухові маятника без впливу тертя. За таких умов рух маятника є незатухаючими коливаннями відносно положення вертикалі. Тому, якщо програма працює вірно, на графіках повинні спостерігатися саме такі коливання маятника. Результат роботи створеної програми при цих умовах поданий на рис. 6.12. Як очевидно, у цьому відношенні програма є адекватною прийнятій математичній моделі.

Проведення досліджень. Створена програма тепер може бути використана для моделювання й дослідження різноманітних нелінійних ефектів, які спостерігаються у фізичного маятника при поступальній вібрації точки його підвісу. На рис. 6.13 – 6.18 продемонстровані деякі можливості створеної програми.

На рис. 6.13 приведені параметричні коливання маятника, що можуть виникати при вібрації точки підвісу у вертикальному напрямку. З рисунка очевидно, що в цьому випадку коливання маятника щодо вертикалі спочатку збільшуються по амплітуді, а потім усталюються, причому частота сталих коливань удвічі менше за частоту вібрації основи і складає приблизно 1,15.

Випрямний ефект маятника проілюстрований на рис. 6.14. З нього наочно видно, що одночасна вібрація основи у вертикальному й горизонтальному

напрямок призводить до відхилення середнього положення маятника від вертикалі на кут приблизно -5 градусів.

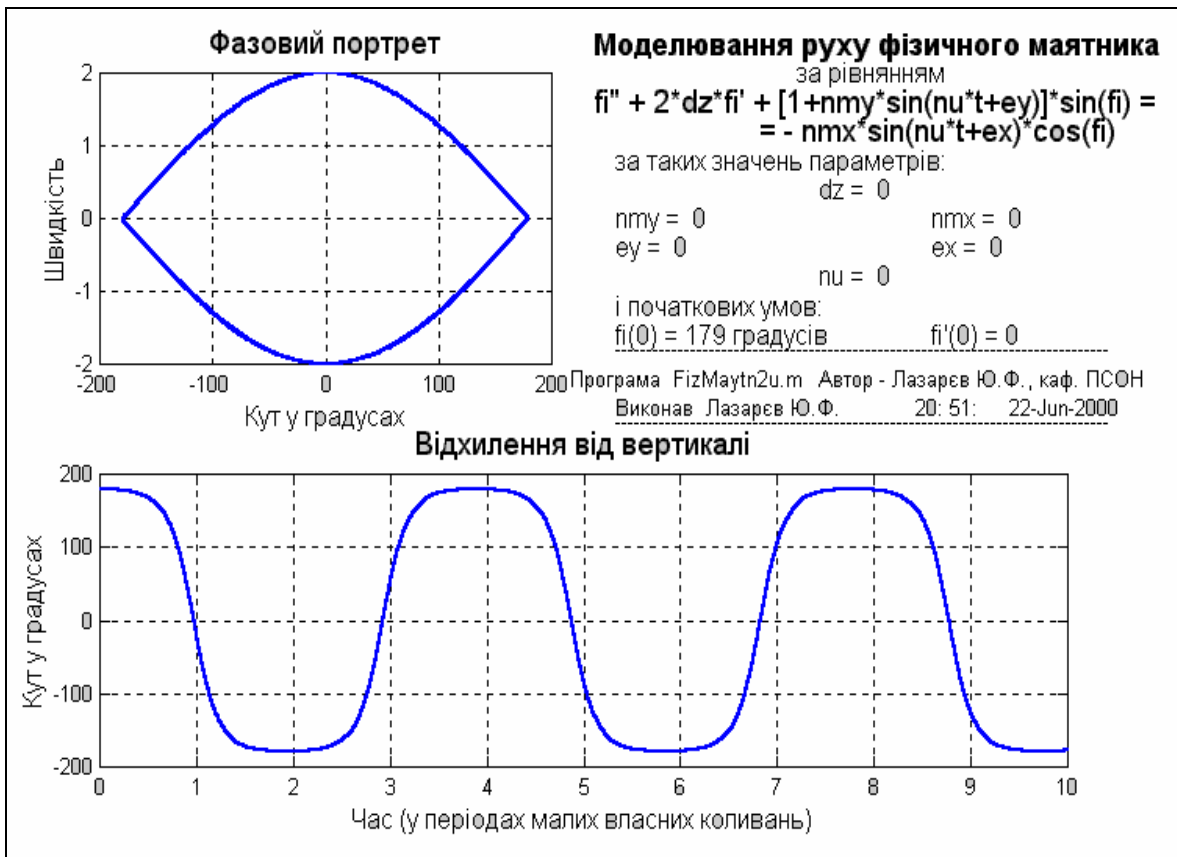


Рис. 6.12. Власні незгасаючі коливання з великою амплітудою

Рис. 6.15 ілюструє стаціонарні коливання маятника щодо верхнього положення рівноваги, які можуть спостерігатися при інтенсивній вертикальній вібрації. Ці коливання при наявності тертя загасають, як показано на рис. 6.16, і маятник "застигає" у верхньому положенні.

Нарешті, рис. 6.17 і 6.18 демонструють можливість значних відхилень середнього положення маятника від вертикалі і при суто горизонтальній вібрації основи. Відповідно до них це відхилення сягає величини понад 40° при прийнятих значеннях параметрів вібрації, причому напрямок відхилення залежить від початкових умов руху маятника.

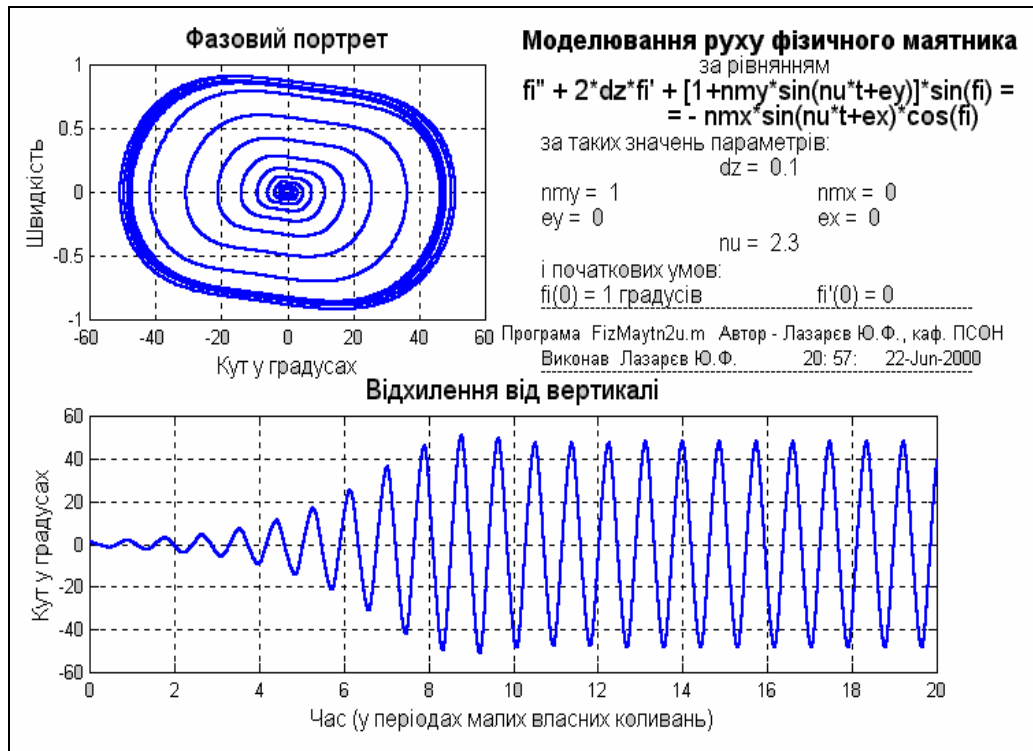


Рис. 6.13. Збудження параметричних коливань

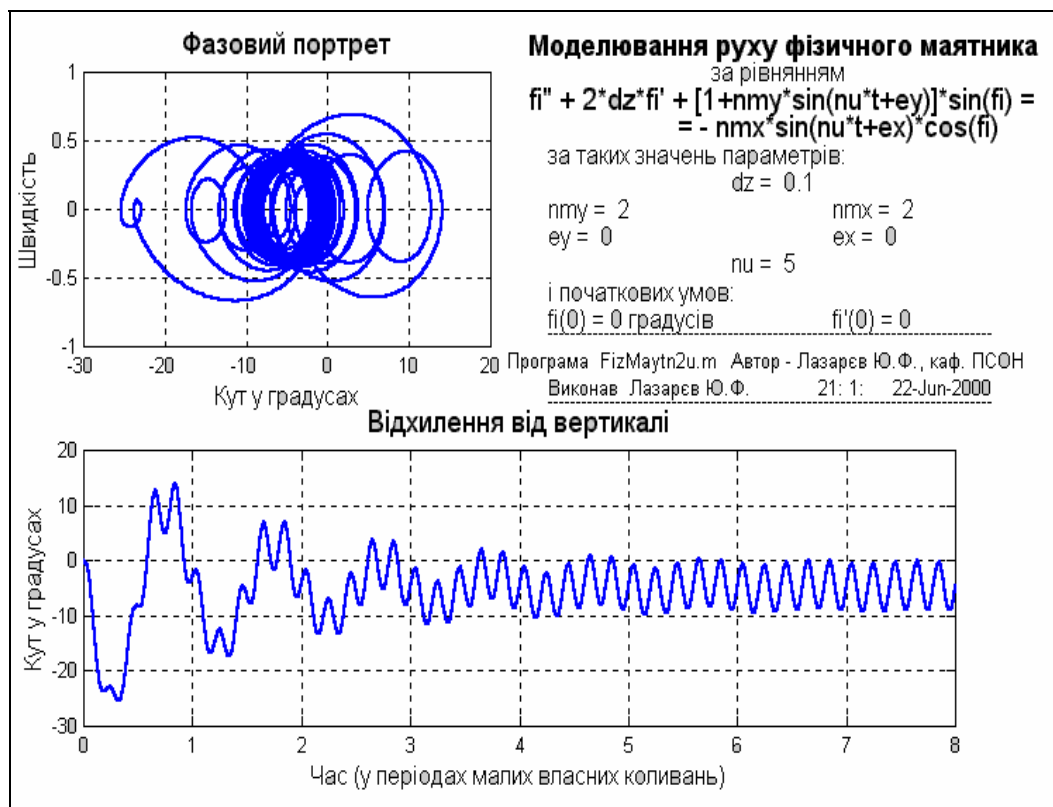


Рис. 6.14. Випрямний ефект при навкісній вібрації точки підвісу

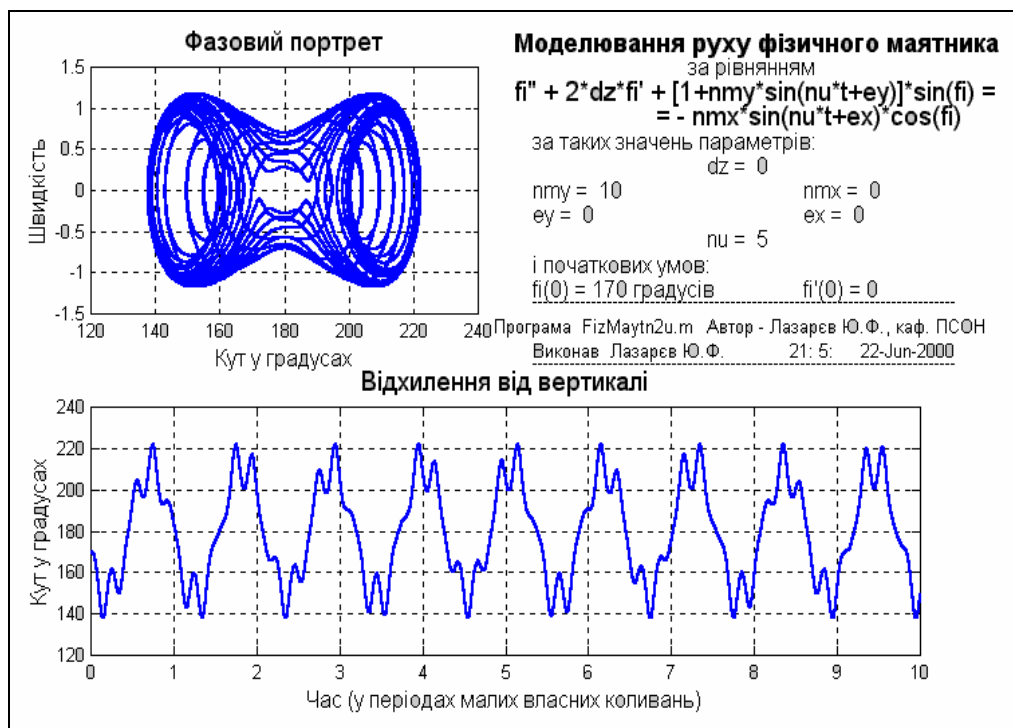


Рис. 6.15. Стійкість верхнього положення рівноваги

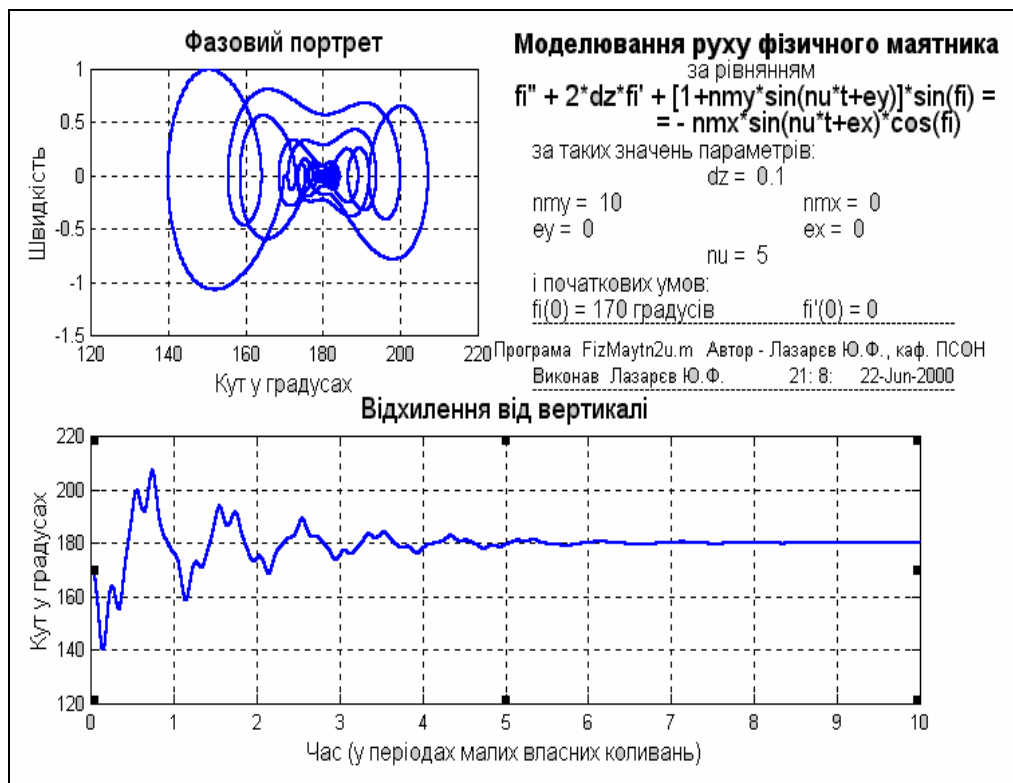


Рис. 6.16. Стійкість верхнього положення рівноваги при демпфіванні

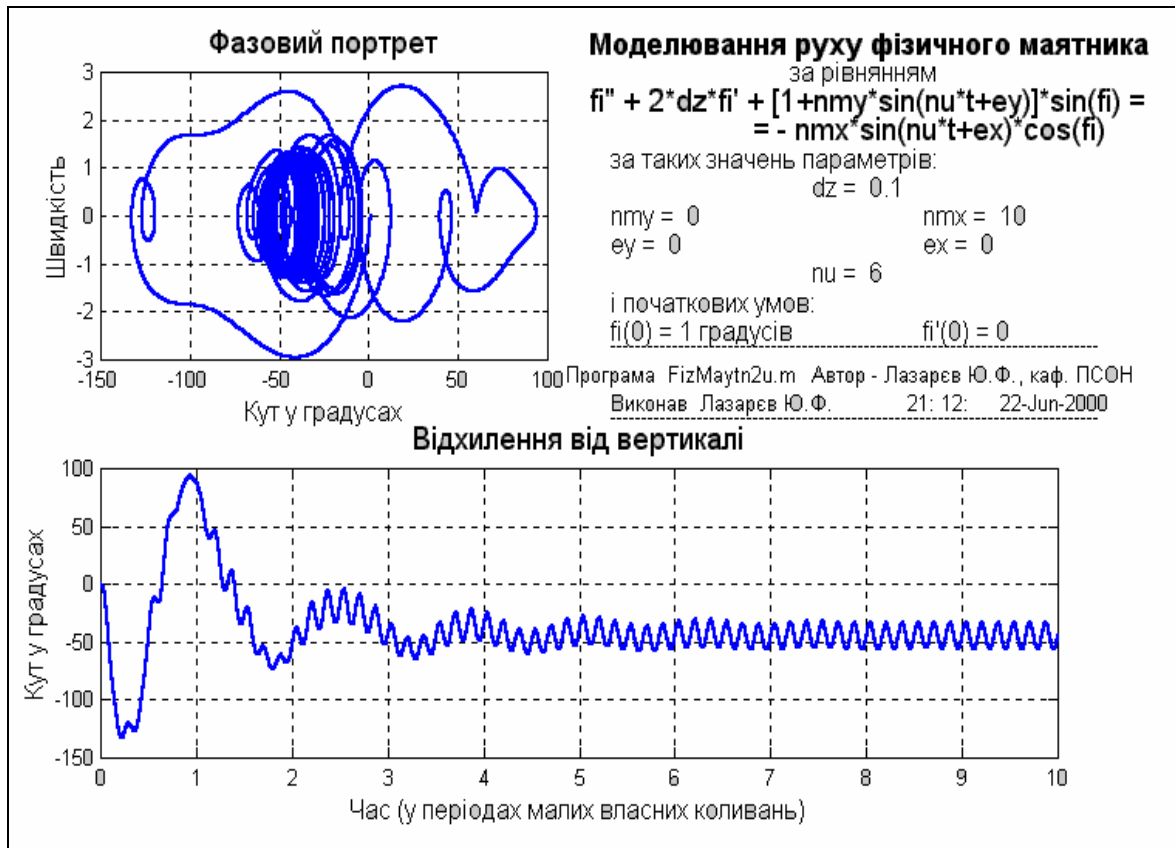


Рис. 6.17. Випрямний ефект при горизонтальній вібрації точки підвісу

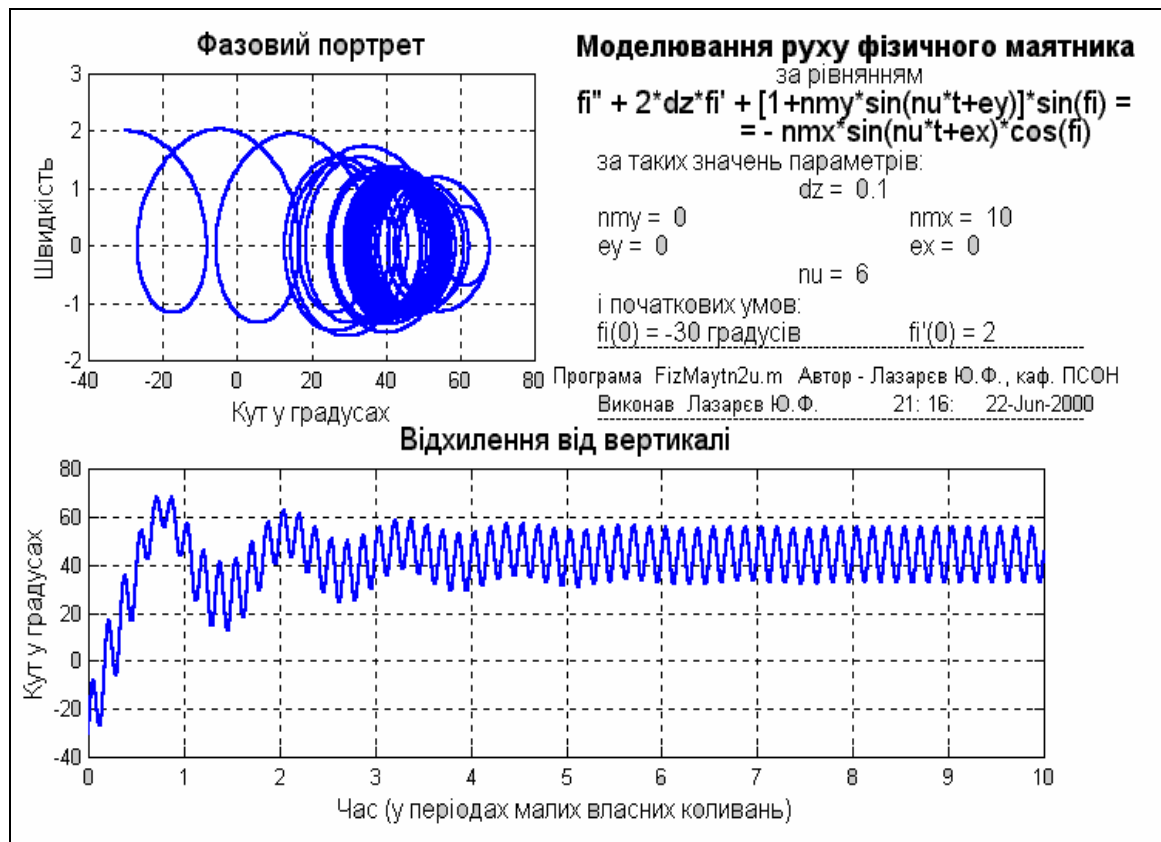


Рис. 6.18. Випрямний ефект при горизонтальній вібрації точки підвісу за інших початкових умов

6.8. Література

1. Лазарєв Ю. Ф. Початки програмування у середовищі MatLab. Навч. посібник. – К.: Корнійчук, 1999. – 160 с.
6. Лазарєв Ю. Ф. MatLAB 5.x. – К.: Издат. группа ВНУ, 2000. - 384 с.
3. Лазарєв Ю. Ф. Моделирование процессов и систем в MATLAB. Учебный курс. – СПб.: Питер; Киев: Издат. группа ВНУ, 2005. – 512 с.
4. Лазарєв Ю. Ф. Моделювання на ЕОМ. Навч. посібник. – К.: Корнійчук, 2007. = 290 с.

7. ПАКЕТ ПРОГРАМ SIMULINK

Пакет SimuLink дозволяє здійснювати дослідження (моделювання у часі) поведінки динамічних нелінійних систем. Утворення чисельної моделі досліджуваної системи здійснюється шляхом графічного складання у спеціальному вікні схеми з'єднань елементарних візуальних блоків, що містяться в бібліотеках SimuLink. Кожний блок фактично являє собою математичну програму. Лінії з'єднання блоків перетворюються на зв'язки між цими програмами, які дозволяють визначити послідовність виклику програм і пересилання інформації. У результаті такого складання утворюється програмна модель, яку надалі називатимемо S-моделлю і яка зберігається у файлі з розширенням **.mdl**. Такий процес утворення обчислювальних програм прийнято називати візуальним програмуванням.

Створення моделей у пакеті SimuLink ґрунтується на використанні технології Drag-and-Drop (*Перетягни й Залиши*). Як "цеглинки" при побудові S-моделі використовуються модулі (блоки), що зберігаються в бібліотеці SimuLink. S-модель може мати ієрархічну структуру, тобто складатися з моделей більш низького рівня, причому кількість рівнів ієрархії є практично необмеженою. Протягом моделювання є можливість спостерігати за процесами, що відбуваються в системі. Для цього використовуються спеціальні блоки "оглядові вікна", що входять до складу бібліотеки SimuLink. Склад бібліотеки SimuLink може бути поповнений користувачем за рахунок розробки власних блоків.

Використання SimuLink є особливо зручним при моделюванні систем, які складаються із з'єднаних певним чином окремих функціональних пристроїв, поведінка яких описується відомими залежностями. Тоді схема з'єднань візуальних блоків у вікні блок-схеми S-моделі збігається з реальними зв'язками між цими пристроями. Ця обставина суттєво спрощує програмний аналіз і синтез систем автоматичного керування.

7.1. Запуск SimuLink

Запуск SimuLink складається з двох процедур:

- 1) виклику створеної раніше S-моделі шляхом введення у командному рядку командного вікна MatLAB ймення відповідного MDL-файла або обравши команду **New > Model** у меню **File**, якщо S-модель ще тільки потрібно утворити; у першому випадку при цьому виникне нове вікно з блок-схемою, а у другому - порожнє вікно **untitled** (вікно, де має бути утворена блок-схема нової S-моделі, MDL-файлу, рис. 7.1);
- 2) виклику браузера бібліотеки SimuLink шляхом натиснення відповідної піктограми у лінійці інструментів командного вікна; виникне вікно **Simulink Library Browser** (рис. 7.2), яке містить перелік основних поділів бібліотеки SimuLink; більш зручним є

користування вікном **Library: simulink** (рис. 7.3), яке викликається за допомогою контекстного меню. В ньому представлені графічні позначення поділів бібліотеки

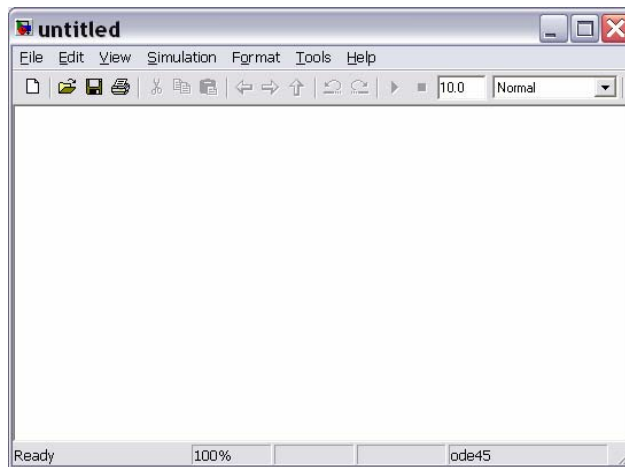


Рис. 7.1. Вікно блок-схеми S-моделі

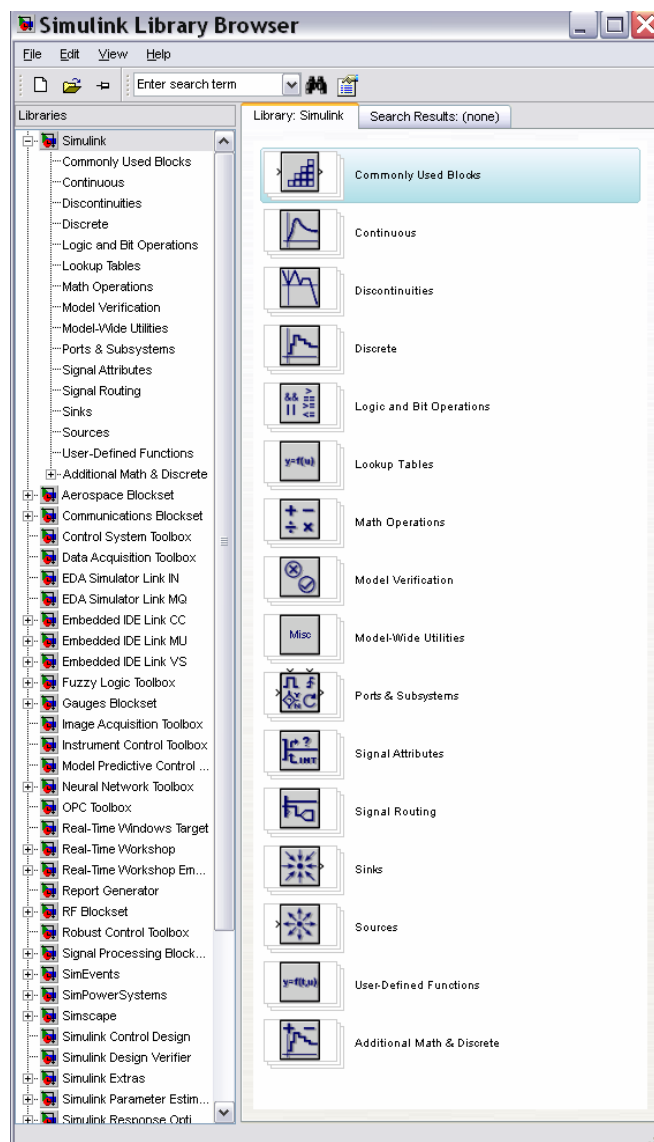


Рис. 7.2. Вікно браузера бібліотеки Simulink

Усі вікна мають подібну структуру і містять рядок меню і робоче поле.

Меню **File** (Файл) містить команди роботи з MDL-файлами, меню **Edit** (редагування) – команди редагування блок-схеми й роботи з бібліотекою, а меню **View** (Подання) – команди змінювання зовнішнього вигляду вікна.

У меню **Simulation** (Моделювання) містяться команди керування моделюванням, а в меню **Format** (Формат) - команди редагування формату (тобто зовнішнього зображення) блоків схеми й блок-схеми в цілому.

Якщо до складу робочої конфігурації MatLAB включений додаток *Real-Time-Workshop*, то меню доповнюється поділом **Tools** (Інструменти), що містить засоби роботи з ним.

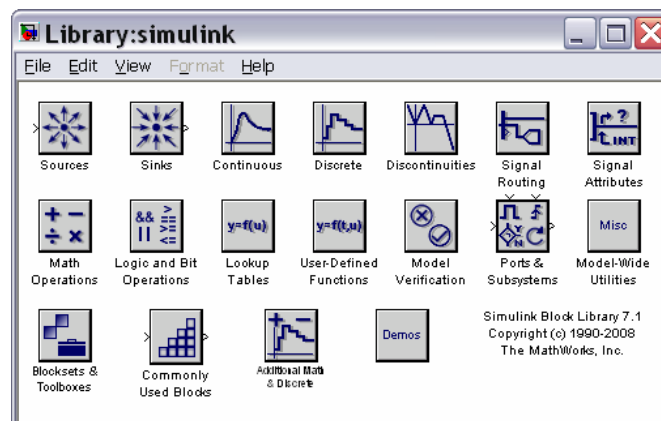


Рис. 7.3. Вікно Library: simulink

7.2. Бібліотека SimuLink

В основі блок-схем S-моделей лежать елементарні блоки, які дозволяють зв'язати блок схему з середовищем Matlab і забезпечити функціонування в ньому S-моделі як програми. Ці блоки містяться у головній бібліотеці пакету *Simulink*, яка має ту саму назву. Бібліотека блоків SimuLink є набором візуальних об'єктів, використовуючи які, можна, з'єднуючи окремі модулі між собою лініями функціонального зв'язку, скласти функціональну блок-схему будь-якого устрою.

Бібліотека блоків складається з 17 поділів. Десять з них є головними і не можуть змінюватися користувачем:

- **Sources** (Джерела);
- **Sinks** (Приймачі);
- **Continuous** (Лінійні неперервні елементи);
- **Discrete** (Дискретні елементи);
- **Discontinuities** (Розривні елементи)
- **Signal Routing** (Пересилання сигналів);
- **Math Operations** (Математичні операції);
- **Logic & Bit Operations** (Логічні та бітові операції);
- **User-defined Functions** (Функції, що визначаються користувачем);
- **Ports & Subsystems** (Порти та підсистеми).

Блоки, що входять у поділ **Sources** (Джерела), призначені для формування сигналів, які забезпечують керування роботою S-моделі в цілому або окремих її частин. Усі блоки-джерела мають по одному виходу і не мають входів.

Блоки, зібрані в поділі **Sinks** (Приймачі), мають тільки входи і не мають виходів. Умовно їх можна поділити на 3 види:

- блоки, які використовуються як оглядові вікна при моделюванні;
- блоки, що забезпечують зберігання проміжних і вихідних результатів моделювання;
- блок керування моделюванням, який дозволяє переривати моделювання при виконанні тих або інших умов.

Поділ **Continuous** (Лінійні неперервні елементи) містить блоки, які можна умовно поділити на дві групи:

- блоки лінійних ланок неперервного часу;
- блоки лінійних стаціонарних ланок із затримкою.

У поділ **Discrete** (Дискретні елементи) входять блоки, за допомогою яких у моделі може бути описане поведіння дискретних систем. Розрізняють два основних типи таких систем: *системи з дискретним часом* і *системи з дискретними станами*. Блоки, що входять у поділ **Discrete** забезпечують моделювання як тих, так і інших.

Поділ **Discontinuities** (Розривні елементи) містить 12 блоків, які реалізують кусково-лінійні залежності.

У поділі **Signal Routing** (Пересилання сигналів) розташовані блоки, що здійснюють об'єднання, роз'єднання сигналів, їх переключення, пересилання тощо.

Блоки поділу **Math Operations** (Математичні операції) реалізують перетворення сигналів, що подаються на входи, за різними типовими математичними залежностями, векторно-матричні операції і перетворення комплексних векторів.

У поділі **Logic & Bit Operations** (Логічні та бітові операції) зосереджені блоки, що здійснюють логічні і бітові операції з сигналами, які поступають до їхнього входу.

Блоки, що входять у склад поділу **User-defined Functions** (Функції, що визначаються користувачем) призначені для утворення користувацьких блоків.

Нерешті, у поділ **Ports & Subsystems** (Порти та підсистеми) входять блоки, які здійснюють зв'язок між моделями різних рівнів ієрархії, а також дозволяють утворити підсистеми, тобто моделі більш низького рівня ієрархії.

Щоб перейти у вікно відповідного поділу бібліотеки, у якому розташовані графічні зображення блоків, достатньо подвійно клацнути мишкою на піктограмі цього поділу

Складання схеми S-моделі полягає в тому, що графічні зображення окремих блоків за допомогою миші перетягуються з вікна поділу бібліотеки у вікно складання схеми, а потім виходи одних блоків у вікні складання з'єднуються із входами інших блоків також за допомогою мишки.

Технологія перетягування зображення блоку така: курсор мишки потрібно встановити на зображенні обраного блоку у вікні поділу бібліотеки, потім натиснути ліву клавішу мишки і, не відпускаючи її, пересунути курсор на поле складання схеми, після чого відпустити клавішу. Аналогічно здійснюються з'єднання у схемі лініями виходів одних блоків із входами інших блоків: курсор мишки підводять до потрібного виходу певного блоку (при цьому курсор має набути форму хрестика), натискають ліву клавішу і, не відпускаючи її, курсор переміщують до потрібного входу іншого блоку, а потім відпускають клавішу. Якщо з'єднання зроблено вірно, на вході останнього блоку виникне зображення чорної затушованої стрілки.

7.2.1. Поділ Sinks (Приймачі)

Після переходу до поділу *Sinks* на екрані виникає вікно цього поділу, зображене на рис. 7.4. З його розгляду випливає, що в цьому поділі розміщено три групи блоків, які не мають виходів, а мають тільки входи:

1) блоки, які при моделюванні відіграють роль оглядових вікон; до них відносяться:

- блок *Scope* з одним входом, який виводить графік залежності величини, яка подається до його входу, від модельного часу;
- блок *XYGraph* із двома входами, який забезпечує побудову графіка залежності однієї модельованої величини (другий зверху вхід) від іншої (перший вхід);
- блок *Display* з одним входом, призначений для відображення чисельних значень вхідної величини;

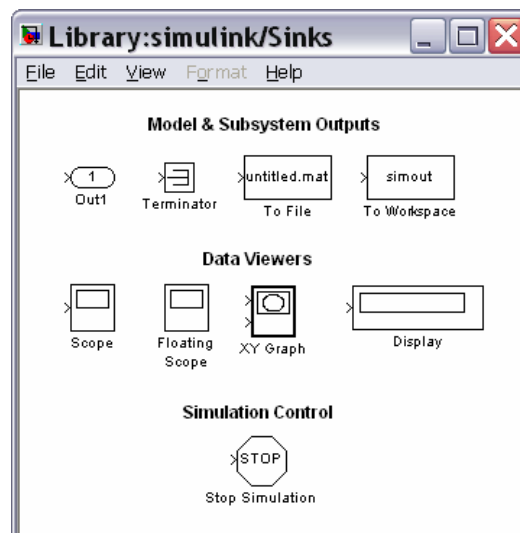


Рис. 7.4. Вміст поділу Sinks

2) блоки для пересилання результатів:

- блок *To File*, який забезпечує зберігання результатів моделювання на диску в MAT-файлі (із розширенням .mat);

- блок *To Workspace*, який зберігає результати в робочому просторі;
- блок-заглушка *Terminator*;
- блок порт виходу *Out*;

3) блок керування моделюванням – *Stop Simulation*, який дозволяє переривати моделювання при виконанні тих або інших умов; блок спрацьовує в тому випадку, коли на його вхід надходить ненульовий сигнал.

Блок Scope

Цей блок дозволяє в процесі моделювання спостерігати на графіку процеси, які відбуваються у модельованій системі і цікавлять дослідника. Він має один вхід, на який подається сигнал, графік залежності якого від часу потрібно вивести у вікні.

Для настроювання параметрів цього блоку потрібно після встановлення зображення блоку у вікно блок-схеми двічі клацнути мишкою на його зображенні. У результаті на екрані з'явиться вікно **Scope** (рис. 7.5). Розмір і пропорції вікна можна змінювати довільно, користуючись мишкою. По горизонтальній осі відкладаються значення модельного часу, а по вертикальній - значення вхідної величини, які відповідають цим моментам часу. Якщо вхідна величина блоку *Scope* є вектором, у вікні будуються графіки змінювання всіх елементів цього вектора, тобто стільки кривих, скільки елементів у вхідному векторі, причому кожна - свого кольору. Одночасно у вікні може відображатися до 30 кривих.

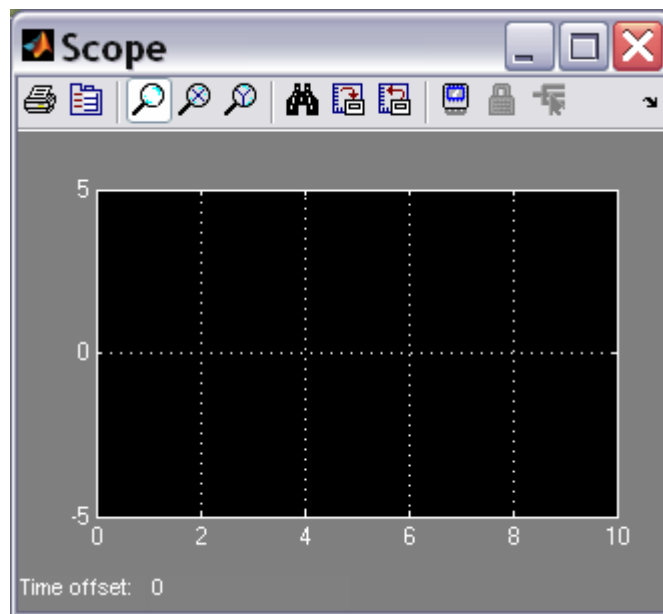


Рис. 7.5. Оглядове вікно Scope

Для керування параметрами вікна в ньому є панель інструментів, що містить 11 піктограм із таким призначенням (зліва направо):

- виведення вмісту вікна на принтер
- виклик вікна настроювання параметрів блоку
- змінювання масштабу одночасно по обох осях графіка;

- змінювання масштабу по горизонтальній осі;
- змінювання масштабу по вертикальній осі;
- автоматичне встановлення оптимального масштабу осей (повний огляд, автошкалювання);
- зберігання встановленого масштабу осей;
- відновлення установок параметрів осей;
- включення холостого під'єднання блоку;
- шлюз селектору сигналів
- селектор сигналів.

Піктограми змінювання масштабу є альтернативними, тобто в кожному момент часу може бути натиснута лише одна з них. Піктограми не активні доки немає графіка у вікні *Scope*. Активними із самого початку є лише перші дві, шоста, сьома і дев'ята піктограми. Щиглик на другій піктограмі призводить до появи діалогового вікна настроювання параметрів '*Scope*' parameters (рис. 7.6).

Це вікно має дві вкладки:

- **General** (Загальні), яка дозволяє встановити параметри осей;
- **Data history** (Встановлювання даних), яку призначено для визначення параметрів подання даних блоку *Scope*.

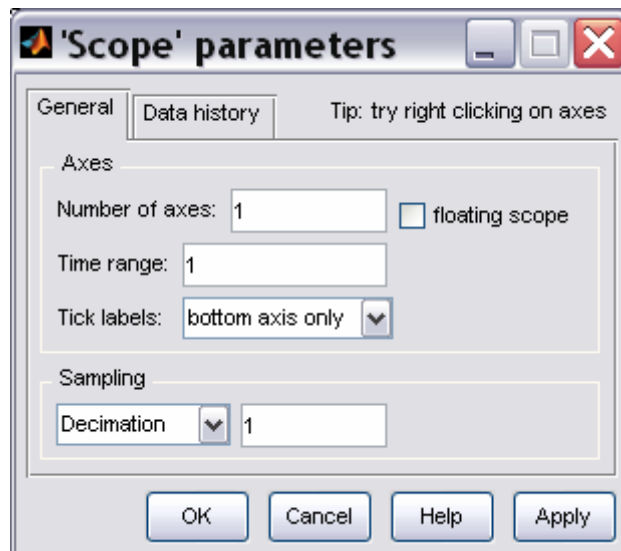


Рис. 7.6. Вікно настроювання блоку *Scope*

У нижній частині вікна розташовані кнопки: *Apply* (Застосувати), *Help* (Довідка), *OK* (Підтвердити установку), *Cancel* (Повернутися назад).

В області *Axes* (Осі) вкладки *General* містять поля введення *Number of axes* (Кількість осей) і *Time range* (Інтервал часу), а також список *Tick labels* (Мітки осей).

У першому полі задається кількість графічних полів у вікні *Scope* (одночасно змінюється кількість входів блоку *Scope*).

У другому полі встановлюється верхня межа модельного часу, що відкладається по осі абсцис. При цьому слід мати на увазі таке. Якщо розмір заданого

інтервалу моделювання (T_m) не перевищує встановленого у цьому полі значення (тобто весь процес уміщується у вікні **Scope**), під графіком у рядку *Time offset* (Зсув за часом) виводиться значення 0. У випадку ж, коли інтервал моделювання перевищує встановлене значення, у вікні **Scope** відображається тільки графік, що відповідає останньому відрізку часу, меншому за розміром, ніж *Time range* і рівному $T_m - n * \text{Time range}$, де n - ціле число. При цьому в рядку *Time offset* виводиться розмір "прихованого" інтервалу часу - $n * \text{Time range}$. Наприклад, якщо значення *Time range* дорівнює 3, а тривалість інтервалу моделювання встановлена 17, то у вікні **Scope** буде виведений графік модельованого процесу за останні 2 одиниці часу, а рядок під графіком буде мати вид: *Time offset: 15*.

За допомогою списку *Tick labels* (Мітки осей) можна задати вид оформлення осей координат в графіках вікна **Scope**. Зазначений список містить три пункти: *all* (все), *bottom axis only* (лише нижньої осі), *none* (нет). В результаті обрання першого з них поділки по осях будуть нанесені вздовж кожної з осей усіх графіків. Обрання другого означає, що поділки по горизонтальних осях графічних полів (якщо їх декілька) за винятком нижньої будуть відсутні. нарешті, якщо обрати третій пункт, то зникнуть поділки по осях графіків і написи на них, графік займе усе поле вікна і останнє прийме вид, показаний на рис. 7.7.

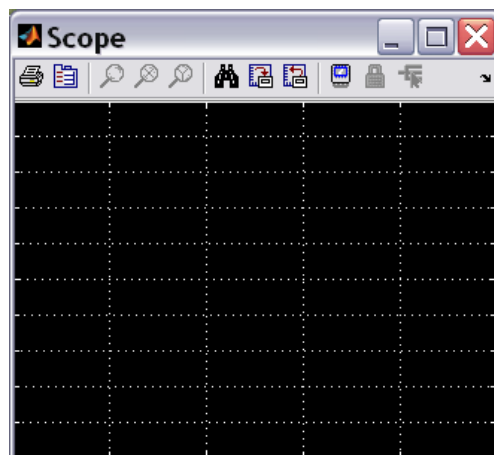


Рис. 7.7. Вікно *Scope* при встановленні *NONE*

Якщо в області *Axes* (Осі) встановити прапорець *Floating scope*, то входи у блок **Scope** будуть відімкнені. У цьому випадку блок відображується як такий, що не має входу, і якщо від був зв'язаний з іншими блоками, ці зв'язки обриваються. Той самий ефект справляє клацання на кнопці з тою самою назвою, що міститься на панелі інструментів блоку.

В області *Sampling* (Дискретизація) міститься список, в якому обраний елемент *Decimation* (Проріджування), і поле, де можна ввести ціле додатне число, яке визначає, через яку кількість проміжків часу (дискретів часу) одержані дані використовуватимуться для побудови графіків в окні **Scope**.

Вкладка *Data History* (Історія даних) вікна '**Scope**' **parameters** (рис. 7.8) дозволяє задати максимальну кількість (починаючи з кінця) елементів ма-

сивів даних, що використовуються для побудови графіків у вікні *Scope* (поле поряд з прапорцем *Limit data points to last* (Максимальна кількість точок даних)).

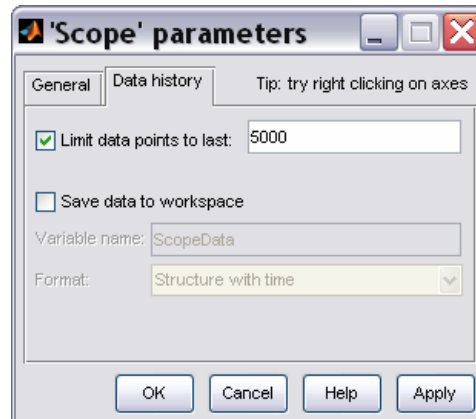


Рис. 7.8. Вкладка Data History

Якщо встановити прапорець *Save data to workspace* (Записати дані у робочий простір), виникне можливість записати у робочий простір дані, які виводяться на графіці вікна *Scope*. При цьому становляться досяжними поле *Variable name* (Ім'я змінної) і список *Format* (Формат). В поле можна ввести ймення змінної, під яким зберігатимуться дані у робочому просторі (за замовчуванням ці дані будуть записані під ім'ям *ScopeData*), а у списку можна обрати один з трьох форматів запису даних: *Array* (Масив, матриця), *Structure* (Структура) або *Structure with time* (Структура з часом).

Продемонструємо роботу блоку на прикладі. Перетягнемо у вікно блок-схеми з вікна поділу *Sources* блок *Sine Wave*, а з вікна поділу *Sinks* – блок *Scope* і з'єднаємо вихід першого блоку зі входом другого. Одержимо схему, показану на рис. 7.9

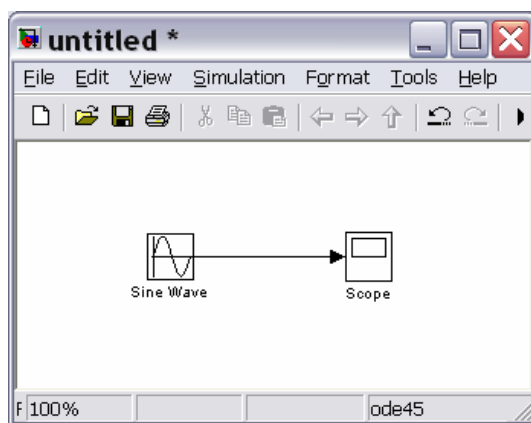


Рис. 7.9. Простіша блок-схема з блоком Scope

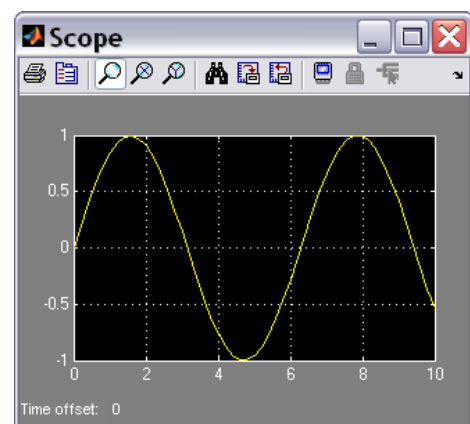


Рис. 7.10. Вікно Scope

Викличемо у вікні цієї блок-схеми команду *Simulation > Start* (Моделювання > Почати), потім подвійно клацнемо на зображенні блоку *Scope*. На екрані виникне вікно *Scope* цього блоку з зображенням графіка змінювання у часі гармонічного сигналу (рис. 7.10).

Блок *XYGraph*

Цей блок також є оглядовим вікном. На відміну від *Scope*, він має два входи: на перший (верхній) подається сигнал, значення якого відкладаються по горизонтальній осі графіка, а на другий (нижній) - по вертикальній осі.

Якщо перетягнути цей блок на поле блок-схеми, а потім на зображенні його подвійно клацнути мишкою, то на екрані виникне вікно налаштування блоку (рис. 7.11), яке дозволяє встановити межі змінювання обох вхідних величин, в яких буде побудований графік залежності другої величини від першої, а також задати дискрет за часом.

Наведемо приклад використання блоку *XYGraph*. Для цього перетягнемо у вікно блок-схеми зображення цього блоку з вікна *Library simulink/Sinks*, а з вікна *Library simulink/Sources* - два блоки-джерела *Clock* і *Sine Wave*. З'єднаємо виходи блоків-джерел із входами блоку *XYGraph*. Одержимо блок-схему, наведену на рис. 7.12.

Перш ніж запуснути процес моделювання цієї схеми, необхідно настроїти блок *XYGraph*, вводячи у діалоговому вікні *Sink Block Parameters: XY Graph* очікувані діапазони змінювання величин x (у розглядуваному випадку – часу) і y (синусоїдального сигналу). Вказано у полях введення x -min, x -max, y -min, y -max відповідно значення 0, 20, -1 і 1, як це відображено на рис. 7.11. Зазначимо, що у випадку використання блоку *Scope* вводити діапазони змінювання величин не потрібно, вони встановлюються автоматично.

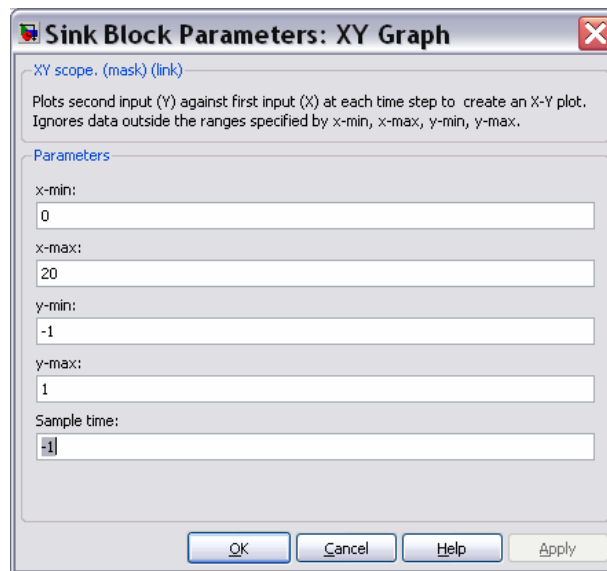


Рис. 7.11. Вікно налаштування блоку *XYGraph*

Якщо тепер вибрати мишкою меню *Simulation* у рядку меню вікна блок-схеми, а в ньому – команду *Start*, то по закінченні розрахунків у вікні блоку *XYGraph* з'явиться зображення, подане на рис. 7.13.

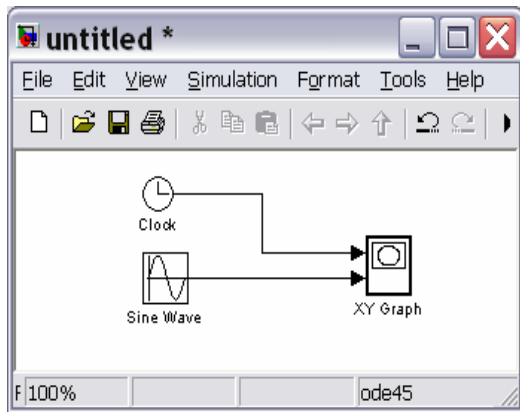


Рис. 7.12. Блок-схема перевірки роботи блоку *XYGraph*

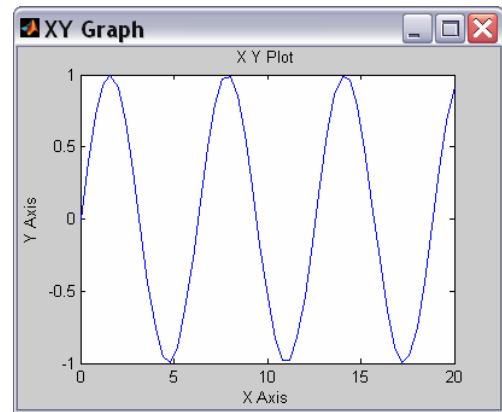


Рис. 7.13. Вікно блоку *XYGraph* з графіком синусоїди

Блок *Display*

Цей блок призначений для виведення на екран чисельних значень величин, що фігурують у блок-схемі.

Блок має 3 параметри настроювання (рис. 7.14). Список *Format* - задає формат виведення чисел; вид формату обирається за допомогою спадного списку, що містить 5 пунктів: *short*, *long*, *short_e*, *long_e*, *bank*. Поле введення *Decimation* дозволяє задати періодичність (у дискретах часу) виведення значень у вікні *Display*. Перемикач *Floating display* дозволяє визначати блок *Display* як блок без входу, обриваючи його зв'язки.

Блок *Display* може використовуватися для виведення як скалярних, так і векторних величин. Якщо відображувана величина є вектором, то вихідне подання блоку змінюється автоматично, про що свідчить поява маленького чорного трикутника в правому нижньому рогу блоку. Для кожного елемента вектора створюється своє міні-вікно, але щоб вони стали видимими, необхідно розтягнути зображення блоку. Для цього слід виділити блок, підвести курсор мишки до одного з його рогів, натиснути ліву клавішу мишки і, не відпускаючи її, розтягнути зображення блоку, поки не зникне чорний трикутник.

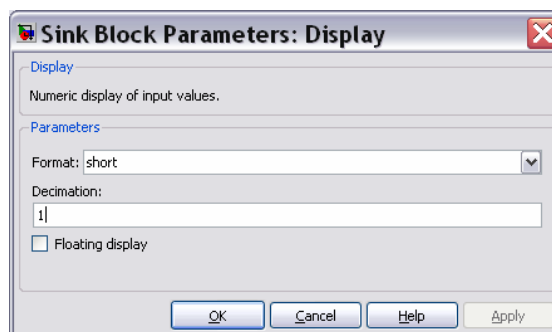


Рис. 7.14. Вікно настроювання блоку *Display*

Для прикладу створимо блок-схему (рис. 7.15) із двох елементів – блоку-джерела *Constant* і блоку-приймача *Display*. Викликавши вікно настроювання блоку *Constant* (рис. 7.16), встановимо в ньому значення константи-вектора, який складається із чотирьох елементів $[pi \ 1e-17 \ 2309 \ -0.00087]$. Викликаючи

вікно настроювання блоку *Display*, встановимо з його допомогою формат виведення чисел *short_e*. Після активізації команди *Start* із меню *Simulation*, розтягуючи зазначеним чином зображення блоку *Display* на блок-схемі, одержимо картину, подану на рис. 7.17.

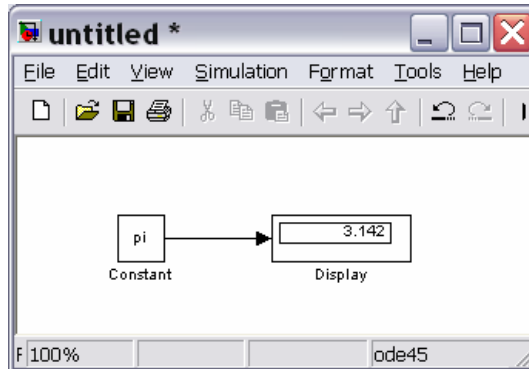


Рис. 7.15. Блок-схема перевірки блоку *Display*

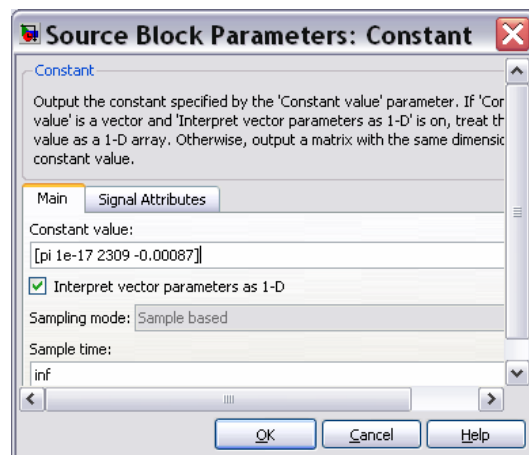


Рис. 7.16. Вікно настроювання блоку *Constant*

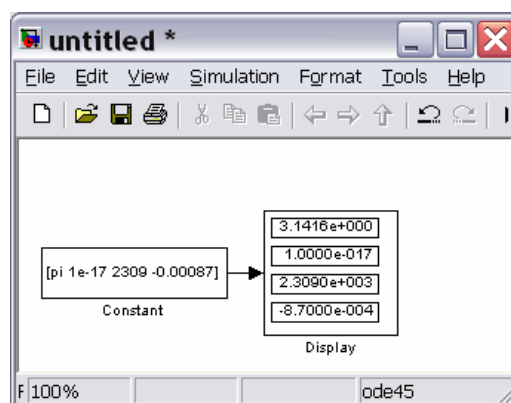


Рис. 7.17. Результат перевірки роботи блоку *Display*

Блок To File

Цей блок забезпечує запис значень величини, поданої на його вхід, у MAT-файл даних для використання їх у наступному в інших S-моделях.

Блок має такі параметри настроювання (див. рис. 7.18):

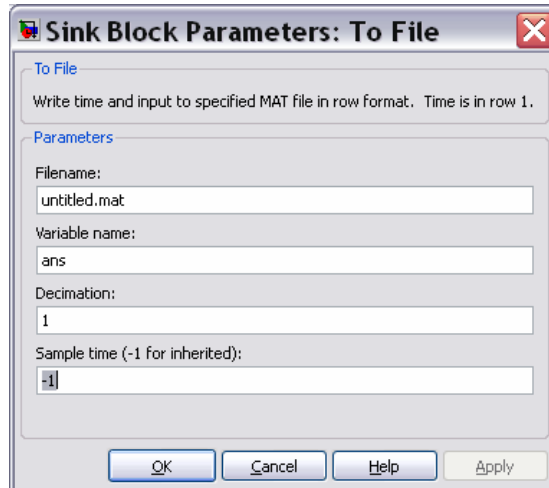


Рис. 7.18. Вікно налаштування блоку To File

Filename – ім'я MAT-файлу, в який записуватимуться значення вхідної величини; за замовчуванням - untitled. mat; ймення файлу виводиться на зображенні блоку в блок-схемі;

Variable name – ім'я змінної, за яким можна буде звертатися до даних, записаних у файлі (для того, щоб переглянути або змінити їх у командному вікні MatLAB); за замовчуванням використовується системне ім'я *ans*;

Decimation – кількість дискретів часу, через яке провадитиметься запис даних у файл;

Sample Time – розмір дискрету часу для даного блоку.

Слід зазначити, що значення даних, що подаються до входу блоку записуються у вихідну змінну (наприклад, *ans*) у такий спосіб: перший рядок матриці утворюють значення відповідних моментів часу; другий рядок містить відповідні значення першого елемента вхідного вектора, третій рядок - значення другого елемента і т. д. В результаті записується матриця розміром $(k+1)*N$, де k – кількість елементів вхідного вектора, а N - кількість точок вимірювання (або кількість моментів часу, у які здійснено вимірювання).

Блок To Workspace

Цей блок призначається для зберігання даних у робочому просторі системи MatLAB. Дані зберігаються у виді матриці розміром $(N*k)$, структура якої відрізняється від структури даних у MAT-файлі тим, що:

- значення величин, що зберігаються, розташовані по стовпцях, а не по рядках;
- не записуються значення модельного часу.

Блок має 4 параметри налаштування (див. рис. 7.19):

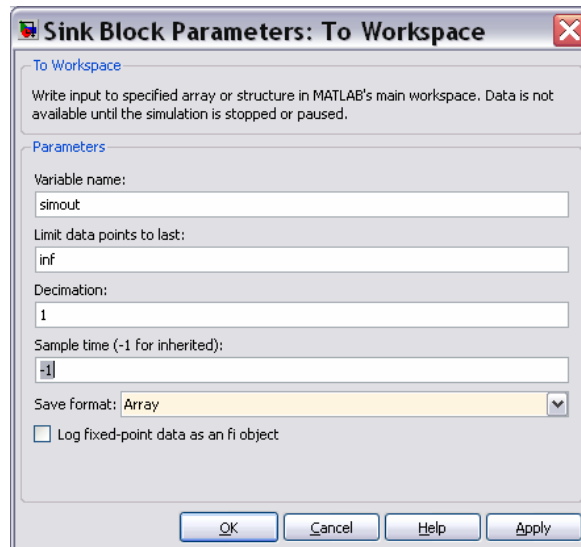


Рис. 7.19. Вікно настроювання блоку *To Workspace*

Variable name – ім'я, із яким дані зберігаються в робочому просторі (за замовчуванням - *simout*);

Limit data points to last – максимально припустима кількість точок, тобто значень даних, що записуються; за замовчуванням вона задається константою *inf*, тобто дані реєструються на всьому інтервалі моделювання;

Decimation і *Sample Time* мають той самий зміст, що і раніше.

7.2.2. Поділ *Sources* (Джерела)

Блоки, що входять до поділу *Sources* (Джерела), призначені для формування сигналів (послідовності числових даних), які при моделюванні забезпечують роботу S-моделі у цілому або окремих її частин. Усі блоки мають по одному виходу і не мають входів.

Після вибору поділу *Sources* бібліотеки SimuLink на екрані з'явиться додаткове вікно, показане на рис. 7.20.

Як бачимо, поділ містить дві групи блоків: *Model & Subsystem Inputs* (Входи моделей і підсистем) та *Signal Generators* (Генератори сигналів).

Як джерела сигналів передбачені такі блоки:

- **Constant** – формує постійну величину (скаляр, вектор або матрицю);
- **Signal Generator** – створює (генерує) неперервний коливальний сигнал однієї із хвильових форм на вибір - синусоїдальний, прямокутний, трикутний чи випадковий;
- **Pulse Generator** – генератор неперервних прямокутних імпульсів;
- **Signal Builder** – побудувач сигналів,
- **Ramp** – створює лінійно висхідний (або спадний) сигнал (сходінку за швидкістю);

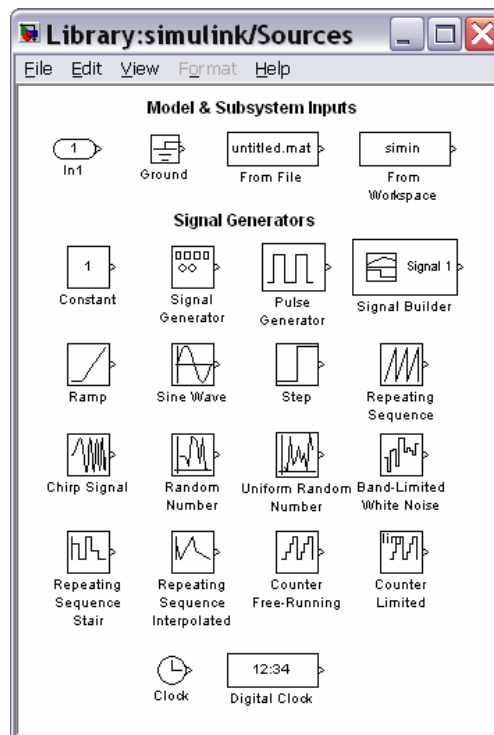


Рис. 7.20. Вміст поділу Sources

- **Sine Wave** – генерує сигнал, що змінюється у часі за гармонічним законом;
- **Step** – генерує сигнал у виді поодинокі сходинок (східчастий сигнал) із заданими параметрами (початку сходинок і її висоти);
- **Repeating Sequence** – генерує періодичну послідовність;
- **Chirp Signal** – генератор гармонічних коливань із частотою, яка лінійно змінюється з часом;
- **Random Number** – джерело сигналу, значення якого є випадковою величиною, розподіленою за нормальним (гауссовим) законом;
- **Uniform Random Number** – джерело сигналу, значення якого є випадковою рівномірно розподіленою величиною;
- **Band-Limited White Noise** – генератор білого шуму з обмеженою смугою частот;
- **Clock** (Годинник) – джерело неперервного сигналу, пропорційного до модельного часу;
- **Digital clock** (Цифровий годинник) – формує дискретний сигнал, пропорційний часові.

У групі блоків **Model & Subsystem Inputs** передбачені 4 блоки, які забезпечують використання в моделі даних, отриманих раніше. Перший з них – **In1** є вхідним портом, завдяки якому можна завести у модель сигнал, одержаний в іншій моделі. Завдяки блоку **Ground** (Земля) можна ввести порожній (нульовий) сигнал у модель. Блок **From File** призначений для введення в S-модель даних, що зберігаються на диску в MAT-файлі. Нарешті блок **From Workspace** забезпечує введення в модель даних безпосередньо з робочого простору MatLAB. Нагадаємо, що структура даних у MAT-файлі є багатовимірним маси-

вом із змінною кількістю рядків, яка визначається кількістю змінних, що реєструються. Елементи першого рядка містять послідовні значення модельного часу, елементи в інших рядках - відповідні окремим елементам записаного вектора значення змінних.

Як і інші блоки бібліотеки SimuLink, блоки-джерела можуть настраюватися користувачем, за винятком блоку *Clock*, робота якого ґрунтується на використанні апаратного таймера комп'ютера.

Блок Constant

Блок призначений для генерування процесів, що є незмінними у часі, тобто мають сталі значення. Він має один параметр настраювання (рис. 7.16) - *Constant value*, який може бути введений і як вектор-рядок з кількох елементів по загальних правилах MatLAB. Приклад його використання наведений раніше при розгляді блоку *Display*.

Блок Signal Generator

Вікно настраювання цього блоку виглядає так, як показано на рис. 7.21. Як видно, у параметри настраювання входять:

- *Wave form* – форма хвилі – дозволяє обрати одну з таких форм періодичного процесу
 - 1) *Sine* – синусоїдальні хвилі;
 - 2) *Square* – прямокутні хвилі;
 - 3) *Sawtooth* – трикутні хвилі;
 - 4) *Random* – випадкові коливання;

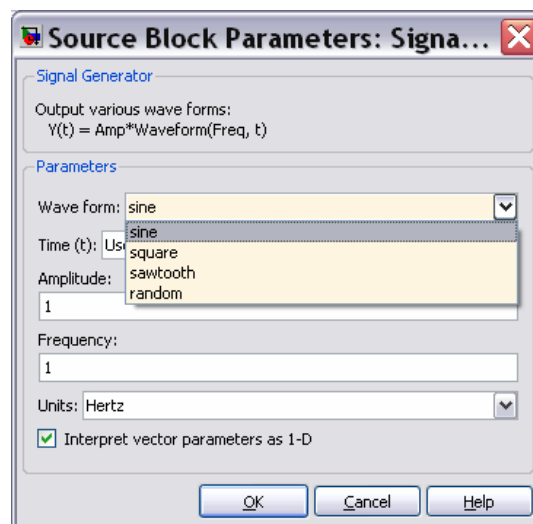


Рис. 7.21. Вікно настраювання блоку *Signal Generator*

- *Amplitude* – визначає значення амплітуди коливань, що генеруються;
- *Frequency* – задає частоту коливань;
- *Units* – дозволяє обрати одну з одиниць виміру частоти за допомогою в спадного меню – *Hertz* (у Герцах) і *Rad/Sec* (у радіанах у секунду).

На рис. 7.22 показано найпростішу блок-схему S-моделі, що складається з блоку *Signal Generator* і оглядового блоку *XY Graph*.

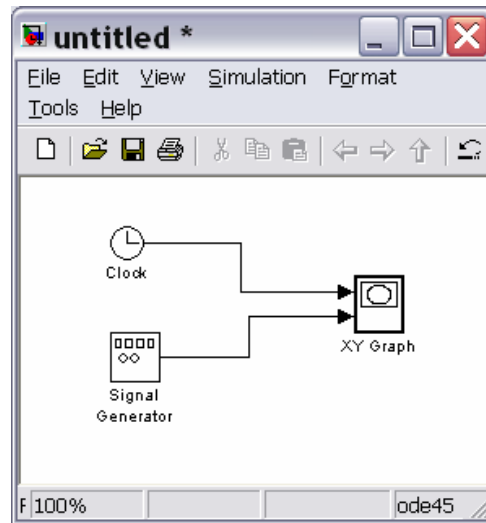


Рис. 7.22. Блок-схема перевірки функціонування блоку *Signal Generator*

На наступному рис. 7.23а відображений вміст блоку *XY Graph* після проведення моделювання при таких параметрах настроювання: вид коливань - *Sine*; амплітуда - 4,5; частота – 0,5 Гц. Рис. 7.23б відбиває результат генерування прямокутної хвилі *Square*.

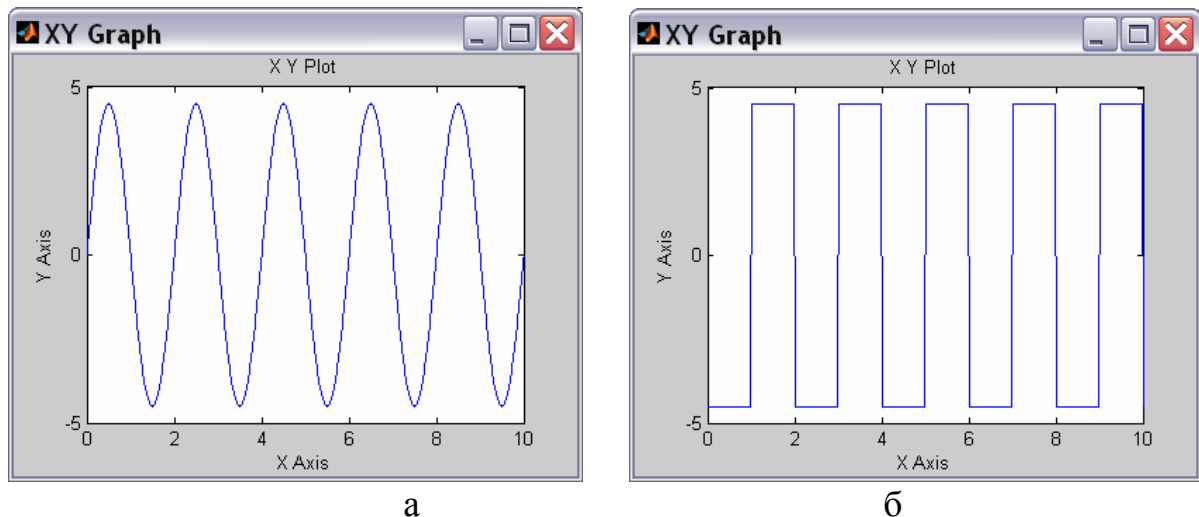


Рис. 7. 23. Результат генерування хвиль *Sine* (а) і *Square* (б)

На рис. 7.24 подані результати, відображені у вікні *XYGraph* у випадку обрання відповідно трикутних і випадкових хвиль при решті тих самих параметрах настроювання.

При обранні пункту *Random* у списку *Wave Form* генерується послідовність даних (сигнал), значення яких рівномірно випадково розподілені у діапазоні, вказаному у параметрі *Amplitude*, а значення моментів часу, у які здійснюються стрибкоподібні змінювання сигналу, відділені один від одного на ве-

личину кроку моделювання, встановлюваного командою *Simulation* > *Configuration Parameters*.

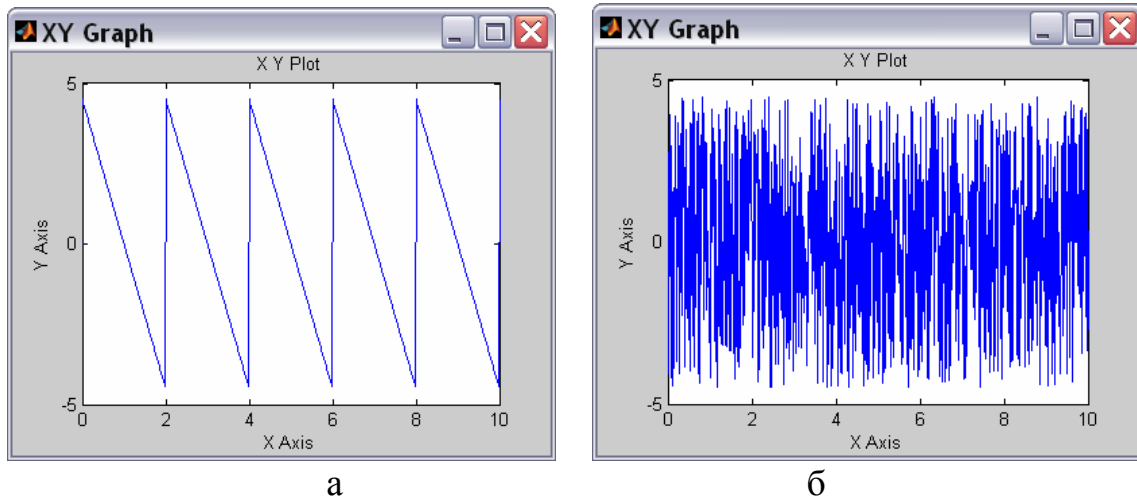


Рис. 7.24. Результат генерування хвиль *Sawtooth* (а) і *Random* (б)

Блок *Pulse Generator*

Блок генерує послідовності прямокутних імпульсів. У число параметрів цього блоку, що настраюються, входять (рис. 7.25):

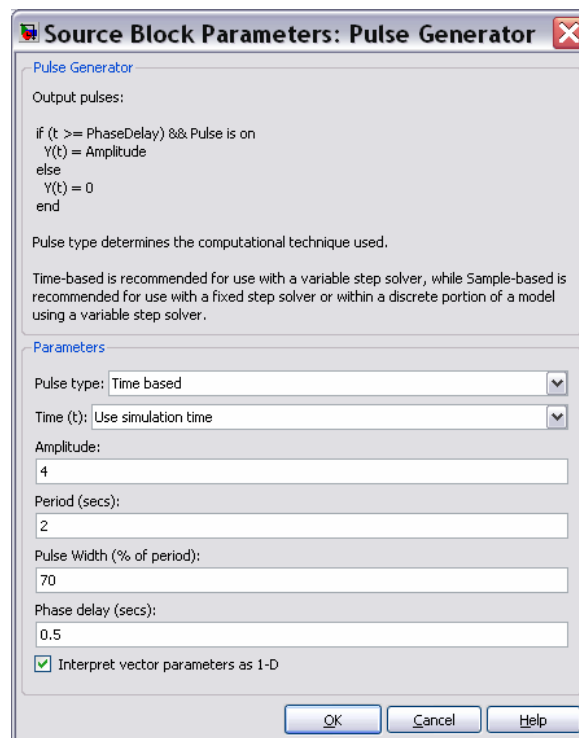


Рис. 7.25. Вікно налаштування блоку *Pulse Generator*

- тип імпульсів (*Pulse Type*); *Time based* (для неперервного сигналу, аргумент – час), *Sample based* (для дискретного часу, аргумент – кількість дискретів часу);
- амплітуда сигналу (*Amplitude*), тобто висота прямокутного імпульсу;

- розмір періоду сигналу (*Period*) у секундах;
- ширина імпульсу (*Pulse width*), у відсотках до періоду;
- розмір затримки імпульсу щодо $t=0$ (*Phase delay*) - у секундах.

Приклад застосування цього блоку при значеннях параметрів, зазначених на рис. 7.25, наведений на рис. 7.26.

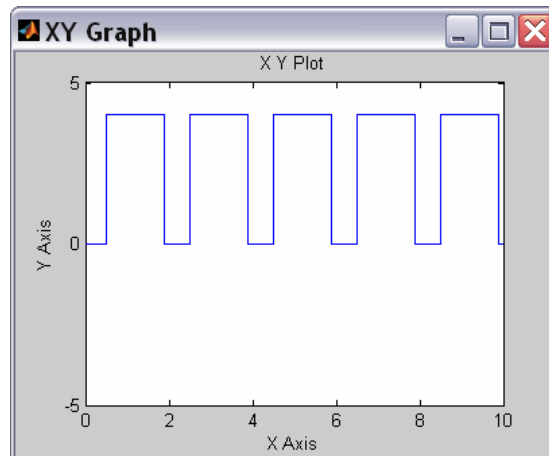


Рис. 7.26. Результат генерування послідовності прямокутних імпульсів блоком *Pulse Generator*

Блок *Ramp*

Цей блок формує неперервно висхідний сигнал і має такі параметри налаштування (рис. 7.27а):

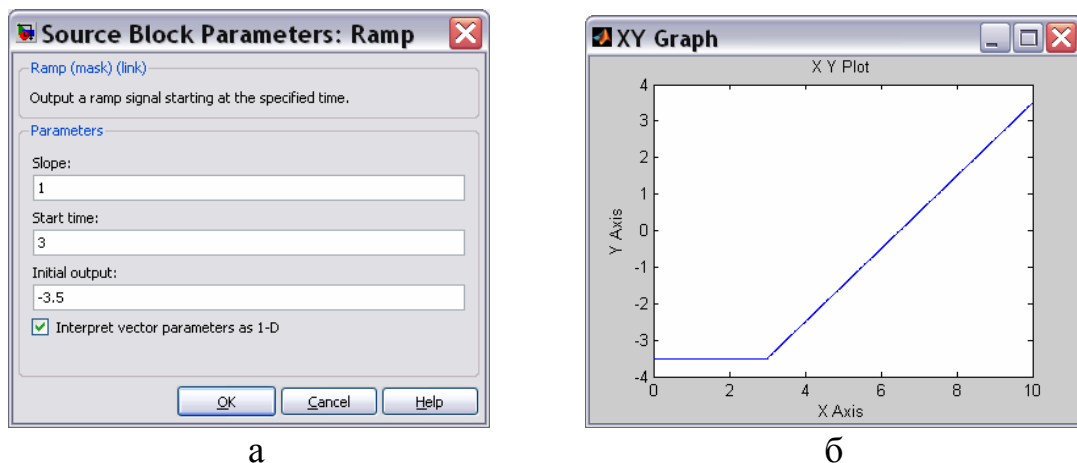


Рис. 7.27. Вікно налаштування (а) і результат роботи (б) блоку *Ramp*

- *Slope* - значення швидкості сходження сигналу (тангенса кута нахилу прямої графіка залежності сигналу від часу);
- *Start time* - час початку сходження сигналу;
- *Initial output* - значення сигналу до моменту початку його сходження.

На рис. 7.27б наведений приклад результату застосування блоку *Ramp* при таких значеннях параметрів: $Slope = 1$; $Start\ time = 3$; $Initial\ output = -3.5$.

Блок *Sine Wave*

Цей блок є генератором гармонічно змінюваного з часом сигналу.

Блок *Sine Wave* має такі настройки (рис. 7.28):

- *Sine Type* - тип синусоїдальної хвилі: *Time based* - для неперервного сигналу (аргумент – час); *Sample based* - для дискретного часу, (аргумент – кількість дискретів часу);
- *Amplitude* - амплітуда синусоїдального сигналу;
- *Bias* – зміщення (стала складова сигналу, середнє його значення);
- *Frequency (rad/sec)* - частота коливань у радіанах у секунду;
- *Phase (rad)* - початкова фаза в радіанах;
- *Sample time* - визначає величину дискрету часу для цього блоку.

Результат застосування блоку (при значеннях параметрів настроювання: амплітуда – 2,5; стала складова сигналу – (-1); частота - 1 радіан у секунду і початкова фаза – $\pi/2$ радіан) показаний на рис. 7.29.

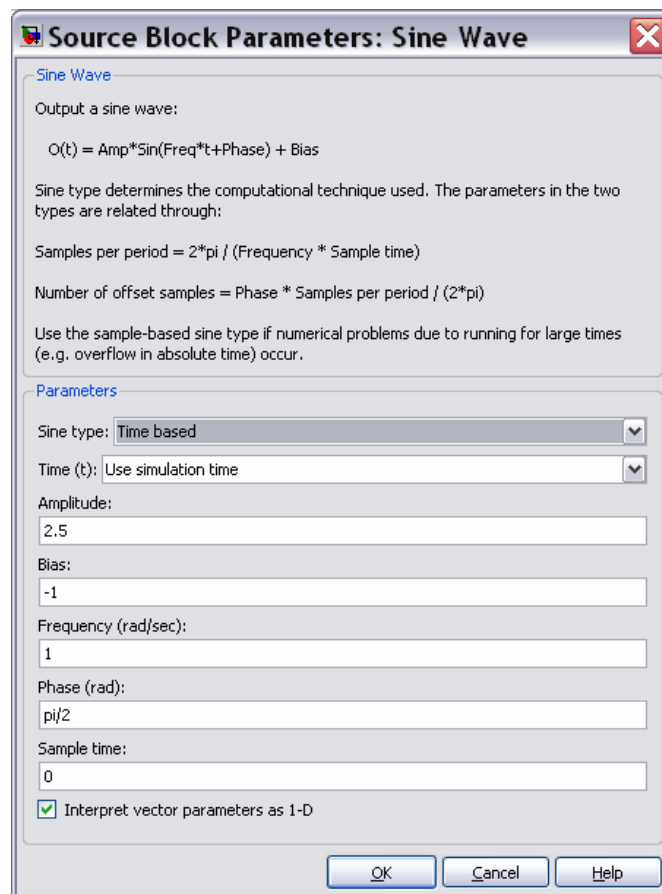
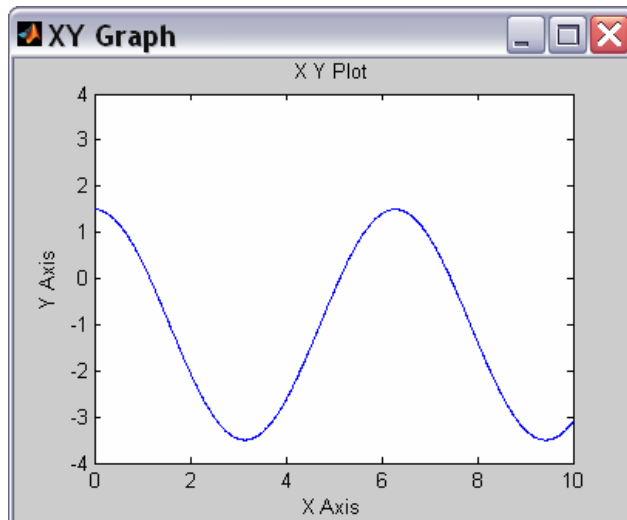


Рис. 7.28. Вікно настроювання блоку *Sine Wave*

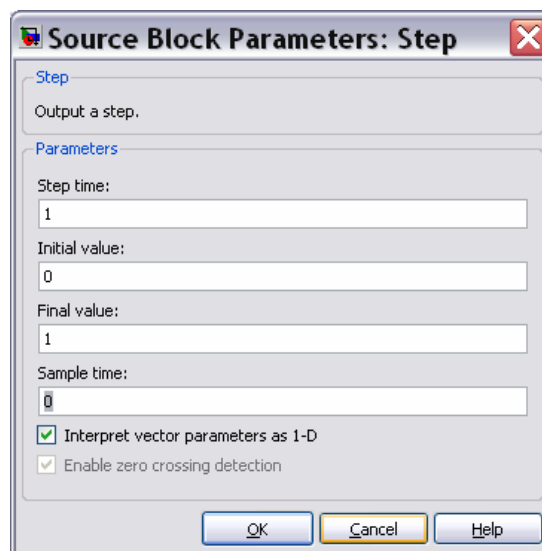
Рис. 7.29. Результат роботи блоку *Sine Wave*

Блок *Step*

Блок забезпечує формування керуючого сигналу у формі сходинок (або, як говорять, – східчастого сигналу).

Блок має 3 параметри настроювання (рис. 7.30):

- *Step time* - час початку сходинок (момент стрибка сигналу) - визначає момент часу, у який відбувається стрибкоподібне змінювання сигналу; за замовчуванням приймається рівним 1;
- *Initial value* - початкове значення - задає рівень сигналу до стрибка; значення за замовчуванням – 0;
- *Final value* - кінцеве значення - задає рівень сигналу після стрибка; значення його за замовчуванням – 1.

Рис. 7.30. Вікно настроювання блоку *Step*

Якщо встановити такі параметри настроювання блоку: *Step time* – 3,5, *Initial value* – (-2), *Final value* – 3, то після активізації моделювання (*Simulation/Start*) одержимо у вікні *Scope* картину, подану на рис. 7.31.

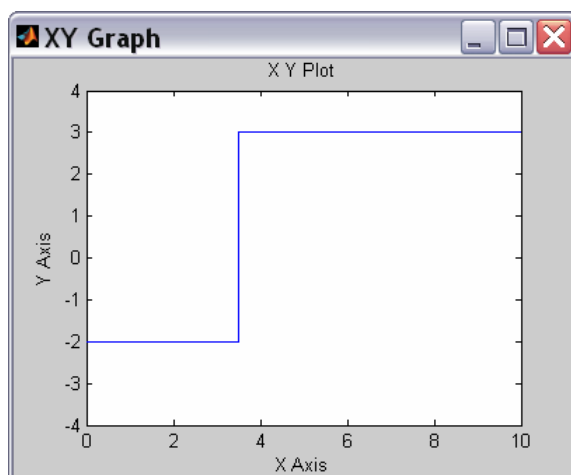


Рис. 7.31. Результат генерування ступінчастого сигналу

Блок *Repeating Sequence*

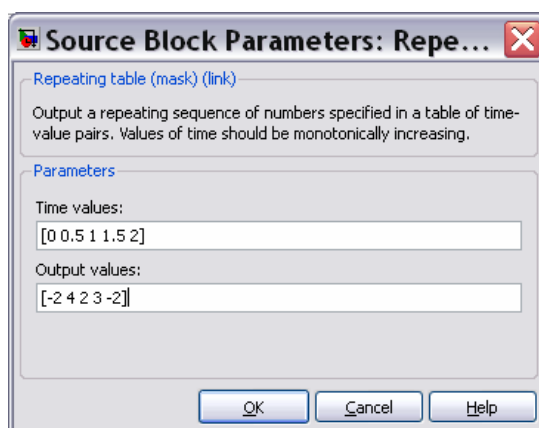
Цей блок являє собою генератор періодичних коливань, хвиля яких складається з відрізків прямих. У вікні містить дві настройки (рис. 7.32):

- *Time values* - вектор значень часу, в які задані значення вихідної величини;

- *Output values* - вектор значень вихідної величини, які вона повинна прийняти в зазначені в першому векторі відповідні моменти часу.

Блок забезпечує генерування коливань із періодом, рівним різниці між останнім значенням вектора *Time values* і значенням першого його елемента. Форма хвилі усередині періоду є ламаною, що проходить через точки із зазначеними у векторах *Time values* і *Output values* координатами.

Як приклад на рис. 7.33 наведене зображення процесу, згенерованого блоком *Repeating Sequence* при параметрах настроювання, зазначених на рис. 7.32.

Рис. 7.32. Вікно настроювання блоку *Repeating Sequence*

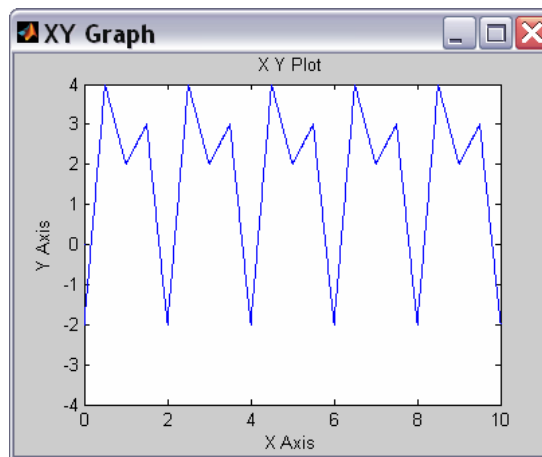


Рис. 7.33. Сигнал, згенерований блоком *Repeating Sequence*

Блок *Chirp Signal*

Цей блок генерує синусоїдальний сигнал одиничної амплітуди змінної частоти, причому значення частоти коливань змінюється з часом за лінійним законом. Відповідно до цього в ньому передбачені такі параметри налаштування (рис. 7.34):

- *Initial frequency (Hz)* - початкове значення (при $t=0$) частоти в герцах;
- *Target time (secs)* - другий (додатний) момент часу (у секундах);
- *Frequency at target time (Hz)* - значення частоти в цей другий момент часу.

Рис. 7.35 демонструє результат використання блоку при параметрах, зазначених на рис. 7.34.

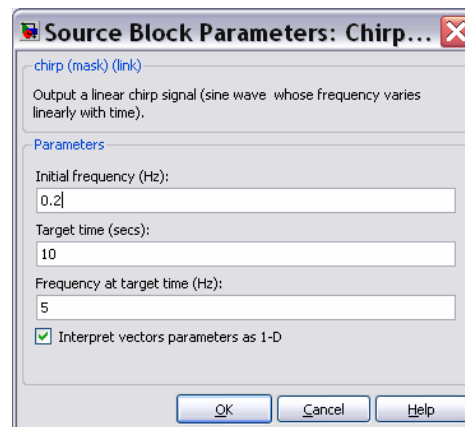


Рис. 7.34. Вікно налаштування блоку *Chirp Signal*

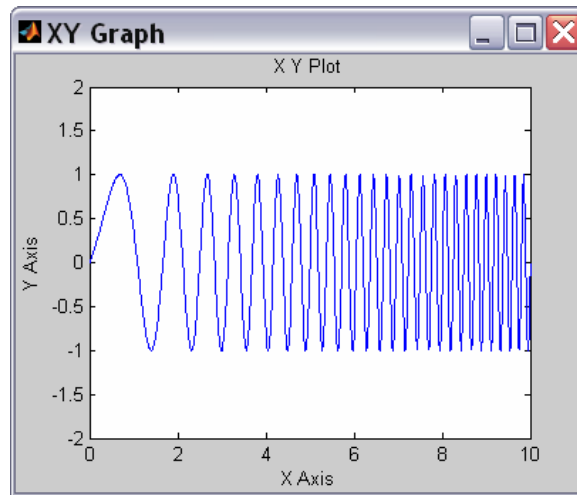


Рис. 7.35. Сигнал, згенерований блоком *Chirp Signal*

Блоки, що генерують випадкові процеси

Блок ***Random Number*** забезпечує формування сигналів, значення яких в окремі моменти часу є випадковою величиною, розподіленою за нормальним (гауссовим) законом із заданими параметрами.

Блок має чотири параметри настроювання (рис. 7.36). Перші два - *Mean* і *Variance* - є параметрами нормального закону (середнє й середньоквадратичне відхилення від цього середнього), третій - *Initial seed* - задає початкове значення бази для ініціалізації генератора послідовності випадкових чисел. При фіксованому значенні цього параметра генератор завжди виробляє ту саму послідовність. Четвертий параметр (*Sample time*), як і раніше, задає величину дискрету часу.

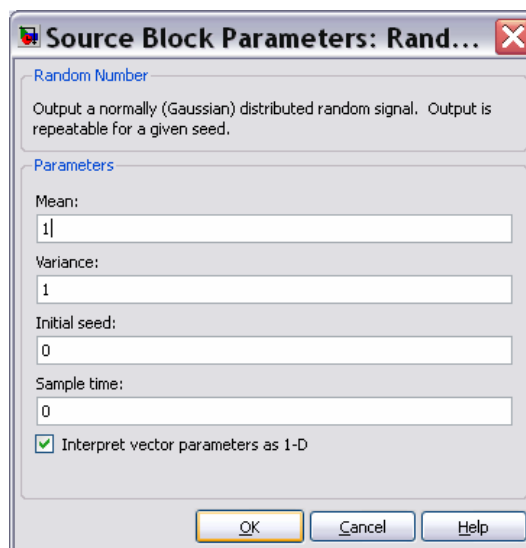


Рис. 7.36. Вікно настроювання блоку *Random Number*

Блок ***Uniform Random Number*** блок формує сигнали, значення яких в окремі моменти часу є випадковою величиною, що рівномірно розподілена в заданому інтервалі. У число параметрів настроювання блоку входять:

- *Minimum* - нижня межа випадкової величини;

- *Maximum* - верхня межа;
- *Initial seed* - початкове значення бази генератора випадкових чисел;
- *Sample time* - дискрет часу.

Блок ***Band-Limited White Noise*** формує процес у виді частотно-обмеженого білого шуму. Параметри настроювання в нього такі:

- *Noise power* - значення потужності білого шуму;
- *Sample time* - значення дискрету часу (визначає верхнє значення частоти процесу);
- *Seed* - початкове значення бази генератора випадкової величини.

Сформуємо блок-схему, яка ілюструє роботи блоків. Для того, щоб у вікні *Scope* можна було розмістити три графіки, кожен з яких відбивав би окремий процес, необхідно викликати команду *Scope* > 'Scope' parameters > General і встановити у полі *Number of axes* кількість осей – 3. Вид блоку *Scope* на блок-схемі при цьому зміниться, на ньому виникнуть зображення трьох входів. У списку *Tick Labels* встановимо значення *all*. Після цього над кожним з трьох графіків, які виводяться, можна буде проставити заголовки. Тексти заголовків розміщуються на лініях, що ведуть до входів блоку *Scope* (див. рис. 7.37)

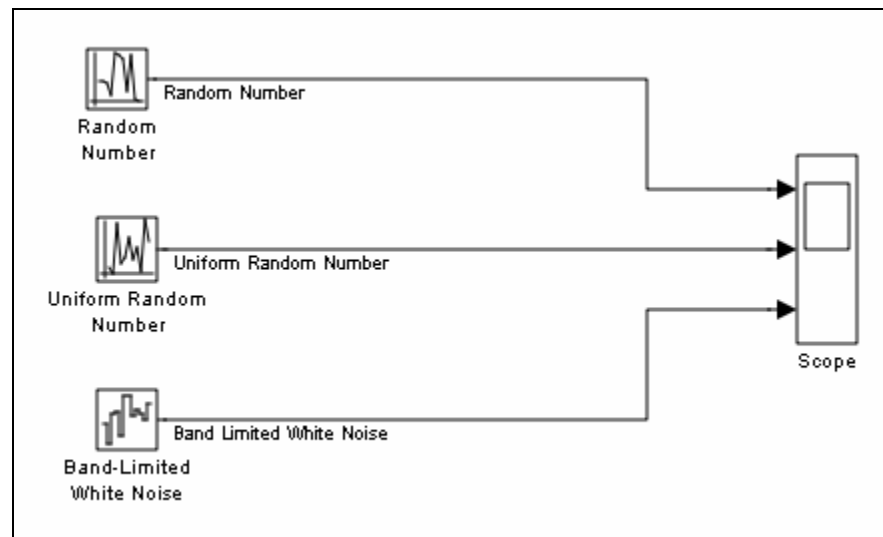


Рис. 7.37. Блок-схема перевірки роботи генераторів випадкових процесів

Перед запуском моделі у вікні блок-схеми викличемо команду *Simulation* > *Configuration parameters* > *Solver*, у віконці якого зі спадним списком оберемо значення *discrete (no continuous state)*. У тому самому вікні у віконці з написом *Fixed step size* (Розмір фіксованого кроку) встановимо 0,05.

У вікнах настроювання усіх трьох блоків встановимо дискрет часу (*Sample time*), рівним 0,1.

Здійснюючи тепер моделювання, одержимо процеси, подані на рис. 7.38.

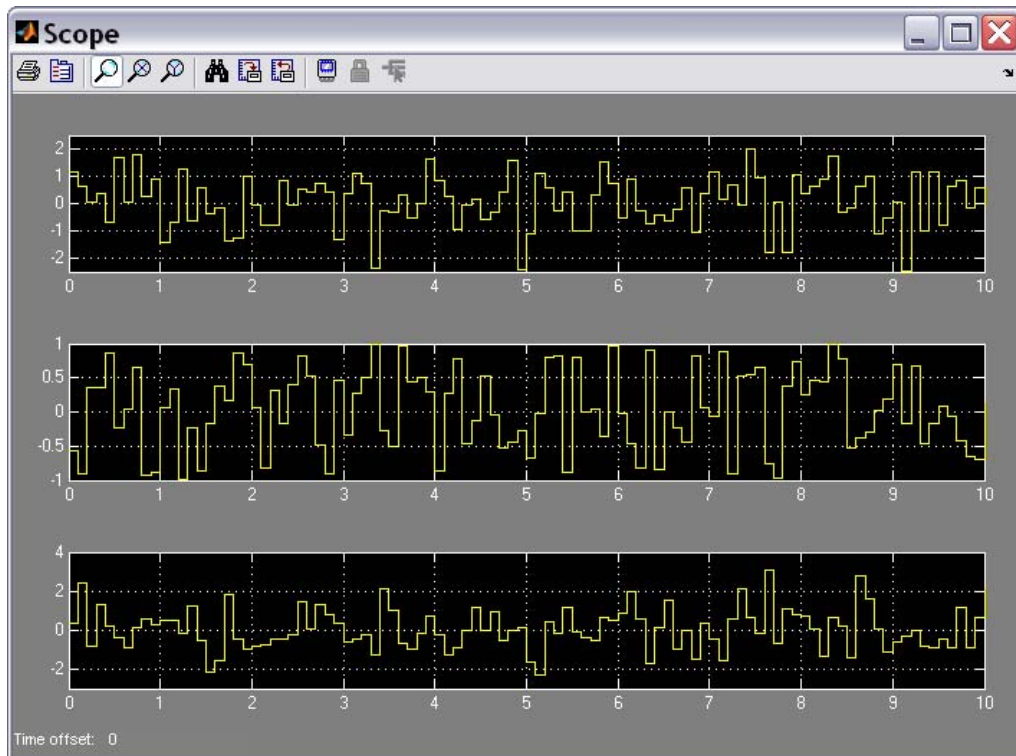


Рис. 7.38. Результати роботи генераторів випадкових процесів при $Sample\ time=0,1$

Як бачимо, модельовані процеси зберігають незмінні значення всередині інтервалу часу $Sample\ time$ (Дискрета часу). Змінювання значень процесу здійснюється стрибкоподібно на межі сусідніх дискретів часу.

Примітка. Слід відрізнявати крок змінювання модельного часу, який встановлюється при проведенні моделювання у вікні *Configuration parameters* і визначає інтервали часу, через які здійснюються обчислення окремих станів системи, що моделюється, і параметр $Sample\ time$ (Дискрет часу), який визначає інтервал часу, всередині якого вихідна величина блоку не змінює свого значення.

Варто зазначити, що блок ***Band-Limited White Noise*** суттєво відрізняється від перших двох блоків-генераторів. Для останніх встановлення значення параметра $Sample\ time$, відмінного від нуля, не є обов'язковим. У блоці ***Band-Limited White Noise*** цей параметр визначає найбільшу частоту у спектрі вихідного сигналу, вона дорівнює величині (у герцах), обернену значенню параметра $Sample\ time$, тому останній не може бути рівний нулеві.

На рис. 7.39 показаний результат моделювання перших двох блоків при значенні параметра $Sample\ time$, рівним нулю. У цьому випадку змінювання величини випадкового процесу здійснюється вже на стику кроків моделювання (0,05), величина яких встановлюється за допомогою команди *Simulation > Configuration parameters* у віконці з написом *Fixed step size* (Розмір фіксованого кроку).

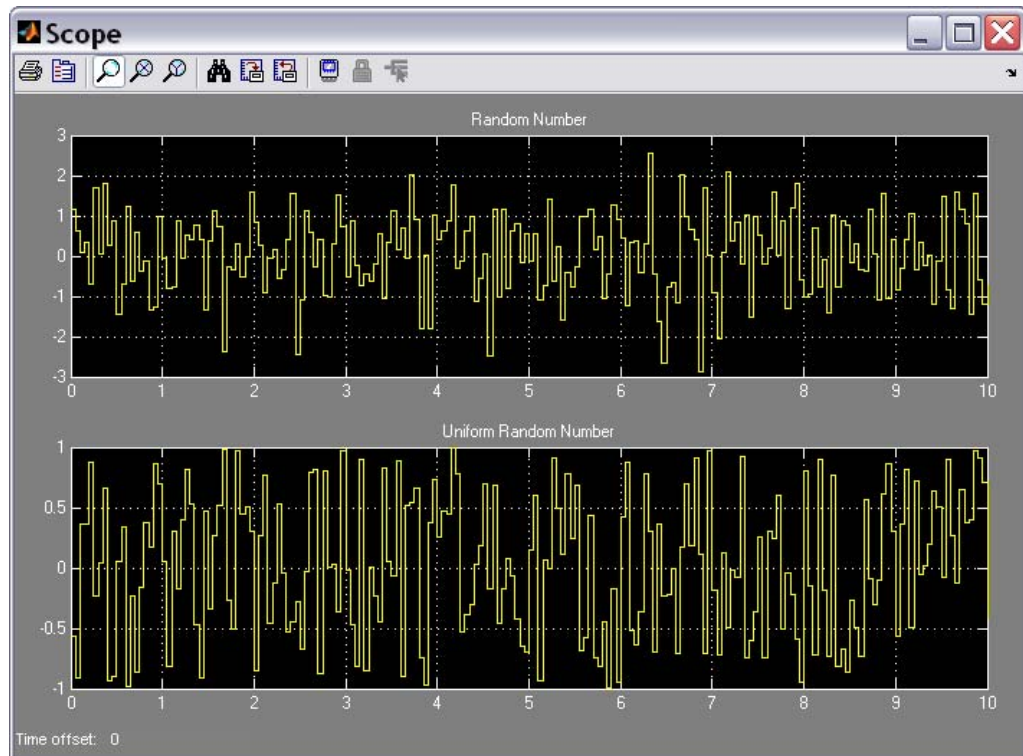


Рис. 7.39. Результати роботи генераторів випадкових процесів при $Sample\ time=0$

7.2.3. Поділ *Continuous* (Неперервні елементи)

Цей поділ бібліотеки містить наступні групи блоків (рис. 3.40):

- *Continuous-Time Linear Systems* (Лінійні системи неперервного часу);
- *Continuous-Time Delays* (Затримки неперервного часу).

Перша група складається з блоків:

- ***Integrator*** – ідеальна інтегруюча ланка (інтегратор);
- ***Derivative*** – ідеальна диференціююча ланка;
- ***State-Space*** – визначення лінійної ланки через завдання чотирьох матриць її простору станів;
- ***Transfer Fcn*** – визначення лінійної ланки через завдання її передатної функції;
- ***Zero-Pole*** – завдання ланки через указівку векторів значень його полюсів і нулів, а також значення коефіцієнта передачі;

Використання більшості блоків-ланок є досить прозорим для тих, хто знайомий з основами теорії автоматичного керування.

Блок *Integrator*

Блок здійснює інтегрування в неперервному часі вхідної величини. Він має наступні параметри настроювання (рис. 7.41):

- підключення додаткового керуючого сигналу (*External reset*);
- визначення джерела (внутрішнє чи зовнішнє) встановлювання початкового значення вихідного сигналу (*Initial condition source*);

- початкове значення вихідної величини (*Initial condition*); значення вводиться в рядку редагування або як числова константа, або у вигляді виразу, що обчислюється;
- прапорець *Limit output* (*Обмеження вихідного значення*) визначає, чи будуть використовуватися наступні 2 параметри настроювання;

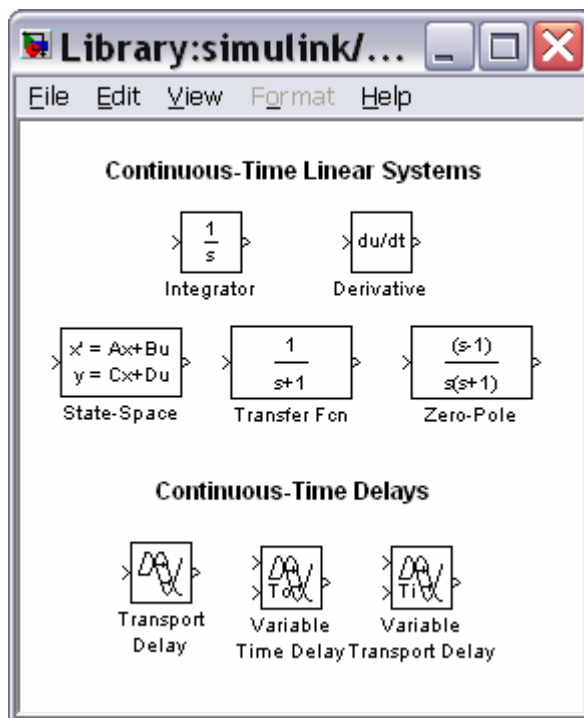


Рис. 7.40. Вміст поділу Continuous

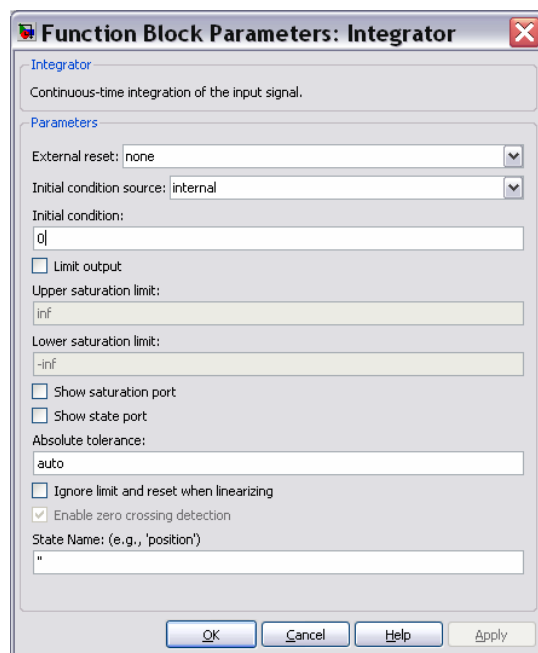


Рис. 7.41. Вікно настроювання блоку Integrator

- верхнє граничне значення вихідної величини (*Upper saturation limit*); за замовчуванням - не обмежене (*inf*);

- нижнє граничне значення вихідної величини (*Lower saturation limit*); за замовчуванням параметр має значення (*-inf*);
- прапорець *Показати порт насичення* (*Show saturation port*);
- прапорець *Показати порт стану* (*Show state port*);
- припустима гранична величина абсолютної похибки (*Absolute tolerance*).

Параметр *External reset* може приймати такі значення (див. його спадне меню): *none* – додатковий керуючий сигнал не використовується; *rising* – для керування використовується висхідний сигнал; *falling* – для керування використовується спадний сигнал; *either* – для керування використовується і висхідний і спадний сигнал.

Параметр *Initial condition source* приймає одне із двох значень:

internal – використовується внутрішнє встановлювання початкового значення вихідної величини;

external – встановлення початкових умов здійснюватиметься ззовні.

Якщо обрані користувачем значення цих двох параметрів припускають наявність додаткових вхідних сигналів, то на графічному зображенні блоку виникають додаткові вхідні порти (після натискання кнопки *Apply* у вікні налаштувань блоку). Якщо прапорець *Limit output* встановлений, то при переході вихідного значення інтегратора через верхню або нижню межу на додатковому виході блоку (*saturation port*) формується одиничний сигнал. Щоб цей сигнал можна було використовувати для керування роботою S-моделі, прапорець *Show saturation port* має бути включений. При цьому на графічному зображенні блоку з'являється позначення нового вихідного порту на правій стороні зображення блоку-інтегратора.

Встановлення прапорця *Show state port* також приводить до появи додаткового виходу *state port* блоку (він виникає звичайно, на нижній стороні зображення блоку). Сигнал, який подається на цей порт, збігається з головним вихідним сигналом, але, на відміну від нього, може бути використаний тільки для переривання алгебричного циклу або для узгодження стану підсистем моделі.

Блок *State-Space* (простір станів)

Блок *State-Space* забезпечує введення лінійної стаціонарної ланки перетворення вхідного сигналу у виді матриць у просторі станів. Його вікно налаштування (рис. 7.42) містить 7 параметрів налаштування:

- матриця *A* системи зв'язку похідних від змінних стану з самими змінними стану;
- матриця *B* зв'язку похідних від змінних стану з вхідним сигналом блоку;
- матриця *C* зв'язку вихідного сигналу зі змінними стану;
- матриця *D* зв'язку вихідного сигналу з вхідним сигналом;
- вектор *Initial conditions* початкових значень вектору змінних стану;

- вектор Absolute tolerance максимально припустимих похибок обчислення кожної зі змінних стану; за замовчуванням встановлюється автоматично;
- State Name – символічний рядок, що містить ймення стану системи.



Рис. 7.46. Вікно налаштування блоку State-Space

Блок *Transfer Fcn* (передатна функція)

Дозволяє ввести лінійну ланку через завдання її передатної функції. Вікно налаштування показано на рис. 7.43 і містить два основні параметри:

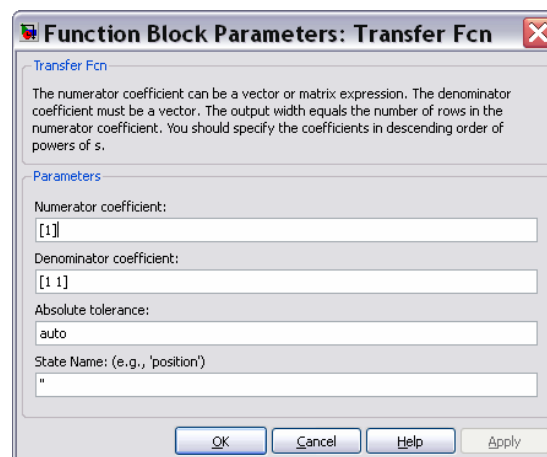


Рис. 7.43. Вікно налаштування блоку Transfer Fcn

- *Numerator coefficients* – вектор значень коефіцієнтів чисельника передатної функції;
- *Denominator coefficients* – вектор значень коефіцієнтів знаменника передатної функції.

Блок *Zero-Pole* (нули - полюси)

Блок *Zero-Pole* дозволяє ввести лінійну ланку через завдання нулів і полюсів її передатної функції.

Головні параметри настроювання блоку є наступними (рис. 7.44):

- *Zeros* – вектор нулів передатної функції (коренів поліному її чисельника);
- *Poles* – вектор полюсів передатної функції (коренів поліному її знаменника);
- *Gain* – вектор коефіцієнтів підсилення.

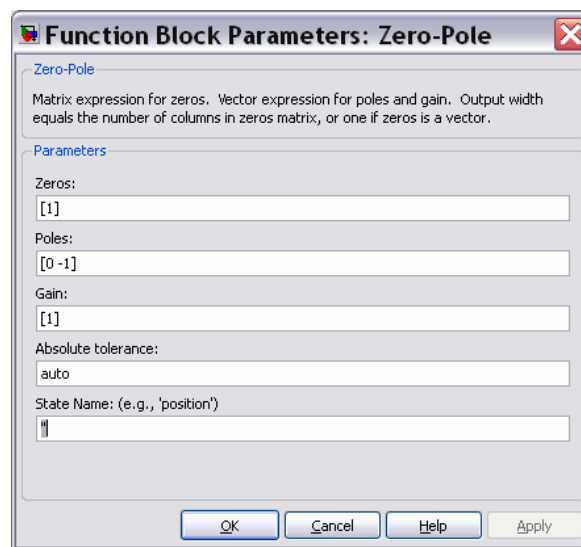


Рис. 7.44. Вікно настроювання блоку *Zero-Pole*

Друга група блоків здійснює затримку неперервних сигналів, що поступають на їхній вхід.

Блок *Transport Delay* забезпечує затримку сигналу на задану кількість кроків модельного часу, причому необов'язково ціле. Настроювання блоку відбувається по трьох параметрах:

Time delay (Час затримки) - кількість кроків модельного часу, на який слід затримати сигнал; може вводитися або в числовій формі, або у формі виразу, що обчислюється;

Initial input (Початкове значення входу) - за замовчуванням дорівнює 0;

Initial buffer size (Початковий розмір буфера) - обсяг пам'яті (у байтах), що виділяється в робочому просторі MatLAB для збереження параметрів затриманого сигналу; повинно бути кратним до 8 (за умовчанням - 1024).

Блок *Variable Transport Delay* дозволяє задавати керовану ззовні величину затримки. З цією метою блок має додатковий вхід. Подаваний на нього сигнал визначає тривалість затримки.

7.2.4. Поділ *Discrete* (Дискретні елементи)

Раніше розглянуті розділи бібліотеки дозволяють формувати неперервну динамічну систему. Розділ *Discrete* містить елементи (блоки), властиві тільки

дискретним системам, а також такі, що перетворюють неперервну систему в дискретну (рис. 7.45). Ці блоки поділені на дві групи

- *Discrete-Time Linear Systems* (Лінійні системи дискретного часу);
- *Sample & Hold Delays* (Затримки дискретного часу).

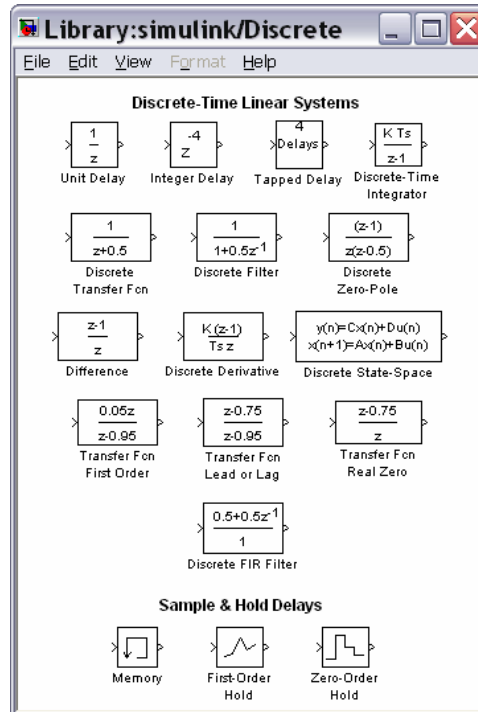


Рис. 7.45. Вміст поділу Discrete

У першу групу входять блоки:

- **Unit Delay, Integer Delay, Tapped Delay** – блоки, які використовуються для затримки сигналу;
- **Discrete-Time Integrator** – дискретний інтегратор;
- **Discrete Transfer Fcn** – блок завдання лінійної дискретної ланки через дискретну передатну дробово-раціональну функцію щодо z ;
- **Discrete Filter** – блок завдання дискретної ланки через дискретну передатну дробово-раціональну функцію щодо $1/z$;
- **Discrete Zero-Pole** – блок завдання дискретної ланки через задання значень нулів і полюсів дискретної передатної функції щодо $1/z$.
- **Discrete Derivative** – формує дискретну похідну вхідного сигналу
- **Discrete State-Space** – блок завдання дискретної лінійної ланки матрицями її стану;
- **Transfer Fcn First Order, Transfer Fcn Lead or Lag, Transfer Fcn Real Zero** – блоки, що формують дискретні ланки з спеціальними передатними функціями;
- **Discrete FIR Filter** – формує дискретний КІХ-фільтр.

Другу групу складають блоки:

- **Memory** – використовується для затримки сигналу на один крок модельного часу;

- **First-Order Hold** - екстраполятор першого порядку;
- **Zero-Order Hold** - екстраполятор нульового порядку;

Блок **Unit Delay** забезпечує затримку вхідного сигналу на задане число кроків модельного часу. Параметрами настроювання для цього блоку є: початкове значення сигналу (*Initial condition*) і час затримки (*Sample time*), який задається кількістю кроків модельного часу.

Блок **Discrete-Time Integrator** виконує чисельне інтегрування вхідного сигналу. Більшість параметрів настроювання цього блоку збігаються з параметрами блоку **Integrator** розділу **Linear**. Відмінності полягають у наступному. У блоці дискретного інтегратора є додатковий параметр - метод чисельного інтегрування (*Integrator method*). За допомогою спадного меню можна вибрати один із трьох методів: прямий метод Ейлера (лівих прямокутників); зворотний метод Ейлера (правих прямокутників); метод трапецій. Друга відмінність - замість параметра *Absolute tolerance* уведений параметр *Sample time*, який задає крок інтегрування в одиницях кроків модельного часу.

Блок **Memory (Пам'ять)** виконує затримку сигналу тільки на один крок модельного часу. Блок має два параметри настроювання: *Initial condition (Початкова умова)* задає значення вхідного сигналу в початковий момент часу; прапорець *Inherit sample time (Спадкування кроку часу)* дозволяє вибрати величину проміжку часу, на який буде здійснюватися затримка сигналу:

- якщо прапорець знятий, то використовується мінімальна затримка, рівна 0,1 одиниці модельного часу;
- якщо прапорець встановлений, то величина затримки дорівнює значенню дискрету часу блоку, що передує блоку **Memory**.

Блоки **First-Order Hold** і **Zero-Order Hold** прислужуються задля перетворення неперервного сигналу у дискретний.

На рис. 7.46. графічно поданий результат проходження неперервного синусоїдального сигналу через ці два блоки-екстраполятори.

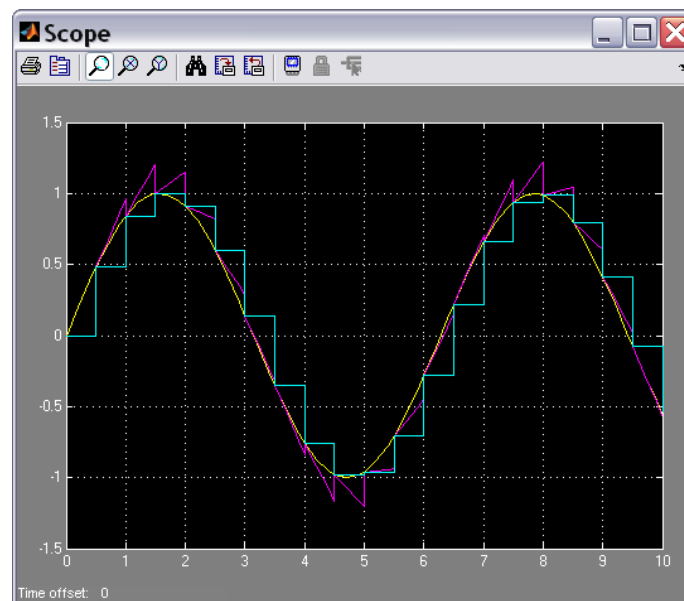


Рис. 7.46. Проходження синусоїдального сигналу через екстраполятори

7.2.5. Поділ *Discontinuities* (Розривні елементи)

Це, мабуть, найбільш корисний розділ бібліотеки *SimuLink*. Він включає 12 блоків (рис. 7.47).

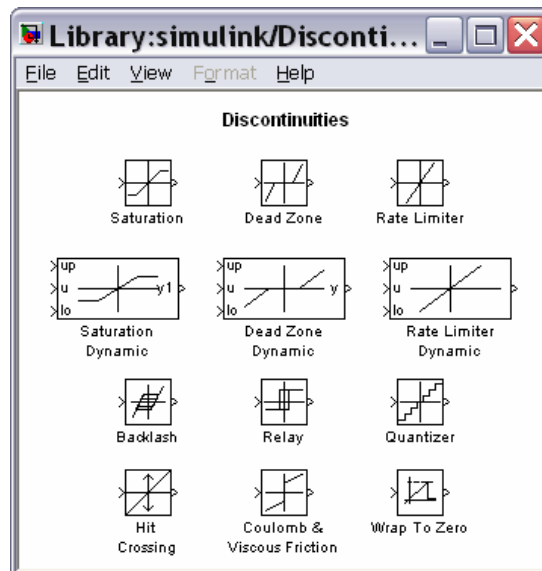


Рис. 7.47. Вміст поділу *Discontinuities*

Блок ***Saturation*** (*Насичення*) реалізує лінійну залежність із насиченням (обмеженням). Вихідна величина цього блоку збігається із вхідною, якщо остання знаходиться усередині зазначеного діапазону. Якщо ж вхідна величина виходить за рамки діапазону, то вихідний сигнал приймає значення найближчої з меж. Значення меж діапазону встановлюються у вікні настроювання блоку.

Блок ***Dead Zone*** (*Мертва зона*) реалізує нелінійність типу зони нечутливості. Параметрів настроювання тут два - початок і кінець зони нечутливості.

Блок ***Rate Limiter*** (*Обмежувач швидкості*) забезпечує обмеження згори і знизу швидкості змінювання сигналу, що проходить крізь нього. Вікно настроювання блоку містить два головних параметри: *Rising slew rate* і *Falling slew rate*. Блок працює у такий спосіб: спочатку обчислюється швидкість змінювання сигналу, що проходить крізь нього, за формулою

$$rate = \frac{u(i) - y(i-1)}{t(i) - t(i-1)},$$

де $u(i)$ - значення вхідного сигналу у момент часу $t(i)$; $y(i-1)$ - значення вихідного сигналу у момент $t(i-1)$. Далі, якщо обчислене значення $rate$ перевищує значення параметра *Rising slew rate* (R), вихідна величина визначається за формулою:

$$y(i) = y(i-1) + R \cdot \Delta t.$$

Якщо $rate$ є меншим за значення *Falling slew rate* (F), вихідна величина розраховується так

$$y(i) = y(i-1) + F \cdot \Delta t.$$

У тому випадку, коли значення $rate$ міститься проміж значень R і F , вихідна величина збігається зі вхідною

$$y(i) = u(i).$$

Блок **BackLash** (Люфт) реалізує нелінійність типу зазору. У ньому передбачено два параметри настроювання: *Deadband width* - величина люфту і *Initial output* - початкове значення вихідної величини.

Блок **Relay** (Реле) працює за аналогією зі звичайним реле: якщо вхідний сигнал перевищує деяке граничне значення, то на виході блоку формується деякий сталий сигнал. Блок має 4 параметри настроювання:

- *Switch on point* (Точка вмикання) - задає граничне значення, при перевищенні якого відбувається вмикання реле;
- *Switch off point* (Точка вимикання) - визначає рівень вхідного сигналу, при якому реле вимикається;
- *Output when on* (Вихід при включеному стані) - установлює рівень вихідної величини при включеному реле;
- *Output when off* (Вихід при виключеному стані) - визначає рівень вихідного сигналу при виключеному реле.

Блок **Quantizer** (Квантувач) здійснює дискретизацію вхідного сигналу за його величиною. Єдиним параметром настроювання цього блоку є *Quantization interval* - Інтервал квантування - розмір дискрету за рівнем вхідного сигналу.

Блок **Hit Crossing** (Виявити перетинання) дозволяє зафіксувати стан, коли вхідний сигнал "перетинає" деяке значення. При виникненні такої ситуації на виході блоку формується одиничний сигнал. Блок має 3 параметри настроювання (рис. 7.48):

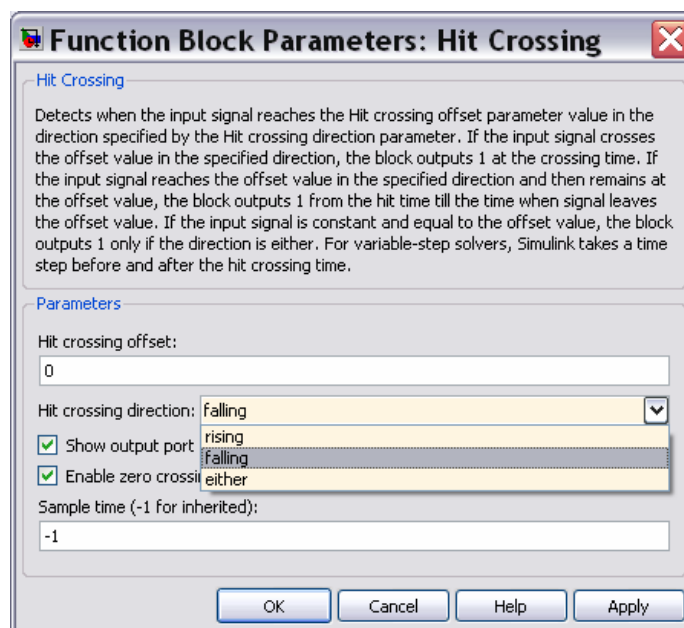


Рис. 7.48. Вікно настроювання блоку Hit crossing

- *Hit crossing offset* - визначає значення, перетинання якого необхідно ідентифікувати;

- *Hit crossing direction* дозволяє зазначити напрямок перетинання, при якому це перетинання повинно виявлятися; значення цього параметра вибирається за допомогою спадного меню, яке містить три альтернативи: *rising* (сходження), *falling* (спадання), *either* (у будь-якому напрямку);
- *Show output port* (зазначити порт виходу) - прапорець, за допомогою якого вибирається формат використання блоку.

При одночасному виконанні умов, що задаються параметрами *Hit crossing offset* і *Hit crossing direction*, на виході блоку формується одиничний сигнал. Його тривалість визначається значенням дискрету часу (параметр *Sample time*) блоку, який передує в моделі блоку ***Hit crossing***. Якщо цей параметр відсутній, то одиничний сигнал на виході блоку існує до його наступного спрацьовування.

Блок ***Coulomb & Viscous Friction*** (Кулонове і в'язке тертя) реалізує нелінійну залежність типу "лінійна з натягом". Якщо вхід додатний, то вихід пропорційний входу з коефіцієнтом пропорційності - "коефіцієнтом в'язкого тертя" - і збільшений на величину "натягу" ("кулонове, сухе тертя"). Якщо вхід є від'ємним, то вихід також є пропорційним входу (із тим же коефіцієнтом пропорційності) за відрахуванням величини "натягу". При вході рівному нулю вихід теж дорівнює нулю. У параметри настроювання блоку входять величини "кулонова тертя" ("натягу") і коефіцієнта "в'язкого тертя".

Блок ***Wrap To Zero*** (Скидання у нуль) забезпечує рівність нулевій вихідного сигналу, якщо значення вхідного сигналу перевищує встановлене значення єдиного параметра настроювання *Threshold* (Попіг). Якщо ж вхідний сигнал менше за *Threshold* вихідний сигнал дорівнює вхідному.

7.2.6. Поділ *Signal Routing* (Пересилання сигналів)

Поділ бібліотеки ***Connections*** (Зв'язки) призначений для побудови складних S-моделей, що складаються з інших моделей більш низького рівня. Склад блоків наведений на рис. 7.49.

Блок ***Mux*** (Мультиплексор) виконує об'єднання вхідних величин у єдиний вихідний вектор. При цьому вхідні величини можуть бути як скалярними, так і векторними. Довжина результуючого вектора дорівнює сумі довжин усіх векторів. Порядок елементів у векторі виходу визначається порядком входів (зверху вниз) і порядком розташування елементів усередині кожного входу. Блок має один параметр настроювання - *Number of inputs* (Кількість входів).

Блок ***Demux*** (Подільник, Демультіплексор) виконує зворотну функцію - поділяє вхідний вектор на задану кількість компонентів. Він також має єдиний параметр настроювання *Number of outputs* (Кількість виходів). У випадку, коли зазначене число виходів (N) задається меншим довжини вхідного вектора (M), блок формує вихідні вектори в такий спосіб. Перші (N-1) виходів будуть векторами однакової довжини, рівної цілій частині відношення $M/(N-1)$. Останній вихід буде мати довжину, рівну залишку від ділення.

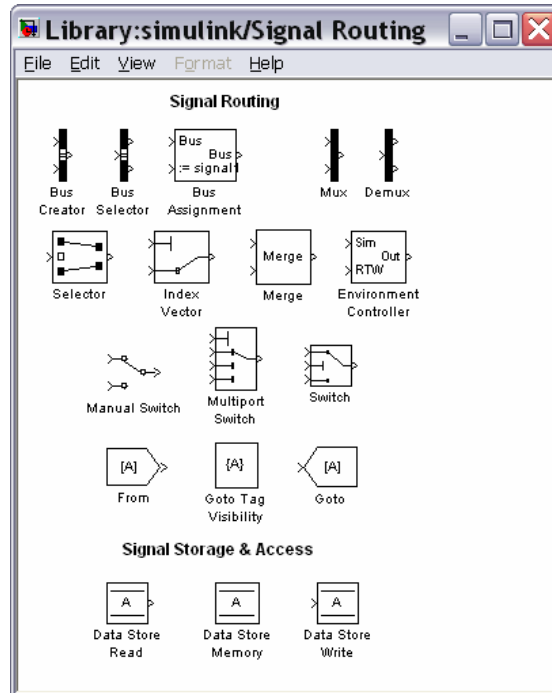


Рис. 7.49. Вміст поділу Signal Routing

Блоки **Bus Creator** (Побудувач шини) і **Bus Selector** (Роздільник шини) взагалі виконують ті самі функції, що й відповідно блоки **Mux** і **Demux**, але мають більші можливості щодо перерозподілу сигналів всередині шини.

Блоки **From** (Прийняти від), **Goto Tag Visibility** (Ознака видимості) і **Goto** (Передати до) використовуються спільно і призначені для обміну даними між різноманітними частинами S-моделі з урахуванням досяжності (видимості) цих даних.

Блоки **Data Store Read** (Читання даних), **Data Store Memory** (Запам'ятовування даних) і **Data Store Write** (Запис даних) також використовуються спільно і забезпечують не тільки передачу даних, але і їхнє збереження на інтервалі моделювання.

Блок **Merge** (Злиття) виконує об'єднання сигналів, що надходять до його входів, у єдиний.

Група блоків-перемикачів, що керують напрямком передачі сигналу, складається з трьох блоків: **Switch** (Перемикач), **Manual Switch** (Ручний перемикач) і **Multiport Switch** (Багатовходовий перемикач).

Блок **Switch** має три входи: два (1-й і 3-й) інформаційні й один (2-й) – керуючий - і один вихід. Якщо величина керуючого сигналу, що надходить до другого входу, не менше за деяке задане межеве значення (параметр *Threshold - Порог*), то на вихід блоку передається сигнал з 1-го входу, у протилежному разі - сигнал із 3-го входу.

Блок **Manual Switch** не має параметрів налаштування. У нього два входи й один вихід. На зображенні блоку показано перемичкою, який саме із двох входів залучений до виходу. Блок дозволяє "вручну" перемикати входи. Для цього необхідно двічі клацнути мишкою на зображенні блоку. При цьому змі-

ниться і зображення блоку - на ньому вихід уже буде сполучений перемичкою з іншим входом.

Блок *Multiport Switch* має не менше трьох входів. Перший (горішній) з них є керуючим, інші - інформаційними. Блок має один параметр настроювання *Number of inputs* (Кількість входів), який визначає кількість інформаційних входів. Номер входу, що з'єднується з виходом, дорівнює значенню керуючого сигналу, що надходить на верхній вхід. Якщо це значення є дробовим числом, то воно округлюється до цілого по звичайних правилах. Якщо воно менше за одиницю, то воно вважається рівним 1; якщо воно більше кількості інформаційних входів, то воно приймається рівним найбільшому номеру (входи нумеруються зверху вниз, крім самого верхнього - керуючого).

7.2.7. Поділ Math Operations (Математичні операції)

У поділі *Math Operations* (Математичні операції) містяться блоки, які реалізують деякі вбудовані математичні функції системи Matlab. Вони об'єднані у три групи (рис. 7.50)

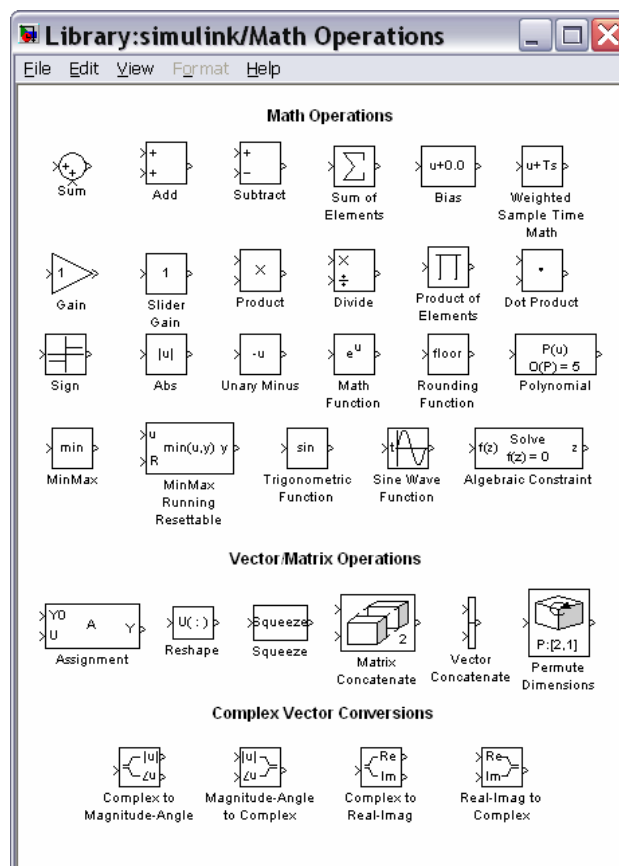


Рис. 7.50. Вміст поділу Math Operations

- *Math Operations* (Математичні операції),
- *Vector/Matrix Operations* (Векторно-матричні операції)
- *Complex Vector Conversions* (Перетворення комплексного вектора).

У першу групу *Math Operations* входять блоки, які здійснюють математичні перетворення вхідного сигналу у вихідний.

Блоки *Sum*, *Add*, *Subtract* і *Sum of Elements* здійснюють підсумовування сигналів, що надаються до їхніх входів.

Блок *Sum* може використовуватися у двох режимах:

- додавання вхідних сигналів (у тому числі з різними знаками);
- підсумовування елементів вектора, що надходить на вхід блоку.

Для керування режимами роботи блоку використовується два параметри (рис. 7.51):

- *Icon Shape* (Форма зображення),
- *List of signs* (Список знаків).

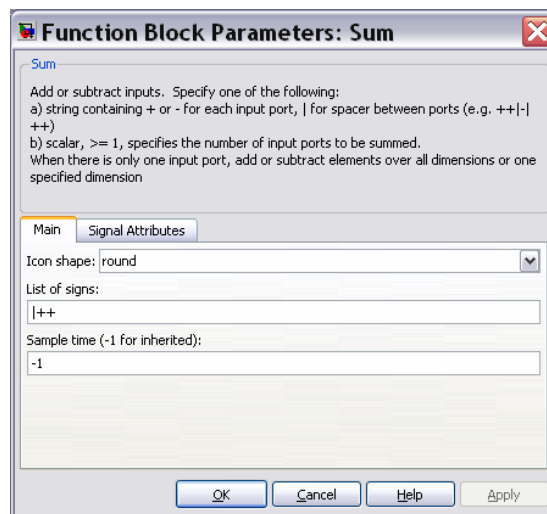


Рис. 7.51. Вікно налаштування блоку *Sum*

Перший параметр може приймати два значення: *round* (круглий) і *rectangular* (прямокутний).

Значення другого параметра можуть задаватися одним із трьох способів:

- у виді послідовності знаків "+" або "-"; при цьому кількість знаків визначає кількість входів блоку, а самий знак - полярність відповідного вхідного сигналу;
- у виді цілої додатної і більше 1 константи; значення цієї константи визначає кількість входів блоку, а усі входи вважаються додатними;
- у виді символу "1", який указує, що блок використовується в другому режимі

Блок *Bias* додає постійну величину (параметр налаштування *Bias*) до вхідного сигналу.

Блок *Gain* є лінійною підсилювальною ланкою, тобто здійснює множення вхідного сигналу на постійне число або вектор, значення якого задається параметром налаштування і вказується на зображенні блоку. Вхідна величина блоку (*u*) може бути скалярною, векторною або матричною. У випадку, коли вхідний сигнал є вектором довжиною *N* елементів, коефіцієнт підсилювання має бути вектором тої самої довжини. У списку *Multiplication* (Множення) оби-

рається один з наступних способів множення вхідної величини на вектор K коефіцієнтів підсилювання: *Element wise*($K \cdot u$) – поелементне множення вхідного вектора на вектор коефіцієнтів підсилювання; *Matrix*($K \cdot u$) – матричне множення вектора коефіцієнтів підсилювання на матрицю вхідних величин; *Matrix*($u \cdot K$) – матричне множення матриці вхідних величин на вектор коефіцієнтів підсилювання; *Matrix*($K \cdot u$) (*u vector*) – матричне множення векторів K і u .

Блок **Slider Gain** є різновидом підсилювальної ланки і одним з елементів взаємодії користувача з моделлю. Він дозволяє в зручній діалоговій формі змінювати значення деякого параметра в процесі моделювання. Щоб відчинити вікно з "повзунковим" регулятором (рис. 7.52), необхідно подвійно клацнути мишкою на зображенні блоку.

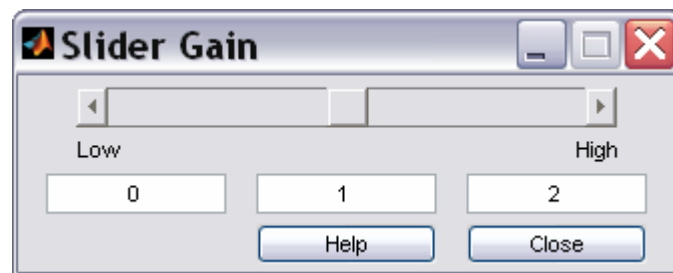


Рис. 7.56. Вікно налаштування блоку *Slider Gain*

Вікно **Slider Gain** має три поля введення інформації:

- для вказівки нижньої межі змінювання параметра (*Low*);
- для вказівки верхньої межі змінювання параметра (*High*);
- для вказівки поточного значення.

Поточне значення має лежати усередині діапазону [*Low*, *High*]. Його можна вводити, записуючи значення у середнє поле, або переміщуючи мишкою повзунок у верхній частині вікна налаштування. Проте при виборі нового діапазону необхідно спочатку зазначити нове значення параметра, а потім змінити межі діапазону.

Блок **Product** виконує множення або ділення кількох вхідних сигналів. У параметри налаштування входять кількість входів блоку (*Number of inputs*) й вид виконуваної операції (*Multiplication*). Завдання значень цих параметрів аналогічно налаштуванню блоку **Sum**. Якщо як значення параметра налаштування блоку ввести "1", буде обчислюватися добуток елементів вхідного вектора. При цьому на зображенні блоку виводиться символ **P**. Вхідні сигнали можуть бути векторними або матричними. У списку *Multiplication* обирається спосіб множення вхідних величин: *Element wise* – поелементне множення вхідних векторів або матриць; *Matrix* – матричне множення вхідних векторів або матриць. У випадку, коли результат виконання має містити ділення на деякі вхідні величини, у полі *Number of inputs* (Кількість входів) слід ввести послідовність символів "*" або "/" (за кількістю входів блоку). Якщо обрано матричне множення, то символ "/" означає множення на матрицю, обернену по відношенню до матриці відповідної вхідної величини.

У блоці **Dot Product** (Скалярний добуток) є лише два входи. Вхідні сигнали блоку мають бути векторами однакової довжини. Вихідна величина блоку у кожний момент часу дорівнює сумі добутків відповідних елементів цих двох векторів. Якщо вектори є комплексними, то перед множенням першій вектор (верхній вхідний порт) замінюється комплексно-спряженим.

Блок **Sign** реалізує нелінійність типу сигнум-функції. У ньому немає параметрів настроювання. Блок формує вихідний сигнал, що приймає тільки три можливих значення: "+1" - у випадку, коли вхідний сигнал є додатним, "-1" - при від'ємному вхідному сигналі і "0" при вхідному сигналі, рівному нулю.

Блок **Abs** формує абсолютне значення вхідного сигналу. Він не має параметрів настроювання.

Блок **Trigonometric Function** забезпечує перетворення вхідного сигналу за допомогою однієї з таких функцій MatLAB: *sin, cos, tan, asin, acos, atan, atan2, sinh, cosh, tanh*. Обрання необхідної функції здійснюється у вікні настроювання блоку за допомогою спадного меню.

Блок **Math Function** дозволяє обрати для перетворення вхідного сигналу елементарні не тригонометричні і не гіперболічні функції, такі як *exp, log, 10^u, log10, square (u^2), sqrt, pow, reciprocal (1/u), hypot, rem, mod*. Потрібна функція обирається за допомогою спадного меню у вікні настроювання.

Блок **Rounding Function** містить різноманітні функції округлення, передбачені в MatLAB. Він здійснює округлення значення вхідного сигналу. Вибір конкретного методу округлення здійснюється також за допомогою спадного меню у вікні настроювання.

Для зазначених вище блоків ім'я обраної функції виводиться на графічному зображенні блоку.

Блок **MinMax** здійснює пошук мінімального або максимального елемента вхідного вектора. Якщо входом є скалярна величина, то вихідна величина збігається із вхідною. Якщо входів декілька, шукається мінімум або максимум серед входів. У число настроювань входить вибір методу й кількість входів блоку.

Блоки третьої групи – **Complex Vector Conversions** (Перетворення комплексного вектора) здійснюють перетворення комплексних вхідних сигналів у дійсні і навпаки. Блоки **Complex to Magnitude-Angle** та **Complex to Real-Imag** здійснюють перетворення комплексного вхідного сигналу у два дійсних сигнали, що є модулем і аргументом вхідного сигналу у першому випадку і дійсною і уявною частинами – у другому випадку. Блоки **Magnitude-Angle to Complex** та **Real-Imag to Complex** перетворюють два вхідних дійсних сигнали у єдиний комплексний.

7.2.8. Поділ *Logic & Bit Operations* (Логічні та бітові операції)

Як вказано у назві поділу, він містить блоки, що забезпечують певну логічну обробку вхідних величин, або перетворення їх двійкового подання (рис. 7.53).

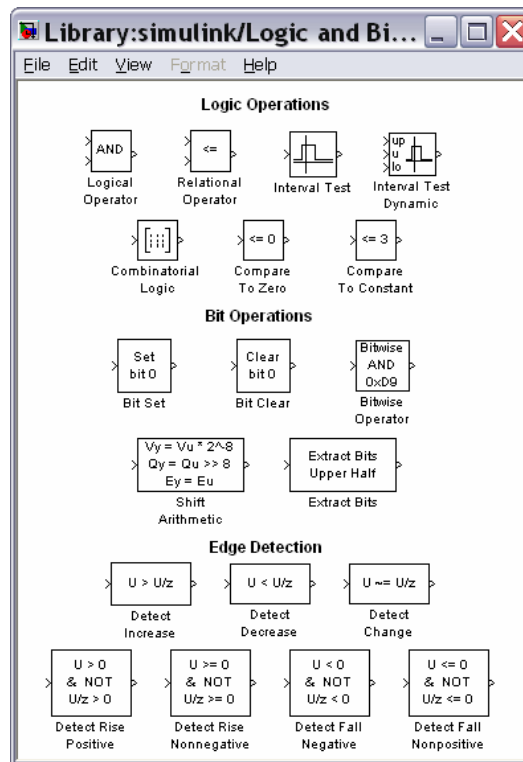


Рис. 7.53. Вміст поділу *Logic & Bit Operations*

Загальним для всіх блоків групи *Logic Operations* є те, що вихідна величина в усіх них є бульовою, тобто може приймати лише два значення: "1" ("істина") або "0" ("хибність"). У багатьох з них бульовими мають бути й усі вхідні величини.

Блок *Relational Operator* реалізує операції відношення між двома вхідними сигналами $>$, $<$, \leq , \geq , $=$, \sim (відповідно: більше, менше, менше або дорівнює, більше або дорівнює, тотожно, не дорівнює). Конкретна операція обирається при налаштуванні параметрів блоку за допомогою спадного меню. Знак операції виводиться на зображенні блоку.

Блок *Logical Operator* містить набір основних логічних операцій AND, OR, NAND, NOR, XOR, NOT. Вхідні величини мають бути бульовими. обрання необхідної логічної операції здійснюється у вікні налаштування блоку за допомогою спадного меню. Другим параметром налаштування є кількість вхідних величин (портів) блоку (*Number of input ports*), тобто кількість аргументів логічної операції.

Блок *Combinatorial Logic* забезпечує перетворення вхідних булевих величин у вихідну у відповідності із заданою таблицею істинності. Блок має єдиний параметр настроювання - *Truth table* (таблиця істинності).

7.2.9. Поділ *User-defined Functions* (Функції, що визначаються користувачем)

У поділі *User-defined Functions* (Користувацькі функції) містяться блоки, призначення яких визначає користувач (рис. 7.54).

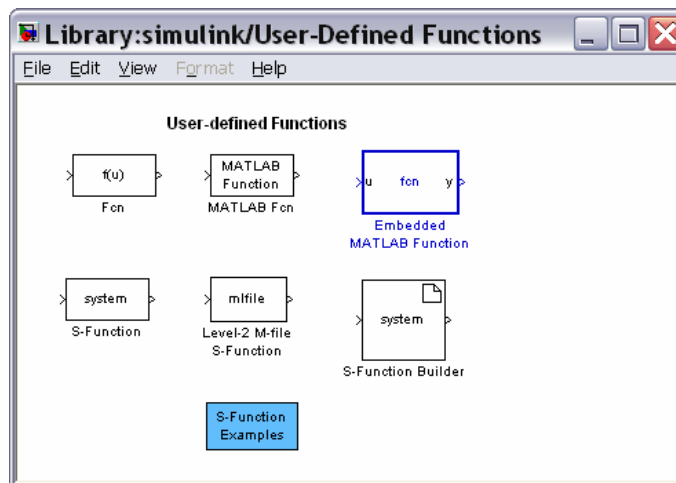


Рис. 7.54. Вміст поділу *User-defined Functions*

Блок *Fcn* дозволяє користувачеві ввести будь-яку скалярну функцію від одного (скалярного або векторного) аргументу, яка виражається через стандартні функції MatLAB. Вираз функції вводиться у вікні настроювання блоку згідно правил М-мови. Для позначення вхідного сигналу (аргументу функції) використовується символ *u*.

Блок *MATLAB Fcn* дозволяє застосувати до вхідного сигналу будь-яку підпрограму обробки, реалізовану у виді М-файлу. На відміну від попереднього блоку, тут до числа параметрів настроювання доданий параметр *Output width* (*Ширина вихідного сигналу*), який визначає кількість елементів вихідного вектора. Окремий *i*-й елемент вихідного вектора у вікні настроювання *MATLAB Fcn* задається у виді функції, що записана на М-мові, якій передує запис $u(i)=$.

За допомогою блоку *S-function* користувач має змогу реалізувати у виді візуального блоку Simulink будь-яку програму обробки вхідного сигналу, включаючи створення складних моделей систем, що описані нелінійними диференційними або кінцево-різницеєвими рівняннями, і обробку дискретних у часі сигналів. Більш детально робота з S-функціями описана далі, у главі 4.

Блок *S-function Builder* (Побудувач S-функцій) дає можливість користувачеві утворювати S-функцію у діалоговому режимі.

7.2.10. Поділ Ports & Subsystems (Порту та підсистему)

Більшість блоків поділу *Ports & Subsystems* призначені для розробки складних за ієрархією S-моделей, що містять моделі більш низького рівня (підсистеми), а також забезпечують встановлення необхідних зв'язків між кількома S-моделями (рис. 7.55).

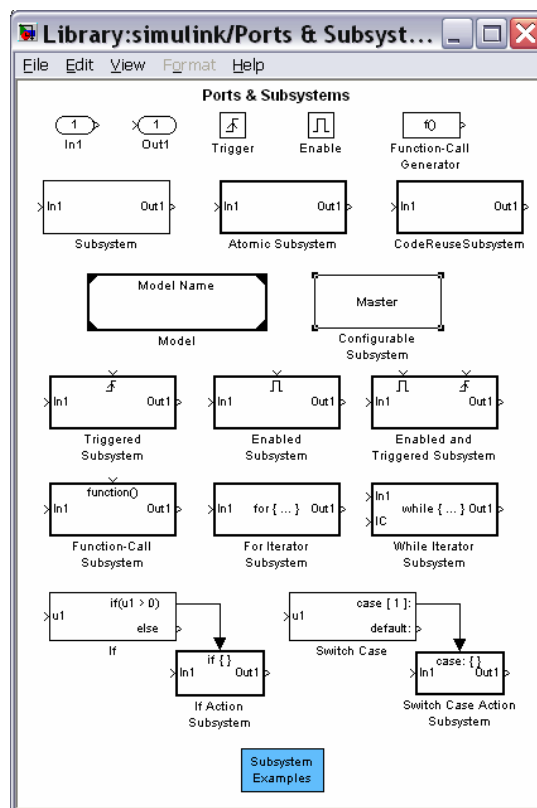


Рис. 7.55. Вміст поділу Ports & Subsystems

Підсистема являє собою S-модель більш низького рівня, у яку можуть бути вкладені підсистеми різних рівнів. підсистеми можуть функціонувати лише у складі основної S-моделі, зв'язок з якою здійснюється через вхідні (**In**) і вихідні (**Out**) порти підсистеми.

Роль підсистеми у S-моделі така сама, що й роль функцій (процедур) в основній M- програмі, яка їх викликає. При цьому вхідні і вихідні величини підсистеми визначаються відповідно її вхідними і вихідними портами. Результати виконання дій у підсистемі (вихідні величини) у подальшому можуть бути використані у S-моделі, яка її викликає (або у підсистемі більш високого рівня).

Застосування підсистем дозволяє звести складання складної S-моделі до утворення сукупності вкладених простих підсистем більш низького рівня, що робить моделювання більш наочним і спрощує налагодження моделі.

Опишемо основні блоки поділу.

Блоки **In** (Вхідний порт) і **Out** (Вихідний порт) забезпечують інформаційний зв'язок між підсистемою і S-моделлю, що її викликає.

Блоки *Enable* (Дозволити) і *Trigger* (Засувка) призначені для логічного керування роботою підсистем S-моделі.

Раніше розглянути блоки *Ground* (Земля) і *Terminator* (Обмежувач) можуть використовуватися як "заглушки" для тих портів, які з якоїсь причини не були з'єднані з іншими блоками S-моделі. При цьому блок *Ground* застосовується для вхідних портів, а блок *Terminator* – для вихідних.

Блок *Subsystem* (Підсистема) є "заготівкою" для створення підсистеми. Подвійне клацання на його зображенні приводить до появи на екрані вікна, в якому розміщена блок-схема, що складається лише з одного вхідного порта, який з'єднаний з одним вихідним портом (рис. 7.56). Це є нагадуванням, що утворювана користувачем підсистема має обов'язково містити з'єднані між собою (можливо, через інші блоки) вхідні і вихідні порти.

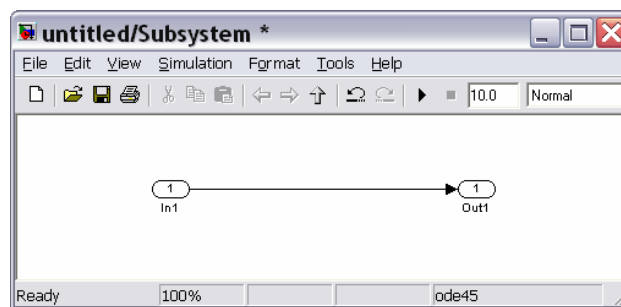


Рис. 7.56. Вікно заготівки блоку *Subsystem*

У вікні, що відчинилося, користувач будує блок-схему підсистеми за звичайними правилами побудови блок-схем, а потім записує її на диск. Розміщення додаткових вхідних і вихідних портів приведе до появи на зображенні блоку *Subsystem* додаткових входів і виходів. При цьому поряд з відповідними входами і виходами блоку *Subsystem* виникнуть написи, що зроблені на вхідних і вихідних портах підсистеми.

7.3. Побудова блок-схем

Розглянемо операції, за допомогою яких можна формувати блок-схеми складних динамічних систем.

7.3.1. Виділення об'єктів

При створенні й редагуванні S-моделі потрібно виконувати такі операції, як копіювання або вилучення блоків і ліній, для чого необхідно спочатку виділити один чи кілька блоків і ліній (об'єктів).

Щоб *виділити окремий об'єкт*, потрібно клацнути на ньому один раз. При цьому з'являються маленькі кола по рогах блоку, або на початку й кінці лінії. При цьому стають невиділеними усі інші попередньо виділені об'єкти. Якщо клацнути по блоку другий раз, він стає невиділеним.

На рис. 7.57 наведений результат виділення лінії, що з'єднує блоки Constant і XYGraph, а на рис. 7.58 - блоку *Clock*.

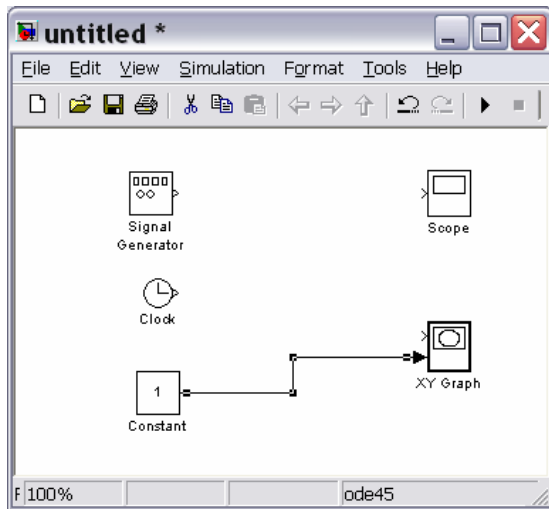


Рис. 7.57. Виділена лінія

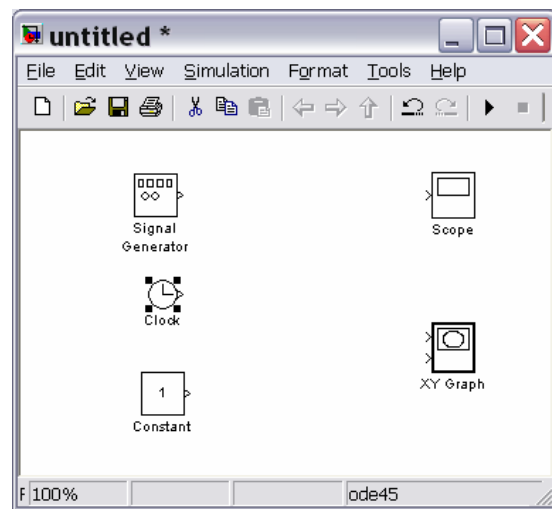


Рис. 7.58. Виділений блок Clock

Щоб виділити кілька об'єктів, слід, утримуючи натисненою клавішу Shift, клацнути на кожному з них, а потім відпустити клавішу.

Саме у такий спосіб на рис. 7.59 виділені блоки *Signal Generator*, *Constant* і *XYGraph*.

Кілька об'єктів можна виділити також за допомогою прямокутної рамки. Для цього потрібно клацнути мишкою у точці, яка буде прислужуватися ро-гом рамки, а потім, утримуючи кнопку миші натисненою, протягнути курсор у напрямку діагоналі прямокутника. В результаті навколо об'єктів, що виділяються має виникнути пунктирна рамка (рис. 7.60). Коли усі потрібні об'єкти будуть охоплені рамкою, потрібно відпустити кнопку миші.

Виділення усієї моделі, тобто усіх об'єктів в активному вікні блок-схеми, здійснюється одним із двох шляхів:

- 1) обранням команди *Select All* у меню *Edit* вікна блок-схеми;
- 2) натисканням сукупності клавіш <Ctrl>+<A>.

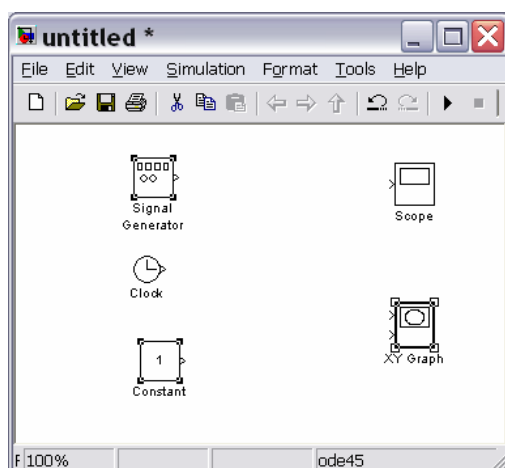


Рис. 7.59. Виділення кількох блоків

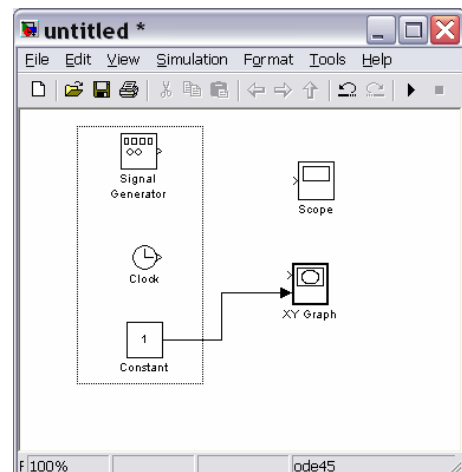


Рис. 7.60. Виділення за допомогою рамки

7.3.2. Операції з блоками

Копіювання блоків з одного вікна у інше

Під час створення й редагування моделі потрібно копіювати блоки з бібліотеки або іншої моделі у поточну модель. Для цього достатньо:

- відчинити потрібний розділ бібліотеки чи вікно моделі - прототипу;
- перетягнути мишкою потрібний блок у вікно утвореної (редагуючої) моделі.

Інший спосіб є таким:

- 1) виділити блок, який потрібно скопіювати;
- 2) обрати команду *Copy* (Копіювати) з меню *Edit* (Редагування);
- 3) зробити активним вікно, в яке потрібно скопіювати блок;
- 4) обрати в ньому команду *Paste* (Встановити) з меню *Edit*.

Кожному зі скопійованих блоків автоматично присвоюється ім'я. Перший скопійований блок матиме те саме ім'я, що й блок у бібліотеці. Кожний наступний блок того ж типу матиме те саме ім'я з додаванням порядкового номера. Користувач може перейменувати блок (див. далі).

При копіюванні блок одержує ті самі значення настроюваних параметрів, що й блок - оригінал.

Переміщення блоків у моделі

Щоб перемістити блок усередині моделі з одного місця у інше, достатньо перетягнути його у це положення за допомогою мишки. При цьому будуть автоматично перерисовані лінії зв'язків інших блоків з тим, якого переставлено.

Переставити кілька блоків одночасно, включаючи з'єднувальні лінії можна у такий спосіб:

- виділити блоки й лінії (див попередній розділ);
- перетягнути мишкою один із виділених блоків на нове місце; решта блоків, зберігаючи всі відносні відстані, посядуть нові місця.

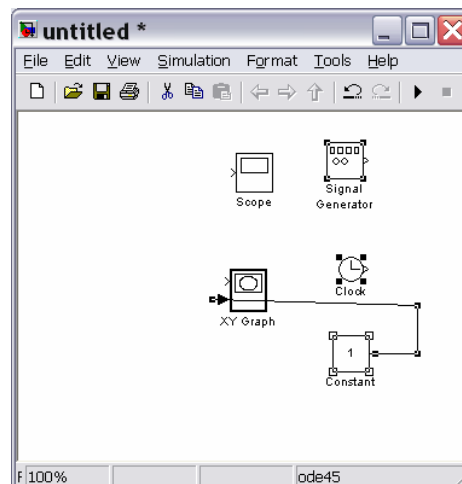


Рис. 7.61. Результат переміщення блоків, виділених на рис. 7.60

На рис. 7.61 показаний результат таких дій з блоками, виділеними на рис. 7.60.

Дублювання блоків усередині моделі

Щоб **скопювати блоки усередині моделі** потрібно зробити наступне:

- 1) натиснути клавішу <Ctrl>;
- 2) не відпускаючи клавішу <Ctrl>, встановити курсор на блок, який необхідно скопіювати й перетягнути його у нове положення.

Того самого результату можна досягти, якщо просто перетягнути мишкою блок у нове положення, але за допомогою правої клавіші мишки.

На рис. 7.62 подано результат копіювання блоків *Scope* і *XYGraph*.

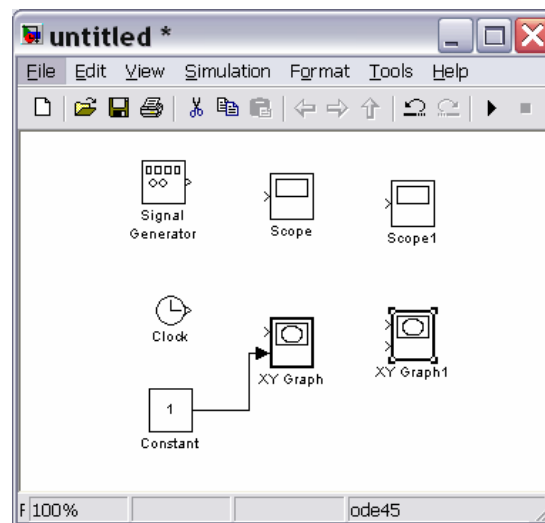


Рис. 7.62. Результат копіювання блоків

Встановлення параметрів блоку

Функції, які виконує блок, залежать від значень параметрів блоку. Встановлення цих значень здійснюється у вікні настроювання блоку, яке виникає, якщо подвійно клацнути на зображенні блоку у блок-схемі.

Вилучення блоків

Для **вилучення непотрібних блоків** із блок-схеми достатньо виділити ці блоки так, як було зазначено раніше, і натиснути клавішу <Delete> або <Backspace>. Можна також використати команду *Clear* або *Cut* із розділу *Edit* меню вікна блок-схеми. Якщо використано команду *Cut*, то у подальшому вилучені блоки можна скопіювати зворотню у модель, якщо скористатися командою *Paste* того ж розділу меню вікна схеми.

Від'єднання блоку

Для **від'єднання блоку від з'єднуючих ліній** достатньо натиснути клавішу <Shift> і, не відпускаючи її, перетягнути блок у деяке інше місце.

Змінювання кутової орієнтації блоку

У звичайному зображенні сигнал проходить крізь блок зліва направо (ліворуч містяться входи блоку, а праворуч - виходи). Щоб **змінити кутову орієнтацію блоку** потрібно:

- виділити блок, який потрібно повернути;
- обрати меню *Format* у вікні блок-схеми;
- у додатковому меню, яке з'явиться на екрані обрати команду *Flip Block* - поворот блоку на 180 градусів, або *Rotate Block* - поворот блоку за годинниковою стрілкою на 90 градусів.

На рис. 7.63 показаний результат застосування команди *Rotate Block* до блоку **Constant** і команд *Rotate Block* і *Flip Block* - до блоку **Signal Generator**.

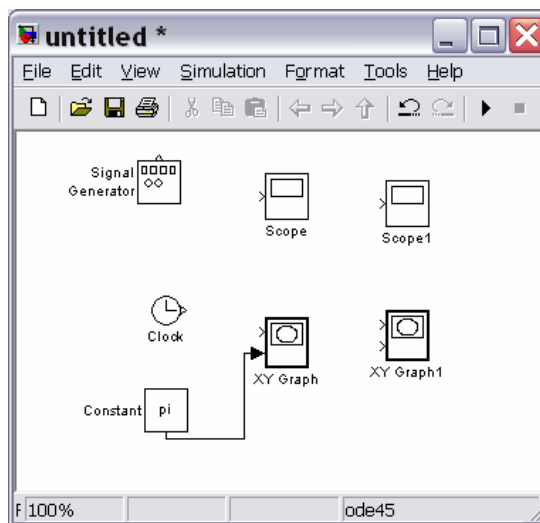


Рис. 7.63. Результат змінювання орієнтації блоків

Змінювання розмірів блоку

Щоб змінити розміри блоку, потрібно зробити наступне:

- виділити блок, розміри якого треба змінити;
- навести курсор мишки на одну з рогових міток блоку; при цьому на екрані у цієї мітки повинен з'явитися новий курсор у вигляді обопільної стрілки під нахилом 45 градусів;
- захопити цю мітку мишкою і перетягнути у нове положення; при цьому протилежна мітка цього блоку залишиться нерухомою.

На рис. 7.64 показаний результат процесу збільшення розмірів блоку **XYGraph1**.

Змінювання імен блоків та маніпулювання з ними

Усі імена блоків у моделі мають бути унікальними і мати, як мінімум один символ. Якщо блок орієнтований зліва направо, то ім'я, за замовчуванням, міститься під блоком, якщо справа наліво - понад блоком, якщо ж зверху униз або знизу уверх - праворуч блоку (див. рис. 7.64).

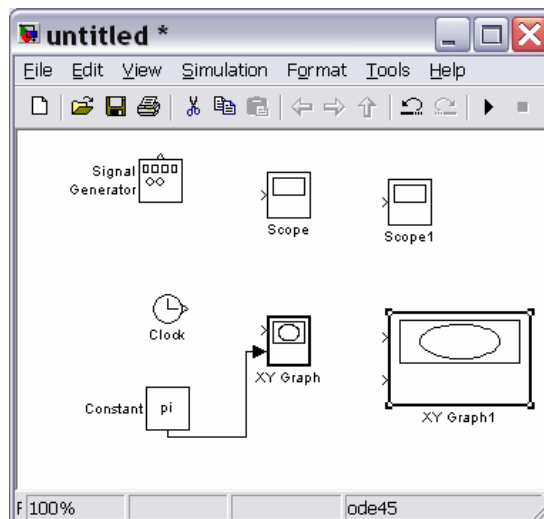


Рис. 7.64. Результат розтягування блоку XYGraph1

Змінювання ймення блоку здійснюється у такий спосіб: треба клацнути на існуючому імені блоку, потім, використовуючи клавіші звичайного редагування тексту, змінити це ім'я на потрібне.

Для **змінювання шрифту** слід виділити блок, потім обрати команду *Font* (Шрифт) із меню *Format* вікна моделі і обрати потрібний шрифт із поданого переліку.

Щоб **змінити місце розташування ймення виділеного блоку**, існують два шляхи:

- перетягнути ім'я на протилежний бік мишкою;
- скористатися командою *Flip Name* (Розвернути ім'я) із поділу *Format* меню вікна моделі – вона теж переносить ім'я на протилежний бік.

Вилучити ім'я блоку можна, використовуючи команду *Hide Name* (Сховати ім'я) з меню *Format* вікна моделі. Щоб **відновити** потім **відображення імені** поряд із зображенням блоку, слід скористатися командою *Show Name* (Показати ім'я) того самого меню.

7.3.3. Проведення з'єднувальних ліній

Сигнали у моделі передаються по лініях. Кожна лінія може передавати або скалярний, або векторний сигнал. Лінія з'єднує вихідний порт одного блоку із входним портом іншого блоку. Лінія може також з'єднувати вихідний порт одного блоку із входними портами кількох блоків через розгалужування лінії.

Створення лінії між блоками

Для сполучення вихідного порту одного блоку із входним портом іншого блоку слід виконати таку послідовність дій:

- встановити курсор на вихідний порт першого блоку; при цьому курсор має перетворитися на перехрестя;

- утримуючи натисненою ліву кнопку миші, пересунути перехрестя до вхідного порту другого блоку;
- відпустити кнопку миші; Simulink замінить символи портів з'єднувальною лінією з поданням напрямку передавання сигналу.

Саме таким чином з'єднано на рис. 7.60 вихід блоку **Constant** із входом блоку **XYGraph**.

Лінії можна рисувати як от вхідного порту до вихідного, так і навпаки.

Simulink малює з'єднувальні лінії, використовуючи лише горизонтальні й вертикальні сегменти Для утворення діагональної лінії натисніть і утримуйте клавішу <Shift> протягом рисування.

Створення розгалуження лінії

Лінія, що розгалужується, починається з існуючої і передає її сигнал до вхідного порту іншого блоку. Як існуюча, так і відгалужена лінія передають той самий сигнал. Розгалужена лінія дає можливість передати той самий сигнал до кількох блоків.

Щоб **утворити відгалуження** від існуючої лінії, потрібно:

- установити курсор на точку лінії, від якої має відгалужуватися інша лінія;
- натиснувши й утримуючи клавішу <Ctrl>, натиснути й утримувати ліву кнопку миші;
- провести лінію до вхідного порту потрібного блоку; відпустити клавішу <Ctrl> і ЛКМ (див. рис. 7.65).

Те саме можна зробити, використовуючи праву кнопку миші, при цьому не потрібно користатися клавишою <Ctrl>.

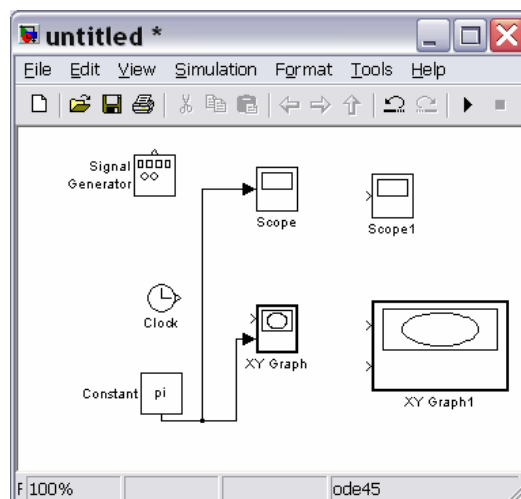


Рис. 7.65. Утворення розгалуження лінії

Створення сегмента лінії

Лінії можуть бути нарисовані по сегментах. У цьому разі для створення наступного сегмента слід установити курсор у кінець попереднього сегмента і нарисувати (за допомогою миші) наступний сегмент. У такий спосіб, напри-

клад, з'єднані на рис. 7.66 блоки *Clock* з *XYGraph1* та *Signal Generator* з *Scope1*.

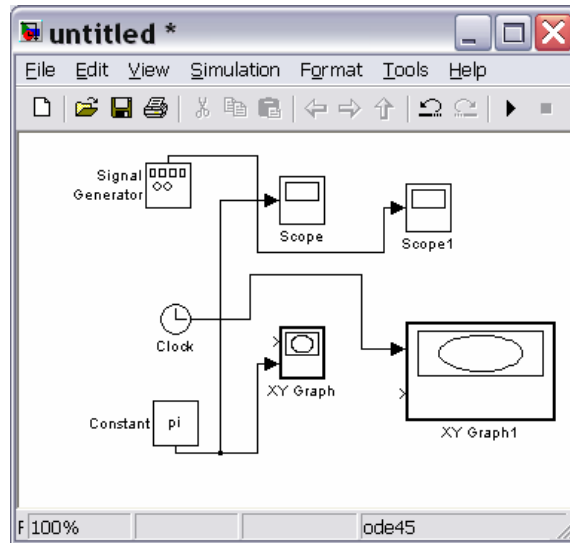


Рис. 7.66. Лінії, нарисовані по сегментах

Пересування сегмента лінії

Щоб *пересунути окремий сегмент* лінії, необхідно виконати наступне:

- установити курсор на сегмент, який потрібно пересунути;
- натиснути й утримувати ліву кнопку миші; при цьому курсор має перетворитися на "хрест";
- пересунути "хрест" до нового положення сегмента;
- відпустити кнопку миші.

На рис. 7.67 показаний результат пересування вертикального сегменту лінії, що з'єднує блоки *Random Number* з *XYGraph1*.

Не можна пересунути сегмент, який безпосередньо прилягає до порту блоку.

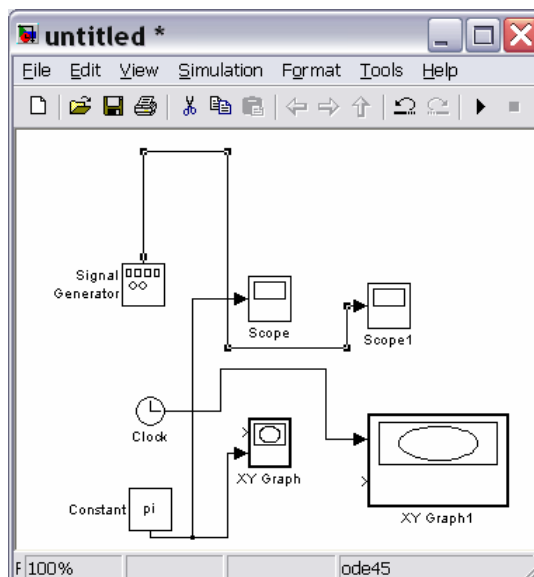


Рис. 7.67. Сегмент лінії переміщений угору

Поділення лінії на сегменти

Щоб *поділити лінію на два сегменти*, потрібно:

- виділити лінію;
- установити курсор у ту точку виділеної лінії, у якій лінія має бути поділеною на два сегменти;
- утримуючи натисненою клавішу <Shift>, натиснути й утримувати ліву кнопку миші; курсор перетвориться на маленьке коло; на лінії утвориться злам;
- пересунути курсор у нове положення зламу;
- відпустити кнопку миші і клавішу <Shift>.

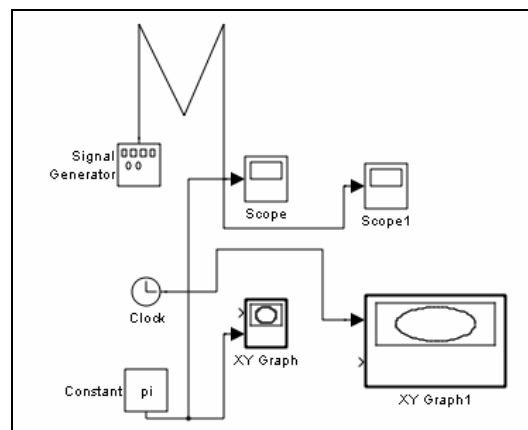


Рис. 7. 68. Верхня горизонтальна лінія поділена на два сегменти

Приклад проведення цих дій подано на рис. 7.68, де лінія, яка з'єднує блоки *Sine Wave* і *XYGraph1* поділена на два сегменти.

Пересування зламу лінії

Для *пересування зламу* лінії достатньо перетягнути точку цього зламу у нове положення на блок-схемі.

7.3.4. Мітки сигналів і коментарів

Задля наочності оформлення блок-схеми й зручності користування нею можна супроводжувати лінії мітками сигналів, що прямують по ній. Мітка розміщується під або над горизонтальною лінією, ліворуч або праворуч вертикальної лінії. Мітка може бути розташована на початку, у кінці або посередині лінії.

Створення й маніпулювання мітками сигналів

Щоб *створити мітку сигналу*, треба подвійно клацнути на сегменті лінії і ввести мітку (рис. 7.69). При створенні мітки сигналу необхідно подвійно клацнути саме точно на лінії, бо інакше буде створений коментар до моделі.

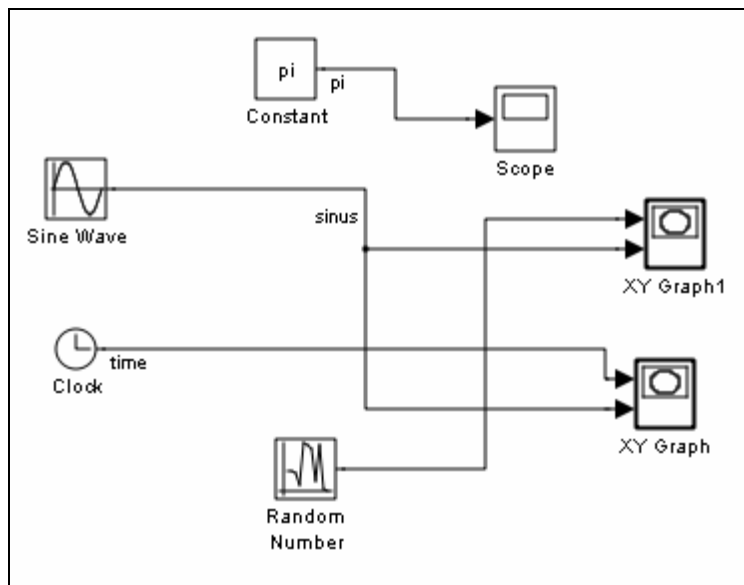


Рис. 7. 69. Створені мітки pi, sinus і time

Для *пересування мітки* треба її просто перетягнути на нове місце мишкою.

Щоб *скопювати мітку*, слід натиснути й утримувати клавішу <Ctrl> і перетягнути мітку до нового положення на лінії, або обрати інший сегмент лінії, на якому потрібно встановити копію мітки і подвійно клацнути по цьому сегменту лінії.

Відредагувати мітку можна клацнувши на ній і здійснюючи потім відповідні змінювання як у звичайному текстовому редакторі.

Щоб *вилучити мітку*, натисніть і утримуйте клавішу <Shift>, виділіть мітку і знищити її, використовуючи клавіші <Delete> або <Backspace>. При цьому будуть вилучені усі мітки цієї лінії

Розповсюдження міток лінії

Розповсюдження міток лінії - це процес автоматичного переносу імені мітки до сегментів однієї лінії, що розірвані блоками *From/Goto*, *Mux*.

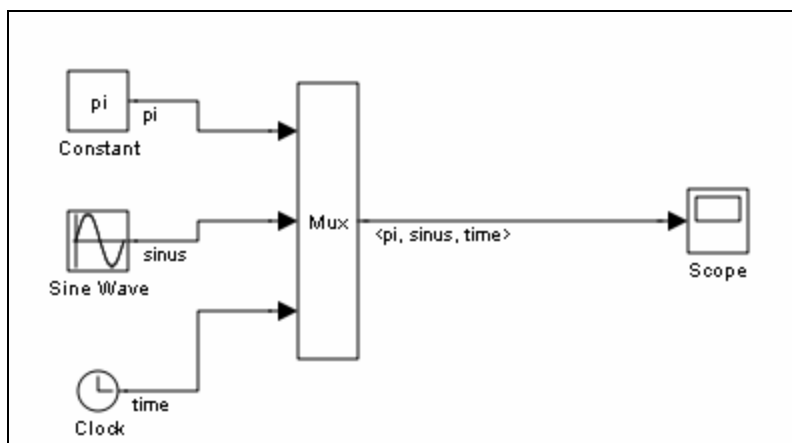


Рис. 7.70. Розповсюдження міток

Щоб **розповсюдити мітки**, створіть у другому й наступних сегментах лінії мітки з ім'ям '<'. Після повернення до вікна блок-схеми у цих сегментах автоматично будуть проставлені мітки (див. рис. 7.70)

Створення коментаря й маніпулювання ним

Коментарі дають можливість опоряджувати блок-схеми текстовою інформацією про модель та окремі її складові. Коментарі можна проставляти у будь-якому вільному місці блок-схеми (див. рис. 7.71).

Для **створення коментаря** двічі клацніть у будь-якому вільному місці блок-схеми, а потім уведіть коментар у прямокутнику, що виник.

Для **переміщення коментаря** у інше місце його потрібно перетягнути на це місце за допомогою миші.

Щоб **скопювати коментар**, достатньо натиснути клавішу <Ctrl> і, утримуючи її у цьому положенні, перетягти коментар у нове місце.

Для **редагування коментаря** треба клацнути на ньому, а потім внести потрібні зміни як у звичайному текстовому редакторі. Щоб **змінити шрифт**, його розмір або стиль, слід виділити текст коментаря, який потрібно змінити, а потім обрати команду *Font* із меню *Format* вікна блок-схеми, вибрати у вікні, що виникне, назву шрифту, його розмір, атрибути й стиль і натиснути кнопку <Ok> у цьому вікні.

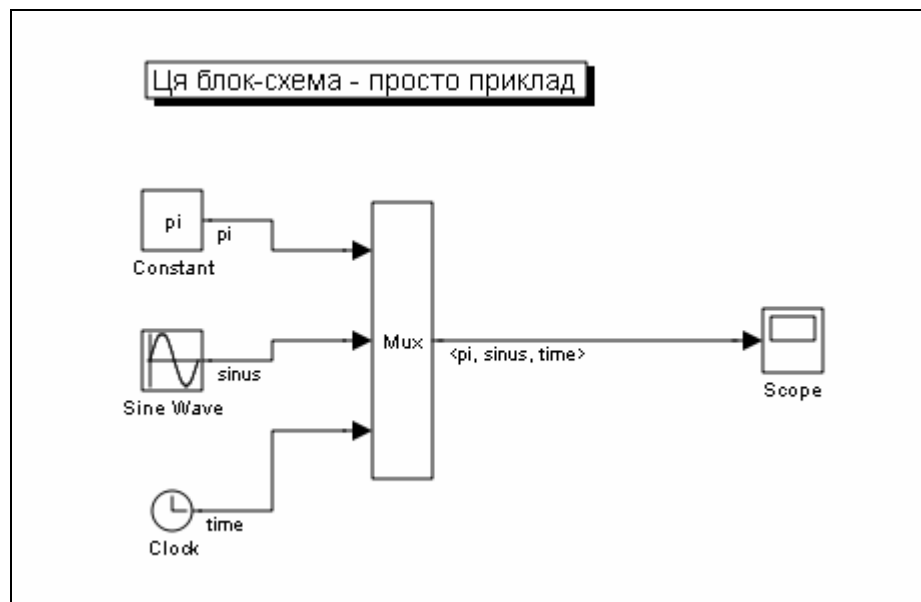


Рис. 7.71. Приклад розміщення коментаря на блок-схемі

Щоб **вилучити коментар**, натисніть і утримуйте клавішу <Shift>, виді- лить коментар і натисніть клавішу <Delete> або <Backspace>.

7.3.5. Створення підсистем

Якщо складність і розміри блок-схеми моделі стають надто великими, її можна суттєво спростити, групуючи блоки у підсистеми. Використання підсистем дає наступні переваги:

- скорочення кількості блоків, що виводяться у вікні моделі;
- об'єднання у єдину групу (підсистему) функціонально пов'язаних блоків;
- можливість створення ієрархічних блок-схем.

Існують три можливості створення підсистем:

- додати блок **Subsystem** у модель, потім увійти у цей блок і створити підсистему у вікні підсистеми, яке при цьому виникне;
- виділити частину блок-схеми моделі й об'єднати її у підсистему.

Створення підсистеми через додавання блоку **Subsystem**

У цьому разі слід зробити наступне:

- скопіювати блок **Subsystem** у вікно моделі, перетягнувши його з поділу **Ports&Subsystems**;
- розкрити вікно блоку **Subsystem**, подвійно клацнувши на зображенні блоку у блок-схемі;
- у порожньому вікні моделі створити підсистему, використовуючи блоки **In** і **Out** для створення входів і виходів підсистеми.

Створення підсистеми через групування існуючих блоків

Якщо блок-схема вже містить блоки, які потрібно об'єднати у підсистему, то останню можна зробити у такий спосіб:

- виділити рамкою блоки і з'єднувальні лінії, які потрібно включити у склад підсистеми (див. рис. 7.72);
- обрати команду **Create Subsystem** із меню **Edit** вікна моделі; при цьому SimuLINK замінить виділені блоки одним блоком **Subsystem** (див. рис. 7.73)

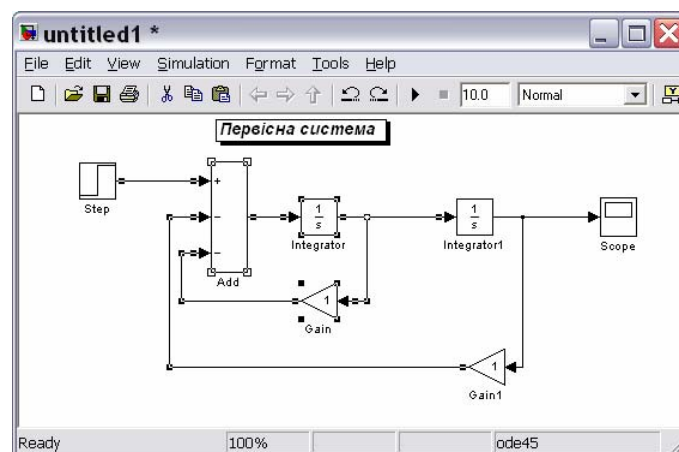


Рис. 7.72. Первісна система з виділеними блоками

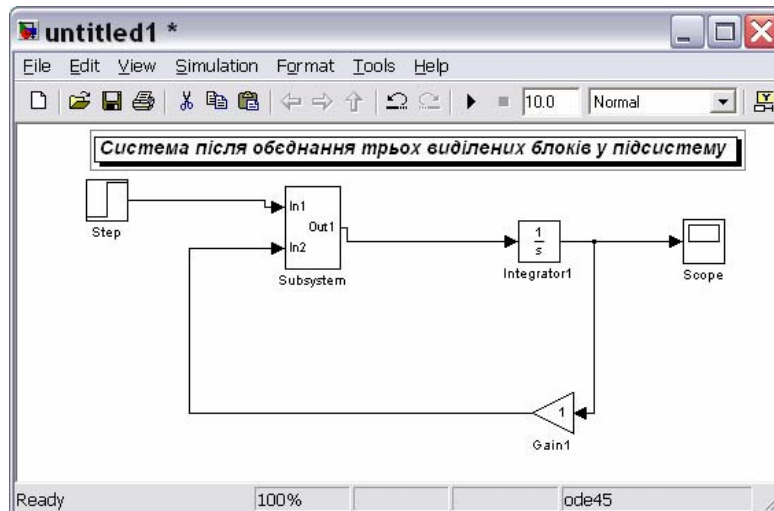


Рис. 7.73. Система після об'єднання виділених блоків у підсистемі

Якщо розкрити вікно блоку **Subsystem**, подвійно клацнувши на ньому, то SimuLINK відобразить блок-схему створеної підсистеми (рис. 7.74). Як видно, SimuLINK додав блоки **In** і **Out** для відображування входів і виходів у систему вищого рівня.

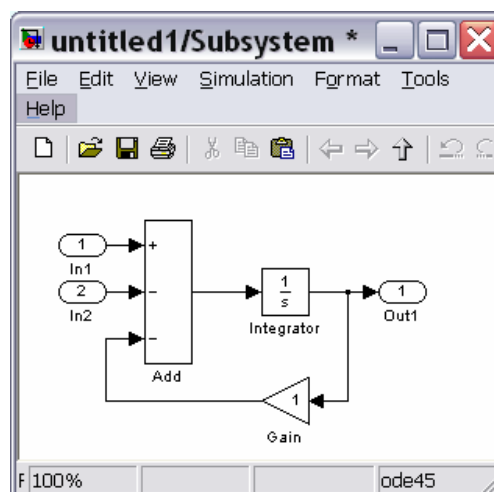


Рис. 7.74. Утворена підсистема

7.3.6. Зберігання і виведення до друку блок-схеми

Для запису моделі (блок-схеми) на диск потрібно викликати команду *Save* (Зберегти) або *Save As* (Зберегти як) з меню *File* (Файл) вікна моделі. Simulink записує у вказану директорію файл з заданим (введеним з клавіатури) ім'ям, присвоюючи йому розширення **.mdl**. При цьому у цей файл автоматично записуються усі вложені підсистеми моделі.

Щоб роздрукувати блок-схему на принтері, потрібно скористуватися командою *File > Print* (Файл > Друк) вікна моделі.

Доволі цікавою і важливою є можливість вставлення блок-схеми у документ Word. Для цього слід використати команду *Copy model to clipboard* (Копіювати модель у буфер) меню *Edit* (Правка) вікна моделі, яка поміщає у буфер

обміну вміст вікна моделі. Якщо після цього перейти у вікно текстового редактора і натиснути клавиші Ctrl+V, у відкритому документі редактора виникне зображення блок-схеми моделі. Саме у такий спосіб отримані рис. 7.68 – 7.71.

7.4. Приклади утворення S-моделей

При утворенні власних S-моделей користувач має вирішити ряд проблем, що виникають у процесі побудови блок-схеми заданої математичної моделі

7.4.1. Модель поведження фізичного маятника

Опишемо процес створення S-моделі на прикладі задачі моделювання поведження фізичного маятника при гармонічній вібрації точки його підвісу.

Користуючись результатами раніше проведених перетворень (див. п. 6.6.2 глави 2), вихідне рівняння руху маятника подамо у такій безрозмірній формі:

$$\varphi'' + \sin \varphi = S(\tau, \varphi, \varphi'), \quad (7.1)$$

де функція $S(\tau, \varphi, \varphi')$ має наступний вид

$$S(\tau, \varphi, \varphi') = -2 \cdot \zeta \cdot \varphi' - [n_{mx} \cdot \sin(\nu \cdot \tau + \varepsilon_x) \cdot \cos \varphi + n_{my} \cdot \sin(\nu \cdot \tau + \varepsilon_y) \cdot \sin \varphi], \quad (7.2)$$

а безрозмірні величини ζ і ν визначаються виразами

$$\zeta = \frac{R}{2 \cdot \sqrt{mgl \cdot J}}; \quad \nu = \frac{\omega}{\omega_0}; \quad \left(\omega_0 = \sqrt{\frac{mgl}{J}} \right).$$

Вхідними (тими, що задаються) параметрами для моделювання вважати- мемо:

- 1) параметри самого маятника; до них у розглядуваному випадку відноситься лише відносний коефіцієнт загасання ζ ;
- 2) параметри, що характеризують зовнішню дію; сюди входять: амплітуди віброперевантажень точки підвісу маятника у вертикальному n_{ym} ш горизонтальному n_{xm} напрямках; відносна (по відношенню до частоти власних малих коливань маятника) частота вібрації основи ν ; початкові фази ε_x і ε_y вібрації основи;
- 3) початкові умови руху маятника, тобто початкове відхилення φ_0 маятника від вертикалі і його безрозмірну кутову швидкість $\varphi'_0 = \dot{\varphi}_0 / \omega_0$.

До вихідних (модельованих) величин віднесемо поточний кут відхилення маятника від вертикалі $\varphi(\tau)$ і його безрозмірну кутову швидкість $\varphi'(\tau)$.

У подальшому застосовуватимемо наступні позначення первісних і шуканих величин у робочому просторі Matlab:

- fi0 – початкове значення φ_0 кута відхилення маятника від вертикалі;
- fit0 – початкова безрозмірна кутова швидкість φ'_0 маятника;

dz - відносний коефіцієнт загасання ζ ;
 nmx - амплітуда n_{xm} віброперевантаження у горизонтальному напрямку;
 nmy - амплітуда n_{ym} віброперевантаження у вертикальному напрямку;
 nu - відносна частота ν вібрації основи;
 ex - початкова фаза ε_x віброприскорення у горизонтальному напрямку;
 ey - початкова фаза ε_y віброприскорення у вертикальному напрямку.

Відмітимо одну вельму важливу особливість взаємодії робочого простору Matlab і середовища Simulink.

УВАГА. *Увесь робочий простір системи Matlab є досяжним для виконуваних S-моделей середовища Simulink. Тому при запису значень параметрів настроювання S-блоків можна замість числових значень вводити імена (ідентифікатори) змінних, існуючих на цей час у робочому просторі Matlab, і навіть вирази, записані M-мовою.*

Це значно полегшує складання блок-схем, оскільки ви можете у робочому просторі формувати масив змінних, а потім при складанні блок-схем виражати через ці змінні значення параметрів настроювання блоків.

Перш ніж приступитися до складання S-моделі, запишемо рівняння (7.1) в іншій формі:

$$\varphi'' = S(\tau, \varphi, \varphi') - \sin \varphi. \quad (7.3)$$

При побудові блок-схеми, що відповідає цьому рівнянню, розмірковуватимемо у такий спосіб.

Припустимо, що процеси $\varphi(\tau)$ і $\varphi'(\tau)$ є відомими. Тоді, здійснюючи операції з ними у відповідності до правої частини рівняння (7.3), можна віднайти й кутове прискорення $\varphi''(\tau)$. Якщо потім проінтегрувати прискорення, можна отримати кутову швидкість $\varphi'(\tau)$. Нарешті, проінтегрувавши і її, можна одержати кут $\varphi(\tau)$ як функцію часу. Дві останні величини (процеси) можна тепер використовувати для формування правої частини рівняння (7.3).

Тому при формуванні бло-схеми, що здійснює чисельне інтегрування диференційного рівняння (7.3), вчинимо у такий спосіб. В основу блок-схеми покладемо два послідовно з'єднаних інтегратори (блоки **Integrator**). Нав хід першого інтегратора подамо кутове прискорення, а у якості початкової умови використаємо початкове значення кутової швидкості φ'_0 . Виходом першого інтегратора буде поточна кутова швидкість $\varphi'(\tau)$. Цю величину слід подати на вхід другого інтегратора з початковою умовою у виді початкового значення кута φ_0 . Виходом другого інтегратора буде шуканий процес $\varphi(\tau)$.

Оформимо описану частину блок-схеми у виді підсистеми. Для цього у поронє вікно блок-схеми (яку назвемо FM) перетягнемо з поділу *Ports&Subsystems* блок **Subsystem** і двічі клацнемо на ньому. У вікні, що виникне, складемо блок-схему підсистеми (рис. 7.75) й назвемо її **Pendulum**.

При настроюванні першого блоку *Integrator* у якості значення його параметра *Initial condition* вкажемо fit_0 . У другому інтеграторі цьому параметру присвоїмо значення fi_0 .

Вихідними величинами створеної підсистеми є поточний кут $\varphi(\tau)$ відхилення маятника від вертикалі і його поточна безрозмірна кутова швидкість $\varphi'(\tau)$.

Використовуючи їх, можна тепер аналогічно сформулювати підсистему, яка обчислює (у відповідності до виразу (7.2)) значення безрозмірного "моменту" $S(\tau, \varphi, \varphi')$ зовнішніх сил, що діють на маятник при вібрації точки його підвісу.

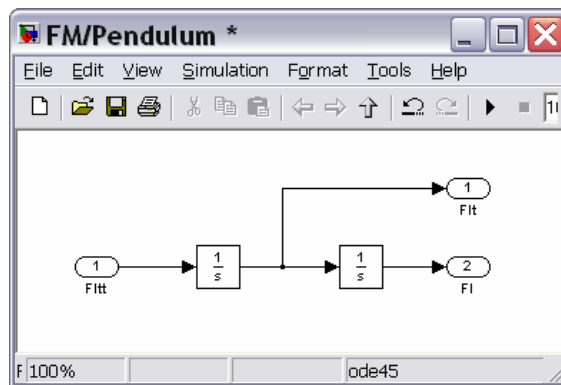


Рис. 7.75. Підсистема Pendulum

Назвемо нову підсистему *MomentsFM* (рис. 7.76).

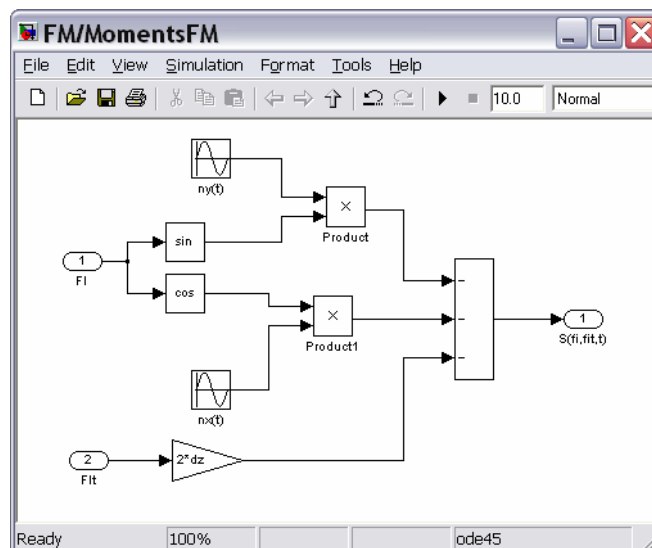


Рис. 7.76. Підсистема MomentsFM

Вона формує поточні значення виразу $S(\tau, \varphi, \varphi')$ по поточних значеннях кута й кутової швидкості маятника, які подаються на її вхід. Її вихідний порт видає величину $S(\tau, \varphi, \varphi')$ моменту сил, що діють на маятник.

Можна помітити, що величина dz (відносний коефіцієнт загасання ζ) використаний у блоці *Gain*, що розташований внизу блок-схеми. Окрім того, при формуванні перевантажень вдовж горизонтальної і вертикальної осей (блоки

$nx(t)$ і $ny(t)$) у параметрах настроювання використані змінні nm_x , nm_y , nu , ex і ey (рис 7.77).

Тепер, використовуючи дві створені підсистеми можна скласти головну блок-схему, яка реалізує моделювання шляхом чисельного інтегрування диференційного рівняння (7.3). Її зображено на рис. 7.78.

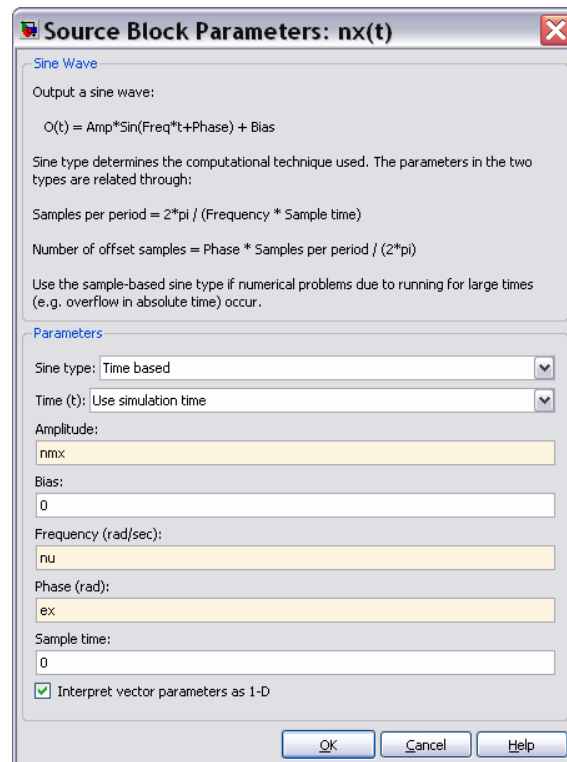


Рис. 7.77. Вікно настроювання блоку $nx(t)$

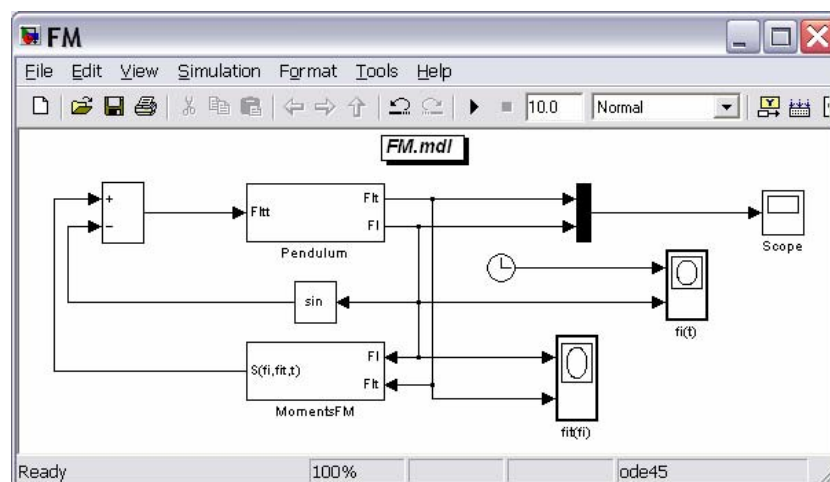


Рис. 7.78. Головна блок-схема FM

Але для здійснення чисельного інтегрування попередньо потрібно встановити програму розв'язувача, яка реалізовуватиме той чи інший метод чисельного інтегрування.

Для цього перед початком моделювання викличемо з вікна блок-схеми командою *Simulation > Configuration Parameters* вікно з тією самою назвою і

встановимо в ньому вкладинку *Solver*. Встановимо у полі Stop time (Час завершення) значення $2 \cdot \pi \cdot 20$, що відповідає інтервалу безрозмірного часу у двадцять повних періодів власних малих коливань маятника. У полі Type задамо тип розв'язувача з фіксованим кроком інтегрування (Fixed-step), і у полі Fixed-step size (Розмір фіксованого кроку) – значення кроку 0,01. У полі *Solver* оберемо конкретний метод інтегрування – ode4 (Runge-Kutta). У цілому це вікно набуде виду, поданому на рис. 7.79.

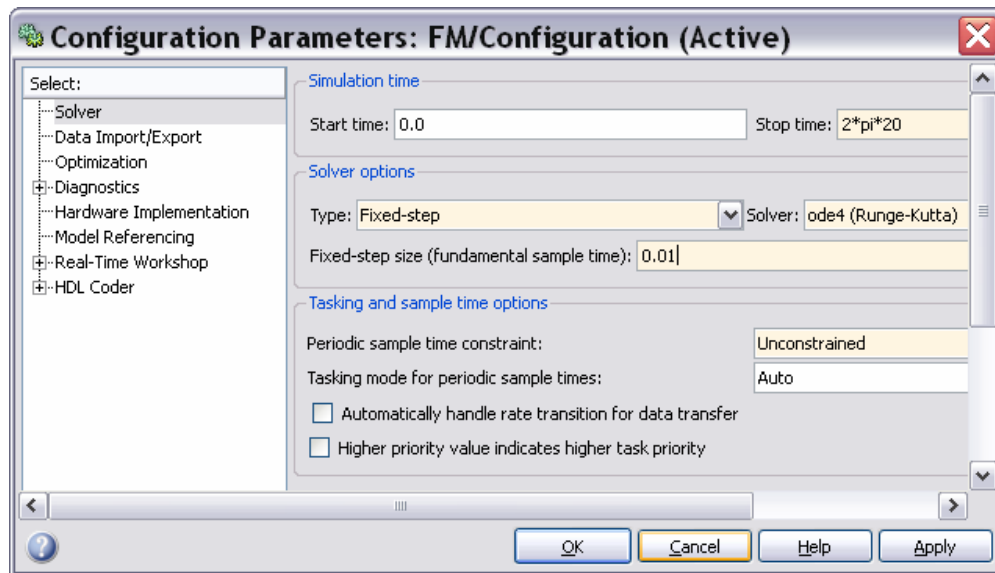


Рис. 7.79. Вкладинка *Solver* вікна *Configuration Parameters*

Тепер практично усе готово до моделювання. Залишилося лише забезпечити присвоювання значень усім змінним, які були використані у параметрах налаштування блоків. Для цього складемо невеличку програму FM_dat.m:

```
% FM_dat
% Програма присвоювання значень даним, що використовуються
% у S-моделі FM.mdl
```

```
clear all, clc
dz = 0.1; fi0 = 1*pi/180; fit0 = 0;
nmy = 1; nmх = 0; ex = 0; ey = 0; nu = 6.3;
```

Дані, що введені у файлі FM_dat, відповідають вертикальній вібрації основи з частотою, яка у 2,3 більша за частоту власних малих коливань маятника, амплітудою 1g по прискоренню і при початковому відхиленні маятника від вертикалі 160° .

Викликавши програму FM_dat з командного вікна Matlab, а потім й S-модель *FM* на моделювання з вікна її блок-схеми, отримаємо у наглядних вікнах моделі результати, подані на рис. 7.80.

У вікні блоку *fi(t)* (див. рис. 7.80а) виведений графік залежності кута повороту маятника від часу. У вікні блоку *fit(fi)* (див. рис. 7.80б) зображений фазовий портрет маятника при обраних параметрах маятника і збурень, а у вікні

блоку *Scope* (див. рис. 7.80в) подані графіки залежності кута й кутової швидкості від часу.

Результати повністю відповідають одержаним раніше (п. 6.7.1, рис. 6.21) і ілюструють виникнення параметричних незгасаючих коливань маятника при вертикальній вібрації точки його підвісу.

Змінюючи дані настроювання у файлі FM_dat, можна проводити дослідження поведінки маятника при різних значеннях вхідних параметрів.

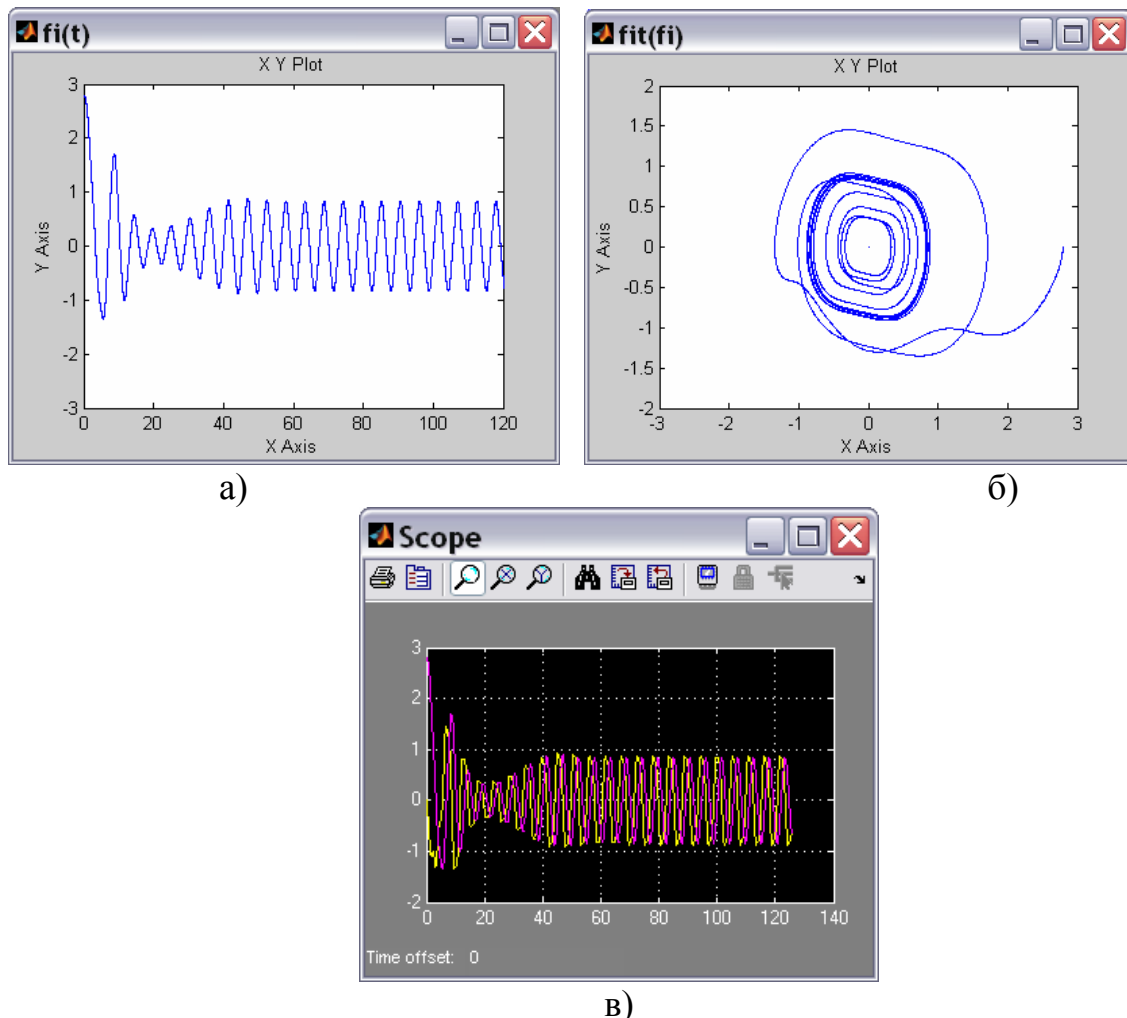


Рис. 7.80. Результати моделювання у різних оглядових вікнах моделі FM: а) у вікні блоку $fi(t)$; б) у вікні блоку $fst(fi)$; в) у вікні блоку *Scope*

З цього прикладу стають наочними переваги й недоліки моделювання динамічних систем за допомогою пакету Simulink у порівнянні з аналогічними дослідженнями з використанням лише програм Matlab.

До переваг застосування Simulink відносяться такі:

- складання блок-схеми часом є набагато більш простим і наочним процесом, аніж написання програм, що реалізують ті самі операції; внаслідок цього виникнення помилок є менш вірогідним; це посилюється наявністю добре відпрацьованих і надійних блоків-програм, які реалізують типові, часом доволі складні, математичні операції і процедури, які часто зустрічаються при практи-

чному моделюванні і відповідають реальним особливостям поведінки динамічних систем;

- блок-схема рівнянь руху є значно більш наочною, аніж код процедури обчислення правих частин диференціальних рівнянь і не потребує приведення рівнянь до форми Коші; завдяки цьому стає більш зрозумілим фізичний сенс окремих блоків і їх взаємозв'язків;

- при проведенні моделювання у середовищі Simulink зникає потреба в організації процесу чисельного інтегрування диференціальних рівнянь і виведення результатів у графічній формі, – цей процес у Simulink автоматизований;

- особливо корисним є наявність значної кількості розв'язувачів – програм різних методів чисельного інтегрування, – і можливість довільного їх обрання.

Недоліки Simulink пов'язані, головним чином, з недостатніми можливостями і гнучкістю виведення результатів у графічній формі:

- неможливо додати власні написи у заголовок графіків у наглядних вікнах і по осях графіків;

- неможливо встановити сітку координатних ліній у вікні блоку *XY-Graph*;

- незручністю застосування оглядового вікна *XYGraph* є необхідність попереднього встановлення діапазонів змінювання обох вхідних величин по осях графіка; якщо ці діапазони встановлені невірно, в оглядовому вікні може взагалі не виникнути зображення графіка, або виникне такий його фрагмент, за яким неможливо зробити правильний висновок щодо поведінки системи; а при дослідженні системи часто неможливо заздалегідь передбачити діапазони змінювання величин;

- відсутні засоби виведення текстової інформації на поле графіка – це робить графічне подання безадресним.

Ці недоліки є суттєвими. Вони можуть бути подолані поєднанням можливостей Simulink і Matlab. Наприклад, можна записати отримані значення вихідних величин у MAT-файл (надсилаючи їх на блок *To File*), потім скласти і використати програму, яка здійснювала би зчитування даних, записаних у MAT-файлі, і формування на цій основі графічного зображення у вікні фігури на зразок того, як це зроблено у п. 6.4. Такий спосіб моделювання реалізований у наступному прикладі.

7.4.2. Моделювання руху трьох тіл під дією сил гравітації

Поглянемо на класичну задачу небесної механіки – визначення руху трьох матеріальних тіл під дією сил взаємної гравітації – з точки зору чисельного моделювання цього руху у середовищі Simulink.

Розглянемо три ізольовані матеріальні точки з масами відповідно m_1 , m_2 і m_3 . Позначимо радіус-вектори цих точок відносно деякої нерухомої в інерці-

льному просторі точки O через \mathbf{R}_1 , \mathbf{R}_2 і \mathbf{R}_3 . Диференційні рівняння руху цих трьох точок можуть бути записані у наступному виді:

$$\begin{cases} m_1 \frac{d^2 \mathbf{R}_1}{dt^2} = \gamma \frac{m_1 m_2}{R_{21}^3} \mathbf{R}_{21} - \gamma \frac{m_1 m_3}{R_{31}^3} \mathbf{R}_{13} \\ m_2 \frac{d^2 \mathbf{R}_2}{dt^2} = -\gamma \frac{m_1 m_2}{R_{21}^3} \mathbf{R}_{21} + \gamma \frac{m_2 m_3}{R_{32}^3} \mathbf{R}_{32} \\ m_3 \frac{d^2 \mathbf{R}_3}{dt^2} = -\gamma \frac{m_3 m_2}{R_{32}^3} \mathbf{R}_{32} + \gamma \frac{m_1 m_3}{R_{31}^3} \mathbf{R}_{13} \end{cases} \quad (7.4)$$

де позначено $\mathbf{R}_{21} = \mathbf{R}_2 - \mathbf{R}_1$; $\mathbf{R}_{32} = \mathbf{R}_3 - \mathbf{R}_2$; $\mathbf{R}_{13} = \mathbf{R}_1 - \mathbf{R}_3$.

Дослідження рівнянь руху (і особливо чисельне) завжди зручніше здійснювати за рівняннями, приведеними до безрозмірної форми, - у цьому випадку скорочується кількість параметрів, від яких залежить розв'язок.

Приведемо рівняння (7.4) до безрозмірної форми. Для цього у якості базових параметрів використаємо такі фізичні величини:

- γ – гравітаційна стала, що має розмірність $L^3 M^{-1} T^{-2}$ (L – одиниця довжини, M – одиниця маси, T – одиниця часу);

- m_1 – маса першого тіла, яке вважатимемо основним; ним зазвичай є найбільш масивне тіло (наприклад, Сонце, яке знаходиться під дією гравітаційного тяжіння Землі (друге, середнє за масою, тіло) і Місяця (третє, найменш масивне тіло)); розмірність - M ;

- R_{210} – початкове значення відстані між першим і другим тілами, розмірність – L .

Тепер введемо безрозмірні величини:

1) безрозмірна маса першого тіла:

$$\mu_1 = \frac{m_1}{m_1} = 1;$$

2) безрозмірна маса другого тіла:

$$\mu_2 = \frac{m_2}{m_1}; \quad (7.5)$$

3) безрозмірна маса третього тіла:

$$\mu_3 = \frac{m_3}{m_1}; \quad (7.6)$$

4) безрозмірні довжини радіус-векторів:

$$\rho_1 = \frac{R_1}{R_{210}}; \quad \rho_2 = \frac{R_2}{R_{210}}; \quad \rho_3 = \frac{R_3}{R_{210}}; \quad (7.7)$$

5) безрозмірні радіус-вектори

$$\mathbf{r}_i = \frac{\mathbf{R}_i}{R_{210}}; \quad \mathbf{r}_{ij} = \frac{\mathbf{R}_{ij}}{R_{210}}; \quad (7.8)$$

6) безрозмірний час

$$\tau = t \sqrt{\frac{\gamma m_1}{R_{210}^3}}. \quad (7.9)$$

Остання формула означає, що у якості одиниці вимірювання часу використовується величина

$$T_0 = 2\pi \sqrt{\frac{R_{210}^3}{\gamma m_1}}. \quad (7.10)$$

Для системи Сонце-Земля-Місяць вона дорівнює року.

Прийнятий безрозмірний час є таким, що безрозмірний період кругового обертання другого тіла навколо першого дорівнює 2π .

З врахуванням цього рівняння (7.4) у безрозмірній формі набудуть такого виду:

$$\begin{cases} \frac{d^2 \mathbf{r}_1}{dt^2} = \frac{\mu_2}{r_{21}^3} \mathbf{r}_{21} - \frac{\mu_3}{r_{31}^3} \mathbf{r}_{13} \\ \frac{d^2 \mathbf{r}_2}{dt^2} = -\frac{1}{r_{21}^3} \mathbf{r}_{21} + \frac{\mu_3}{r_{32}^3} \mathbf{r}_{32} \\ \frac{d^2 \mathbf{r}_3}{dt^2} = -\frac{\mu_2}{r_{32}^3} \mathbf{r}_{32} + \frac{1}{r_{31}^3} \mathbf{r}_{13} \end{cases} \quad (7.11)$$

Три вектори \mathbf{r}_{21} , \mathbf{r}_{32} і \mathbf{r}_{13} пов'язані між собою наступним співвідношенням

$$\mathbf{r}_{21} + \mathbf{r}_{32} + \mathbf{r}_{13} = \mathbf{0}. \quad (7.12)$$

Тому з трьох рівнянь (7.11) незалежними є лише двоє. Щоб їх одержати, віднімемо з другого рівняння перше, а з третього рівняння – друге і вилучимо з отриманих рівнянь вектор \mathbf{r}_{13} , користуючись (7.12). Одержимо шукані два рівняння

$$\begin{cases} \frac{d^2 \mathbf{r}_{21}}{dt^2} = -\left(\frac{1 + \mu_2}{\rho_{21}^3} + \frac{\mu_3}{\rho_{31}^3}\right) \mathbf{r}_{21} + \mu_3 \left(\frac{1}{\rho_{32}^3} - \frac{1}{\rho_{31}^3}\right) \mathbf{r}_{32} \\ \frac{d^2 \mathbf{r}_{32}}{dt^2} = -\left(\frac{\mu_3 + \mu_2}{\rho_{32}^3} + \frac{1}{\rho_{31}^3}\right) \mathbf{r}_{32} + \left(\frac{1}{\rho_{21}^3} - \frac{1}{\rho_{31}^3}\right) \mathbf{r}_{21} \end{cases} \quad (7.13)$$

Тут використані позначення:

$$\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j; \quad \rho_{ij} = |\mathbf{r}_{ij}|. \quad (7.14)$$

До рівнянь (7.13) слід додати співвідношення, що випливає з (7.12):

$$\rho_{13} = |\mathbf{r}_{21} + \mathbf{r}_{32}|. \quad (7.15)$$

Рівняння (7.13) у сукупності з (7.15) покладемо в основу майбутньої моделі. При чисельному інтегруванні слід взяти до уваги, що $\rho_{21}(0) = 1$.

Використовуватимемо наступні позначення у робочому просторі:

- μ_2 – відношення мас другого і першого тіла (μ_2);
- μ_3 – відношення мас третього і першого тіла (μ_3);

- $X_{210}, Y_{210}, Z_{210}$ – початкові безрозмірні координати другого тіла відносно першого ($x_{21}(0), y_{21}(0), z_{21}(0)$);
- $X_{320}, Y_{320}, Z_{320}$ – початкові безрозмірні координати третього тіла відносно другого ($x_{21}(0), y_{21}(0), z_{21}(0)$);
- $V_{21x}, V_{21y}, V_{21z}$ – початкові безрозмірні проекції вектора швидкості другого тіла відносно першого ($V_{21x}(0), V_{21y}(0), V_{21z}(0)$);
- $V_{32x}, V_{32y}, V_{32z}$ – початкові безрозмірні проекції вектора швидкості третього тіла відносно другого ($V_{32x}(0), V_{32y}(0), V_{32z}(0)$).

Варіант блок-схеми S-моделі під назвою GR_3tel.mdl наведений на рис. 7.81:

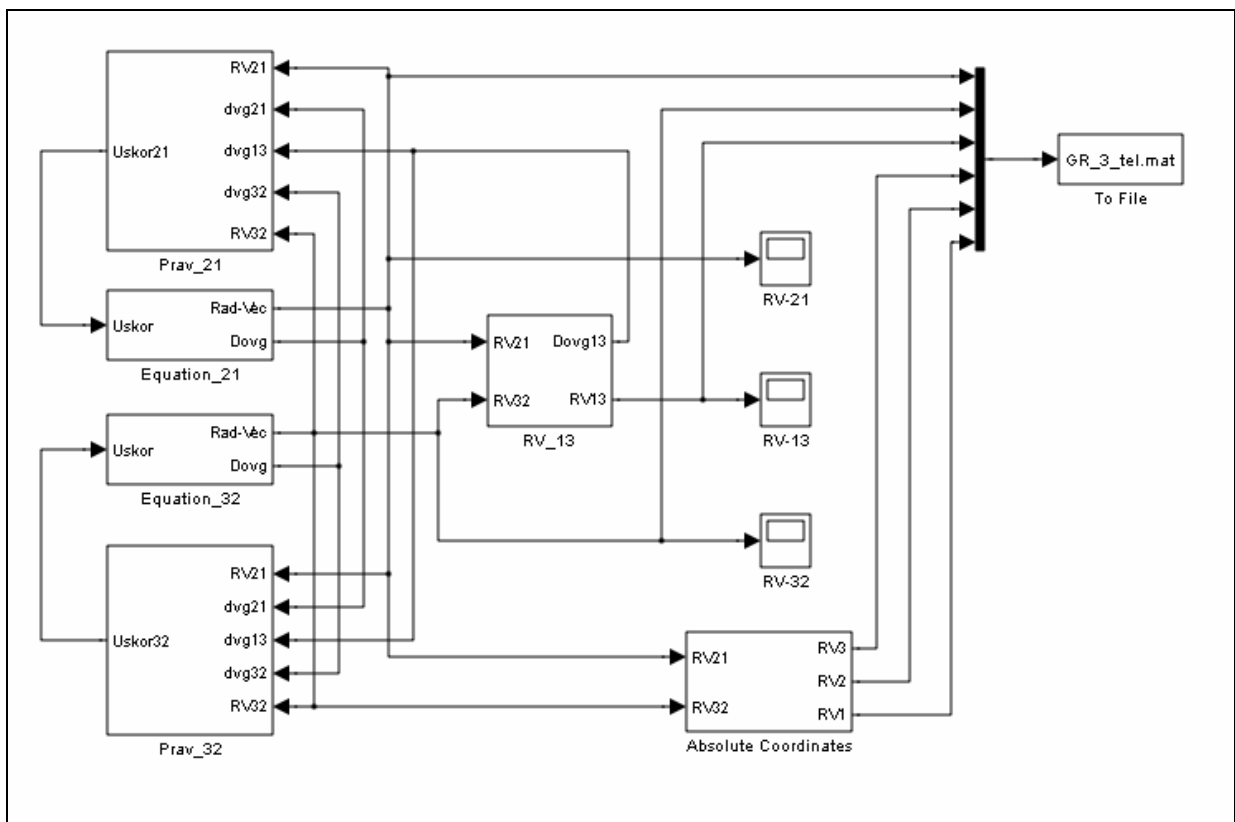


Рис. 7.81. Блок-схема S-моделі GR_3tel.mdl

Вона містить у собі шість підсистем – *Equation_21*, *Equation_32*, *Prav_21*, *Prav_32*, *RV_13*, *Absolute Coordinates* – і блок *To File*. Останній забезпечує запис результатів моделювання у MAT-файл, де вони зберігаються у матриці за ім'ям RV (рис. 7.82).

Основні дії з чисельного інтегрування диференційних рівнянь (7.13) зосереджені у підсистемах *Equation_21* і *Equation_36*. Тут (рис. 7.83) здійснюється подвійне інтегрування прискорень тіл, а також обчислюється поточна відстань між тілами.

Вхідною величиною обох підсистем є векторна величина, елементи якої являють собою проекції вектора прискорення на осі X , Y і Z . Вихідних величин дві. Одна з них – вектор з трьох елементів, що є поточними проекціями ра-

діуса-вектора на ті самі осі. Другий вихід – скалярна величина, довжина цього радіуса-вектора. Блок, який здійснює обчислення довжини реалізований на основі блоку *Fcn* з поділу *User-defined Functions* бібліотеки Simulink.

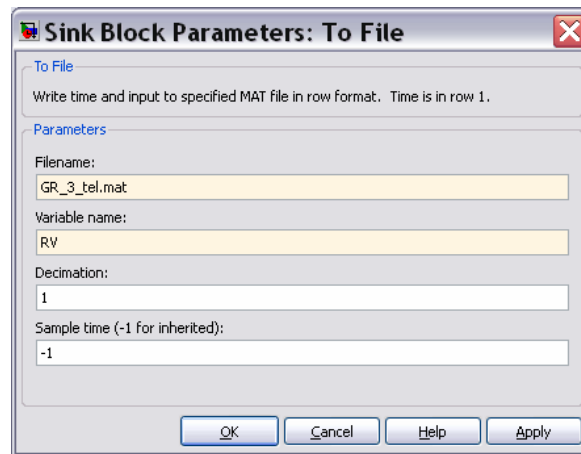


Рис. 7.82. Вікно налаштування блоку *To File*

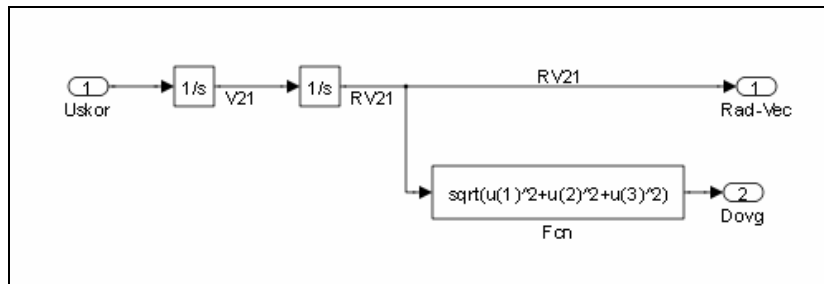


Рис. 7.83. Блок-схема підсистеми *Equation_21*

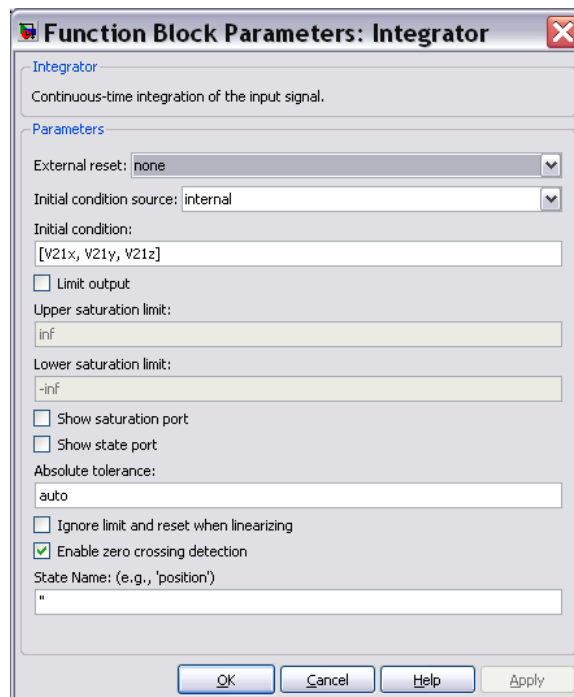


Рис. 7.84. Вікно налаштування першого інтегратора блоку *Equation_21*

Характерною і важливою особливістю цих підсистем, яка відрізняє їх від використаної раніше підсистеми *Pendulum* (див. рис. 7.75) є те, що в них інтегрується не скалярна функція часу, а векторна, яка складається одразу з трьох скалярних величин. Наприклад, у підсистемі *Equation_21* здійснюється одночасне подвійне інтегрування прискорень x''_{21} , y''_{21} і z''_{21} , яке досягається за рахунок лише двох блоків типу *Integrator*. Для цього достатньо у блоці настроювання інтеграторів ввести у якості початкової умови не одну величину, а вектор з трьох величин, кожна з яких є початковою умовою відповідного елемента вектора, який надається на вхід інтегратора. наприклад, вектором початкових умов для першого інтегратора є $[V21x, V21y, V21z]$ (рис. 7.84). Аналогічно, у другому інтеграторі встановлений вектор $[X210, Y210, Z210]$ початкових умов.

Те саме зроблено у підсистемі *Equation_32*, яка обчислює поточні складові вектора \mathbf{r}_{32} , одержані подвійним інтегруванням прискорення \mathbf{r}_{32}'' . Тут в інтеграторах початкові умови встановлені у виді векторів $[V32x, V32y, V32z]$ і $[X320, Y320, Z320]$.

Підсистеми *Prav_21* і *Prav_32*, формують праві частини першого і другого рівнянь (7.13), тобто прискорення у виді вектора з трьох елементів. Вихідна величина у цих підсистемах єдина. Наприклад, у підсистемі *Prav_21* (рис. 7.85) виходом є вектор *Uskor21*, складові якого – проєкції прискорення \mathbf{r}_{21}'' на осі X , Y і Z . Входами є два обчислених раніше вектори $RV21$ і $RV32$, що складені з проєкцій радіус векторів \mathbf{r}_{21} і \mathbf{r}_{32} , і три скалярні величини, що дорівнюють поточним значенням довжин трьох радіусів-векторів, які з'єднують три тіла.

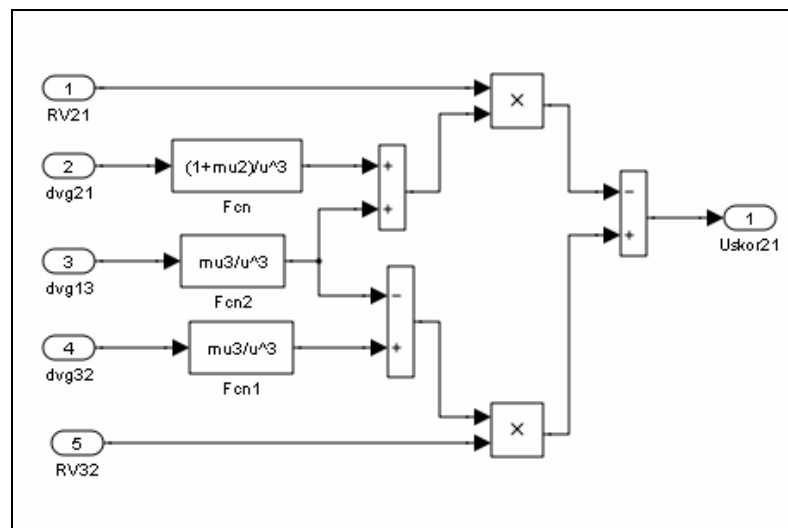


Рис. 7.85. Блок-схема підсистеми *Prav_21*

У підсистемі *Prav_21* використовуються три блоки типу *Fcn* для обчислення коефіцієнтів при векторах у правій частині першого рівняння (7.13) і два блоки *Product* задля перемножування вектора на скалярний коефіцієнт.

Блок-схема підсистеми *RV_13*, яка обчислює вектор \mathbf{r}_{13} і його довжину за відомими векторами \mathbf{r}_{21} і \mathbf{r}_{32} , наведена на рис. 7.86.

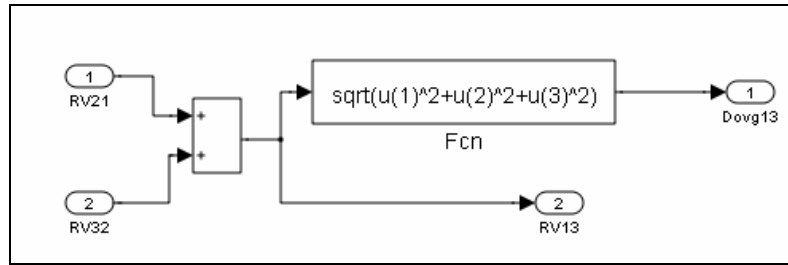
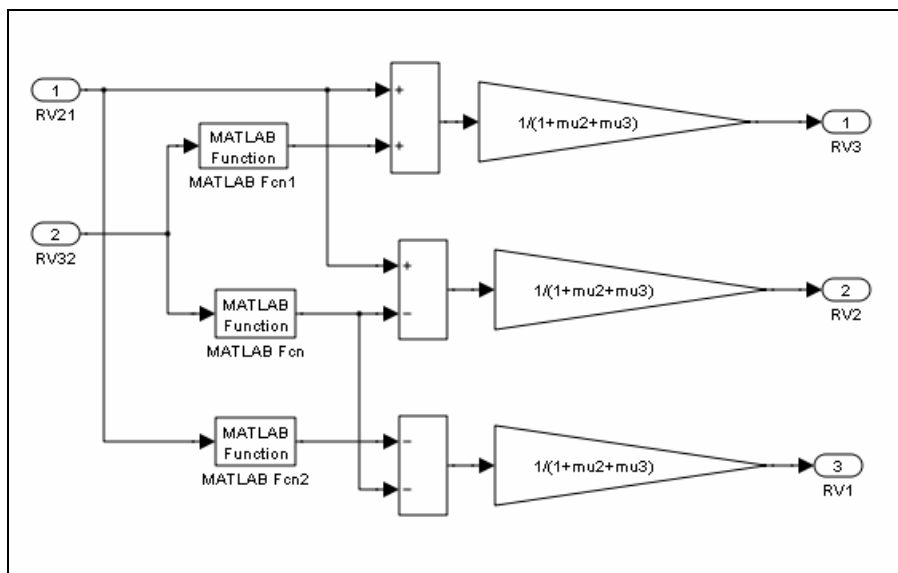


Рис. 7.86. Блок-схема підсистеми RV_13

На рис. 7.87 показана блок-схема підсистеми *Absolute Coordinates*, яка обчислює абсолютні координати першого, другого і третього тіл відносно центру мас системи трьох тіл за формулами:

$$\mathbf{r}_1 = \frac{-(\mu_2 + \mu_3)\mathbf{r}_{21} - \mu_3\mathbf{r}_{32}}{1 + \mu_2 + \mu_3}; \quad \mathbf{r}_2 = \frac{\mathbf{r}_{21} - \mu_3\mathbf{r}_{32}}{1 + \mu_2 + \mu_3};$$

$$\mathbf{r}_3 = \frac{\mathbf{r}_{21} + (1 + \mu_2)\mathbf{r}_{32}}{1 + \mu_2 + \mu_3}.$$

Рис. 7.87. Блок-схема підсистеми *Absolute Coordinates*

Дані, що обчислюються у блок-схемі подаються у блок *To File*. Ці дані записуються на диск у MAT-файл GR_3_tel.mat у виді матриці за ім'ям RV (див. рис. 7.82) у наступному порядку:

- перший рядок утворює масив значень модельного часу, у які обчислені величини, що подаються на вхід;
- другий рядок утворюють значення першого елемента вхідного векторного сигналу (йому відповідає верхній сигнал, що поступає на блок *Mux.*, вихідний сигнал з якого поступає на вхід блоку *To File*), відповідні моментам часу, що записані у першому рядку;
- решта рядків заповнюються значеннями наступних елементів вектора сигналу входу у порядку слідування елементів цього вектора.

Ці особливості варто мати на увазі при зчитуванні даних зі сформованого MAT-файла.

Для того, щоб виконати моделювання по створеній S-моделі і графічно оформити отримані результати, потрібно здійснити наступне.

1. Ввести первісні дані у робочий простір.
2. Запустити S-модель GR_3tel.mdl на моделювання. При цьому результати моделювання будуть записані у MAT-файл GR_3_tel.mat.
3. Завантажити вміст MAT-файла GR_3_tel.mat у робочий простір, ввівши у командному вікні Matlab команду `load GR_3_tel`. При цьому усі дані, що зберігаються у цьому MAT-файлі, будуть записані у робочий простір у виді матриці під ім'ям RV розміром $(1 + 6 \cdot 3) \times n$, де n - розмір вектора значень модельного часу.
4. Використовуючи дані матриці RV, програмно реалізувати виведення результатів у графічне вікно *figure* у графічному виді з відповідним текстовим оформленням.

Усі ці операції можна автоматизувати, запрограмувавши їх у окремій M-програмі, яку зручно оформити у виді M-файла *GR_3tel_upr.m* (керуюча програма для моделі *GR_3tel*. При цьому викликати на моделювання S-модель з програми можна за допомогою команди

```
sim('ім'я S-моделі')
```

Нижче наведений текст керуючої програми.

```
% GR_3TEL_upr
% Програма керування запуском і обробки результатів для S-моделі GR_3TEL

% Лазарєв Ю. Ф. 2-08-2009
clear all, clc
%      1. Введення даних у робочий простір
mu2=0.1; mu3=0.01;
X210=1; Y210=0; Z210=0;
V21x=0; V21y=1; V21z=0;
X320=0.1; Y320=0.0; Z320=0;
V32x=0; V32y=0.0; V32z=1.2;
%      6. Запуск S-моделі
sim('GR_3TEL')
%      3. Завантажування MAT-файла
load GR_3_TEL;
%      4. Присвоювання значень з матриці RV MAT-файла новим змінним
t=RV(1,:);
X21=RV(2,:); Y21=RV(3,:); Z21=RV(4,:);
X32=RV(5,:); Y32=RV(6,:); Z32=RV(7,:);
X13=RV(8,:); Y13=RV(9,:); Z13=RV(10,:);
X3=RV(11,:); Y3=RV(12,:); Z3=RV(13,:);
X2=RV(14,:); Y2=RV(15,:); Z2=RV(16,:);
X1=RV(17,:); Y1=RV(18,:); Z1=RV(19,:);
n=length(t);
%      5. Побудова графіка залежності координат второго тіла від часу
subplot(2,2,1)
plot(t,X21,t,Y21,'--',t,Z21,'.'), grid
set(gca, 'FontSize',14)
title('Рух другого тіла відносно першого')
xlabel('Час (безрозмірний)')
ylabel('Координати (безрозмірні)')
```

```

legend('X_2_1','Y_2_1','Z_2_1',0)
% 6. Побудова графіка залежності координат третього тіла від часу
subplot(2,2,3)
n1=round(n); set(gca, 'FontSize',14)
plot(t(1:n1),X32(1:n1),t(1:n1),Y32(1:n1),'--',t(1:n1),Z32(1:n1),'.'),grid
title('Рух третього тіла відносно другого')
xlabel('Час (безрозмірний)')
ylabel('Координати (безрозмірні)')
legend('X_3_2','Y_3_2','Z_3_2',0)
% 7. Побудова просторових траекторій другого і третього тіл
subplot(4,4,[3,4,7,8,11,12])
plot3(X1,Y1,Z1,'o',X2,Y2,Z2,'.-',X3,Y3,Z3), grid
axis('square')
set(gca, 'FontSize',14)
title('Рух двох тіл у системі трьох гравітуючих тіл','FontSize',16)
xlabel('Координата X')
ylabel('Координата Y')
zlabel('Координата Z')
legend('перше тіло','друге тіло','третє тіло',0)
% 8. Текстова оформлення
subplot(4,4,[15,16]), axis('off')
h= text(-0.1,0.8,'Маси тіл (відносні):', 'FontSize',14);
h= text(0.5,0.8,['\mu-1 = 1; ',sprintf('\mu_2 = %g;',mu2),sprintf('\mu_3 = %g',mu3)], 'Font-
Size',14);
h= text(-0.1,0.650,'Початкові координати (безрозмірні):', 'FontSize',14);
h= text(-0.1,0.50,['другого тіла відносно першого: ',sprintf('X_2_1 = %g;',...
X210),sprintf('Y_2_1 = %g;',Y210),sprintf('Z_2_1 = %g;',Z210)], 'FontSize',14);
h= text(-0.1,0.35,['третього тіла відносно другого: ',sprintf('X_3_2 = %g;',...
X320),sprintf('Y_3_2 = %g;',Y320),sprintf('Z_3_2 = %g;',Z320)], 'FontSize',14);
h= text(0,0.20,'Початкові швидкості (безрозмірні):', 'FontSize',14);
h= text(-0.1,0.05,['другого тіла відносно першого: ',sprintf('V_2_1_x = %g;',...
V21x),sprintf('V_2_1_y = %g;',V21y),sprintf('V_2_1_z = %g;',V21z)], 'FontSize',14);
h= text(-0.1,-0.1,['третього тіла відносно другого: ',sprintf('V_3_2_x = %g;',...
V32x),sprintf('V_3_2_y = %g;',V32y),sprintf('V_3_2_z = %g;',V32z)], 'FontSize',14);
h= text(-0.1,-0.2,'');
h= text(-0.1,-0.3,'Програма GR-3TEL-upr Автор - Лазарєв Ю. Ф. 2-08-2009');
h= text(-0.1,-0.4,'');

```

Запустивши цю програму, можна одразу одержати результати моделювання у такому виді, як показано на рис. 7.88-7.90.

Подані результати відповідають трьом випадкам руху:

- 1) орбітальні рухи другого тіла відносно першого і третього відносно другого здійснюються в одній площині (XY) і в одному напрямку (орбітальні моменти спрямовані однаково – вздовж осі Z);
- 2) вказані орбітальні рухи здійснюються в одній площині, але у протилежних напрямках (орбітальні моменти спрямовані протилежно);
- 3) вказані орбітальні рухи здійснюються у перпендикулярних площинах.

В усіх трьох випадках відношення мас першого, другого і третього тіл дорівнює 100:10:1. Відносні початкові швидкості тіл у всіх випадках прийняті однаковими за величиною ($|V_{21}(0)|=1$ і $|V_{32}(0)|=1$). Зверніть увагу на

відмінність у параметрах орбіти середнього (другого) тіла і періодах його орбітального руху.

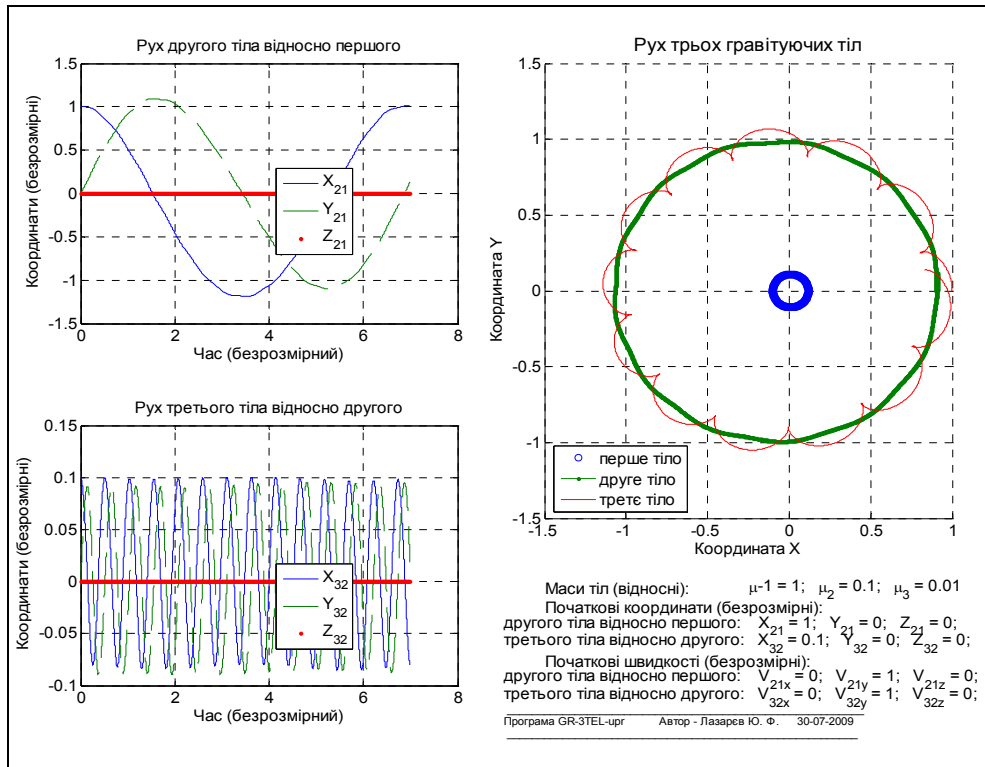


Рис. 7.88. Плаский орбітальний рух, коли тіла обертаються в одному напрямку

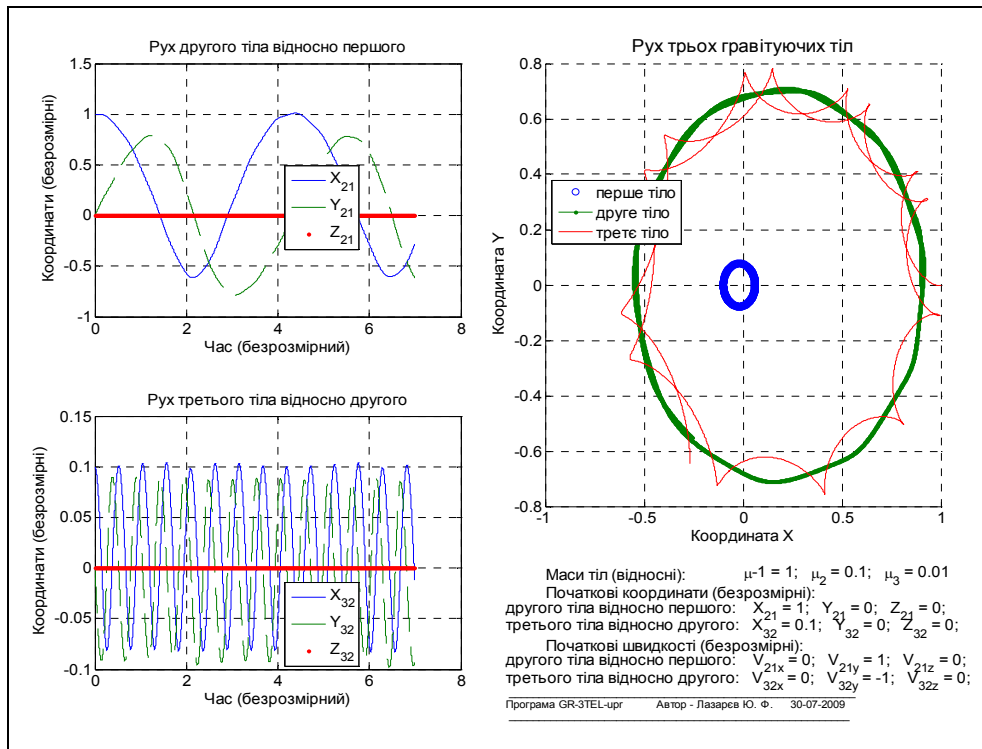


Рис. 7.89. Плаский орбітальний рух, коли тіла обертаються у протилежному напрямку

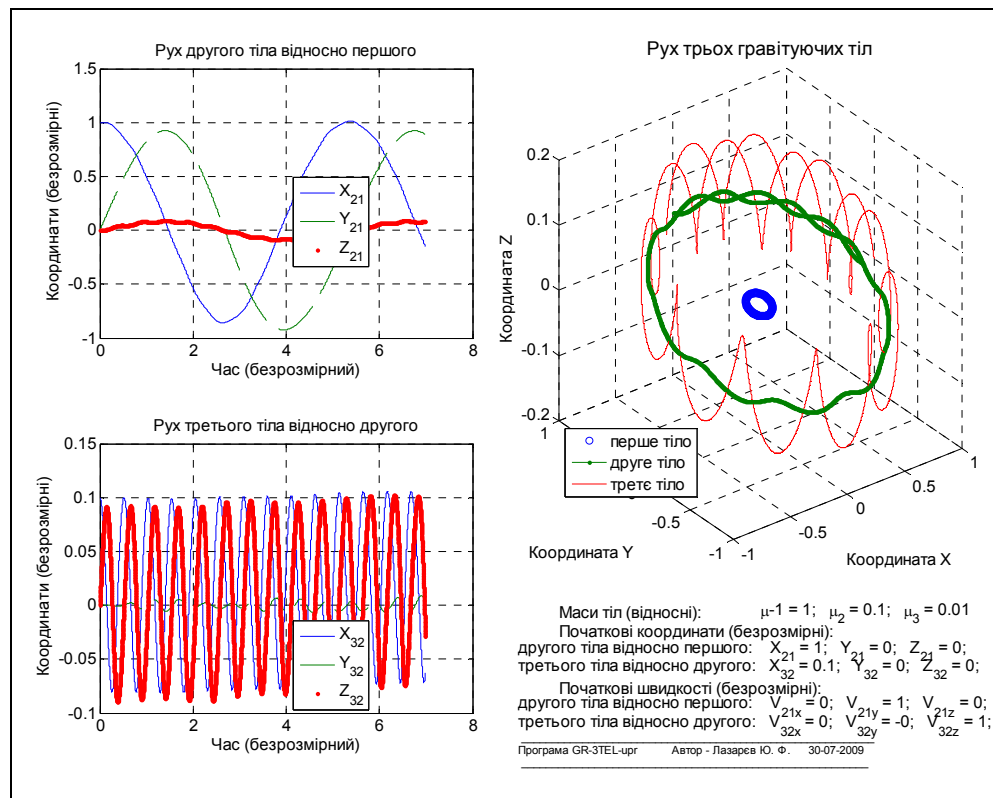


Рис. 7.90. Орбітальний рух, коли площини обертання перпендикулярні

7.5. Запитання для самоперевірки

1. Які переваги має використання пакету Simulink для вирішення обчислювальних задач у порівнянні з застосуванням програмування безпосередньо у середовищі Matlab?
6. Блоки яких поділів бібліотеки Simulink мають обов'язково бути присутніми у блок-схемі будь-якої моделі?
3. Яке головне призначення блоків поділу Source бібліотеки Simulink?
4. Яке головне призначення блоків поділу Sinks бібліотеки Simulink?
5. Блоки якого поділу бібліотеки Simulink забезпечують користувачеві можливість створювати власні блоки?
6. Що таке підсистема і як її утворити?
7. У чому полягають переваги у використанні підсистем?

7.6. Література

1. Лазарев Ю. Ф. Початки програмування у середовищі MatLab. Навч. посібник. – К.: Корнійчук, 1999. – 160 с.
6. Лазарев Ю. Ф. MatLAB 5.x. – К.: Издат. группа BHV, 2000. - 384 с.
3. Лазарев Ю. Ф. Моделирование процессов и систем в MATLAB. Учебный курс. – СПб.: Питер; Киев: Издат.группа BHV, 2005. – 512 с.
4. Лазарев Ю. Ф. Моделювання на ЕОМ. Навч. посібник. – К.: Корнійчук, 2007. = 290 с.

8. ЗАСОБИ ВЗАЄМОДІЇ MATLAB З SIMULINK

Моделювання процесов за допомогою S-моделей поряд зі значними перевагами має й деякі суттєві недоліки.

До переваг використання S-моделей можна віднести:

- ефективність створення програм моделювання складних динамічних систем шляхом складання блок-схеми системи з стандартних готових блоків, які є візуальними відображеннями відповідних математичних програм (візуальне програмування);
- зручні і наочні засоби, що дозволяють перетворити готову блок-схему, або одержати додаткову інформацію про змінювання проміжних процесів;
- широкий набір ефективних програм розв'язувачів (Solvers), які реалізують методи чисельного інтегрування диференційних рівнянь з фіксованим кроком інтегрування, з автоматично змінюваним змінним кроком інтегрування, а також розв'язувачів для так званих жорстких систем диференційних рівнянь;
- відсутність необхідності у спеціальній організації процесу чисельного інтегрування диференційних рівнянь;
- унікальні можливості щодо інтегрування рівнянь нелінійних систем з суттєвими нелінійностями (когда нелінійна залежність має стрибкоподібний характер);
- можливість вельми швидкого і зручного отримання графічної інформації про змінювання модельованих величин з часом.

Недоліками використання S-моделей є:

- жорстка і незручна форма графічного подання сигналів у блоках – оглядових вікнах – *Scope* і *XYGraph* (на відміну від засобів, що використовуються у середовищі Matlab);
- відсутність можливості автоматично (програмно) обробити одержані результати багаторазового моделювання однієї чи кількох S-моделей;
- відсутність можливості раціонально (наприклад, у діалоговій формі) організувати процес змінювання первісних даних S-моделі і параметрів її блоків.

Важливо відзначити також, що для окремих видів диференціальних рівнянь набагато простіше, зручніше і швидше скласти процедури для обчислення їхніх правих частин, чим сформувати відповідну блок-схему.

З викладеного випливає, що програмна реалізація процесу моделювання і моделювання шляхом створення S-моделей маю взаємодоповнювальними можливостями. Тому бажано скористатися перевагами цих двох засобів моделювання, поєднавши програмну реалізацію з застосуванням S-моделей.

8.1. Об'єднання S-моделей з програмами Matlab

Щоб здійснити поєднання програми Matlab з S-моделлю, необхідно мати наявності засоби, які дозволяють забезпечити:

- передавання даних з середовища Matlab у S-модель і зворотно;
- запуск процесу моделювання S-моделі з середовища Matlab, а також можливість змінювання параметрів моделювання і S-блоків з цього середовища;
- виклик програм Matlab з S-моделі;
- створення S-блоків не тільки із інших готових блоків, а і шляхом використання програм, записаних на M-мові.

8.1.1. Керування процесом моделювання у Simulink

Кожний блок S-моделі має такі внутрішні характеристики (рис. 8.1):

- вектор вхідних величин u ;
- вектор вихідних величин y ;
- вектор змінних стану x .

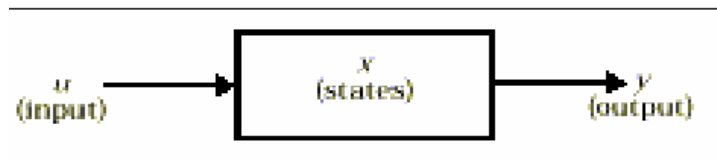


Рис. 8.1. Схема взаємодії величин, що визначають поточний стан блоку

Вектор змінних стану може складатися зі змінних x_c неперервних станів, змінних x_d дискретних станів, або їх комбінації.

Математичні зв'язки між цими величинами можуть бути подані у виді наступних рівнянь:

- формування виходу:

$$y = f_o(t, x, u);$$

- оновлення (формування нового значення) змінних дискретних станів:

$$x_d(k+1) = f_u(t, x, u),$$

- формування значень похідної від вектору змінних стану:

$$\frac{dx}{dt} = f_d(t, x, u),$$

де

$$x = \begin{cases} x_c \\ x_d(k) \end{cases}.$$

Моделювання складається з двох фаз – ініціалізації і власне моделювання. У фазі ініціалізації виконуються наступні дії:

1) параметри блоків передаються у Matlab задля оцінювання (обчислення); результати числових операцій використовуються як фактичні параметри блоків;

2) ієрархія моделі згладжується: кожна не умовно виконувана підсистема замінюється блоками, з яких вона складається;

3) блоки сортуються у тому порядку, в якому їх потрібно змінювати; алгоритм сортування забезпечує такий порядок, що будь-який блок з прямим підключенням не змінюється, поки змінюються блоки, які визначають вхідні величини; на цьому кроку виявляються алгебричні цикли;

4) перевіряються зв'язки між блоками (перш за все збіжність довжини вектора вихідних величин кожного блоку з очікуваною довжиною векторів вхідних величин керованих ними блоків).

Власне моделювання здійснюється шляхом чисельного інтегрування. Кожний з наявних методів інтегрування (ODE) залежить від здатності моделі визначати похідні її неперервних станів. Розрахунок цих похідних здійснюється у два етапи. Спочатку кожна вихідна величина блоку обчислюється у порядку, визначеному у процесі сортування. На другому етапі обчислюються похідні кожного блоку для поточного моменту часу, вхідні змінні і змінні стану. Отриманий вектор похідних використовується для обчислення нового вектора змінних стану у наступний момент часу. Як тільки завершується обчислення нового вектору змінних стану, блоки даних і блоки – оглядові вікна оновлюються.

З переліком програм розв'язувачів (інтеграторів), що прикладаються до пакету Simulink, можна ознайомитися у вікні Configuration Parameters (рис. 8.2), яке виникає на екрані після виклику команди Simulation > Configuration Parameters з меню блок-схеми.

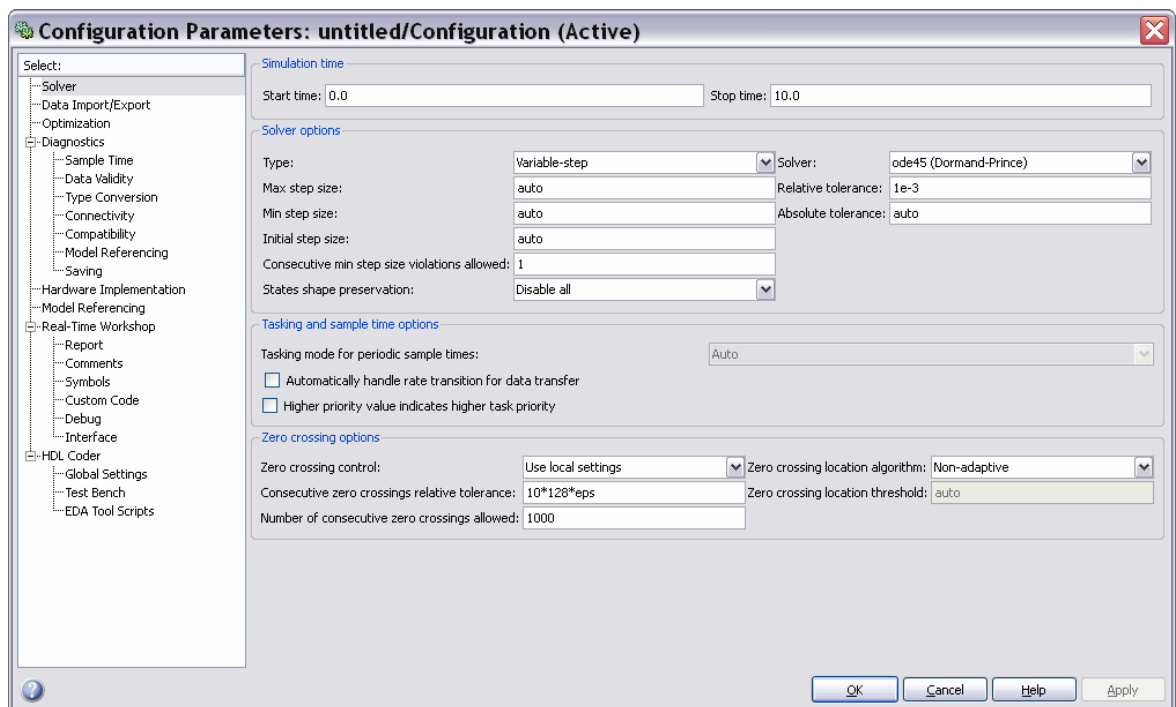


Рис. 8.6. Вкладення Solver вікна Configuration Parameters

У верхній частині вкладки Solver вікна Configuration Parameters містяться поля введення Start time (Початковий час) і Stop time (Кінцевий час), в яких встановлюється відповідно початкове і кінцеве значення аргументу (часу). В

області Solver options (Параметри розв'язувача) у списку *Type* (Тип) обирається тип розв'язувачів, а у спадному списку *Solver* (Розв'язувач) праворуч від нього - конкретний розв'язувач.

Якщо обраний тип розв'язувачів *Fixed-step* (з фіксованим кроком), у списку праворуч виникне такий набір розв'язувачів:

- *discrete (no continuous states)* – дискретний (не неперервні стани);
- *ode5 (Dormand-Prince)* – метод Дормана-Принса (пятого порядку);
- *ode4 (Runge-Kutta)* – метод Рунге-Кутта (четвертого порядку);
- *ode3 (Bogacki-Shampine)* – метод Богацького-Шампена (третього порядку);
- *ode2 (Heun)* – метод Хойна (другого порядку);
- *ode1 (Euler)* – метод Ейлера (першого порядку);
- *ode14x (extrapolation)* - екстраполяція .

При цьому у нижній частині вікна (рис. 8.3) виникає поле *Fixed step size* (Розмір фіксованого кроку), у яке потрібно ввести значення кроку інтегрування. У списку *Tasking mode for periodic sample times*, що виникає нижче, слід обрати один з трьох можливих режимів роботи: *Auto* (автоматический), *SingleTasking* (однозадачний) або *MultiTasking* (багатозадачний).

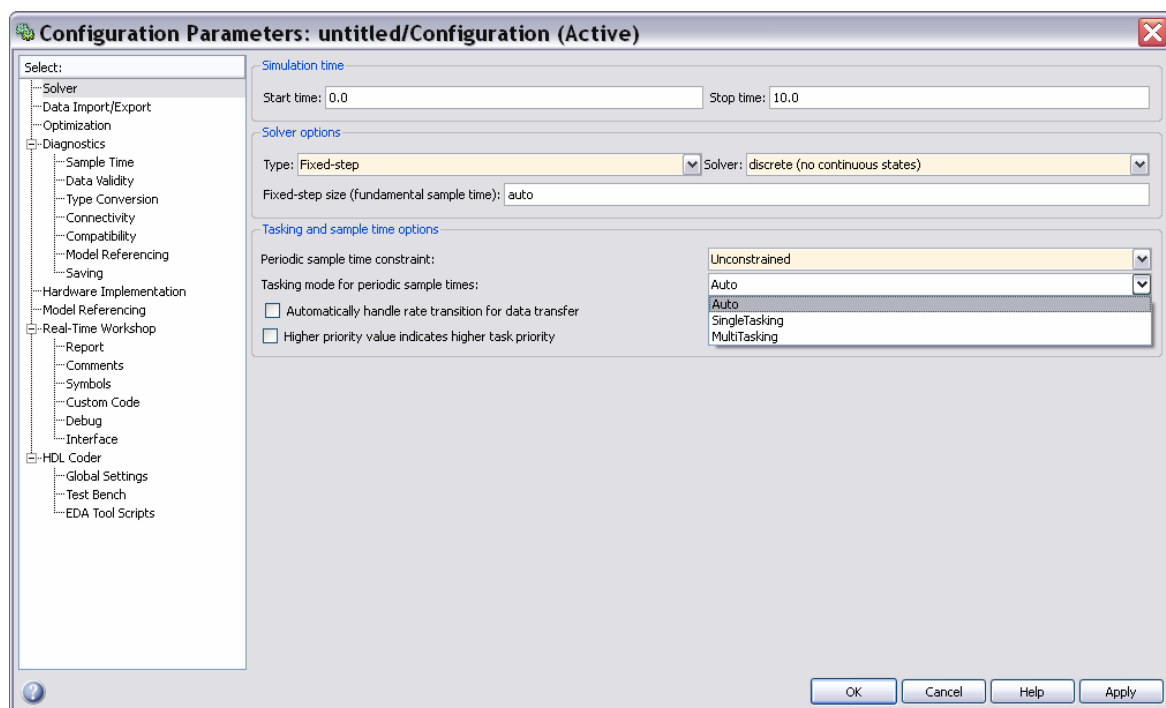


Рис. 8.3. Вид вікна *Configuration Parameters* при встановленні *Fixed-step*

При обранні у списку *Type* (Тип) елемента *Variable-step* (зі змінним кроком), у спадному списку праворуч виникне інший список інтеграторів (методів чисельного інтегрування) (див. рис. 8.2):

- *discrete (no continuous states)* – дискретний (не неперервні стани);
- *ode45 (Dormand-Prince)* – метод Дормана-Принса;
- *ode23 (Bogacki-Shampine)* – метод Богацького-Шампена;
- *ode113 (Adams)* – метод Адамса;

- *ode15s (stiff\NDF)* – метод NDF для жорстких систем;
- *ode23s (stiff\Mod. Rozenbrock)* – модифікація Розенброка для жорстких систем;
- *ode23t (Mod. stiff\Trapezoidal)* – метод трапецій, модифікація для жорстких систем;
- *ode23tb(stiff\TR-BDF2)* – метод TR-BDF2 для жорстких систем.

У цьому випадку у нижній частині поля *Solver options* виникають наступні поля введення:

- *Max step size* (Максимальний розмір кроку);
- *Min step size* (Мінімальний розмір кроку);
- *Initial step size* (Початковий розмір кроку);

а праворуч від них – поля

- *Relative tolerance* (Відносна точність);
- *Absolute tolerance* (Абсолютна точність).

В усіх полях, окрім *Relative tolerance*, встановлено значення *auto*, тобто ці параметри задаються автоматично і змінюються користувачем лише у випадку, коли йому потрібно встановити їх конкретні значення, відмінні від прийнятих за замовчуванням. Відносна точність (точніше, відносна похибка) за замовчуванням дорівнює $1 \cdot 10^{-3}$.

8.1.2. Виявлення перетинання нуля

При моделюванні систем, що містять елементи з розривними характеристиками, такі як реле, люфт, сухе тертя, необхідно максимально точно відтворити особливості поведінки системи у моменти часу, коли відбувається стрибкоподібне змінення значення розривної характеристики. Зазвичай у цей момент стрибкоподібно змінюються й властивості самої системи, а також початкові умови для продовження процесу. У зв'язку з цим вельми важливо максимально точно (з машинною точністю) визначити момент часу, у який здійснюється подібне змінення розривної характеристики, і зафіксувати поточні значення параметрів руху системи. Це потрібно для того, щоб на наступному кроці почати процес інтегрування саме з цього моменту часу з новими початковими умовами, що відповідають стану системи, набутому саме у цей момент часу.

Зазвичай стрибкоподібне змінення розривної характеристики відбувається у момент, коли одна зі змінних стану системи при своєму змінюванні у часі переходить через деякий рівень. При цьому інколи має значення напрямок, у якому здійснюється перетинання змінною цього рівня – при її збільшенні чи зменшенні – від цього може залежати, як саме зміняться (стрибкоподібно) характеристики системи і значення її змінних стану.

Задача виявлення перетинання сигналом деякого сталого рівня легко зводиться до задачі виявлення перетинання нульового рівня. Тому у подальшому процес точного визначення параметрів стану системи у момент, коли здійснюється стрибкоподібне змінення деякої характеристики системи, називатимемо *виявленням перетинання нуля*. У пакеті Simulink виявлення перетинання нуля

використовується для того, щоб зафіксувати різкі зміни у неперервних сигналах. Ця процедура відіграє важливу роль при керуванні стрибками стану і при точному інтегруванні переривчастих сигналів.

Система зазнає стрибка стану, коли змінювання значень змінних стану системи викликає значні миттєві змінення у системі. Простий приклад стрибков стану – відскакування м'яча від підлоги. При моделюванні такої ситуації використовується метод інтегрування зі змінним кроком. Метод чисельного інтегрування зазвичай не передбачає заходів, які дозволили би точно визначити момент контакту м'яча з підлогою. Внаслідок цього при моделюванні м'яч, переходячи через контактну точку, нібито проникає крізь підлогу. Використання виявлення перетинання нуля у Simulink гарантує, що момент стрибка стану системи визначений точно (з машинною точністю). Тому в результаті чисельного моделювання не відбувається проникнення м'яча крізь пол, і перехід від від'ємної швидкості м'яча до додатної у момент контакту відбувається надзвичайно різко.

У пакеті Simulink передбачені наступні блоки, які використовують виявлення перетинання нуля:

Integrator (Інтегратор) – коли представлений порт насичення *Show saturation port* (див. п. 3.6.3), виявляється момент, коли насичення відбувається; якщо вихід обмежений, то тричі виявляється перетинання нуля: коли досягається верхня межа насичення, коли досягається нижня межа насичення і коли зона насичення покинута;

Hit Crossing (Уловлювання перетинання) – виявляє момент, коли сигнал на вході перетинає заданий рівень;

Abs (Абсолютне значення) – визначає момент, коли сигнал на вході перетинає нуль у будь-якому напрямку – зменшуючись чи збільшуючись;

BackLash (Люфт) – використовується двічі: коли вхідний сигнал сягає верхнього і нижнього порогів;

Dead Zone (Мертва зона) – використовується двічі: коли сигнал входить у зону нечутливості (до цього вхідний сигнал був більше вхідного на величину нижньої границі зони) і коли залишає цю зону (вихідний сигнал стає менше вхідного на величину верхньої границі);

MinMax (Мінімум-максимум) – для кожного елемента вихідного вектора виявляє момент часу, коли вхідний сигнал стає мінімальним або максимальним;

Relay (Реле) – виявляє момент часу, коли реле потрібно включити (якщо воно виключено) або виключити (якщо воно включено);

Relational Operator (Оператори відношення) – виявляє момент часу, коли відношення змінюється;

Saturation (Насичення) – використовується двічі: коли вхідний сигнал досягає верхнього порогу або залишає його і коли сигнал досягає нижнього порогу або залишає його;

Sign (Сігнум-функція) – виявляє момент проходження вхідного сигналу через нуль;

Step (Сходінка) – виявляє момент часу, коли відбуватиметься стрибкоподібне змінення рівня вихідного сигналу.

Як показовий приклад розглянемо застосування функції перетинання нуля у програмі демонстрування поведження підстрибуючого м'яча. Для цього введіть у командному вікні Matlab команду **bounce**. В результаті її виконання виникне вікно з зображенням блок-схеми S-моделі поведження м'яча (рис. 8.4).

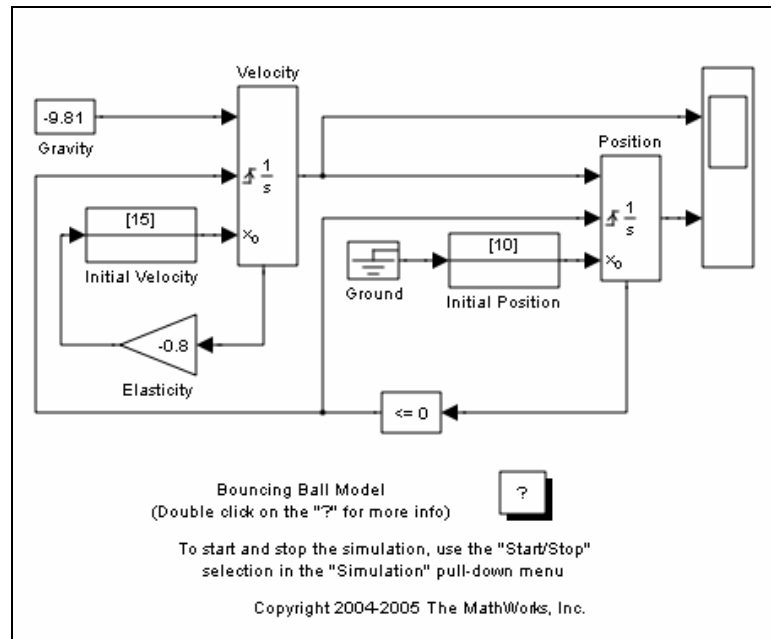


Рис. 8.4. Блок-схема S-моделі Bounce

Ця модель здійснює чисельне інтегрування методом *ode23* (з автоматичним змінюванням кроку інтегрування) диференціальне рівняння

$$\frac{d^2x}{dt^2} = -g.$$

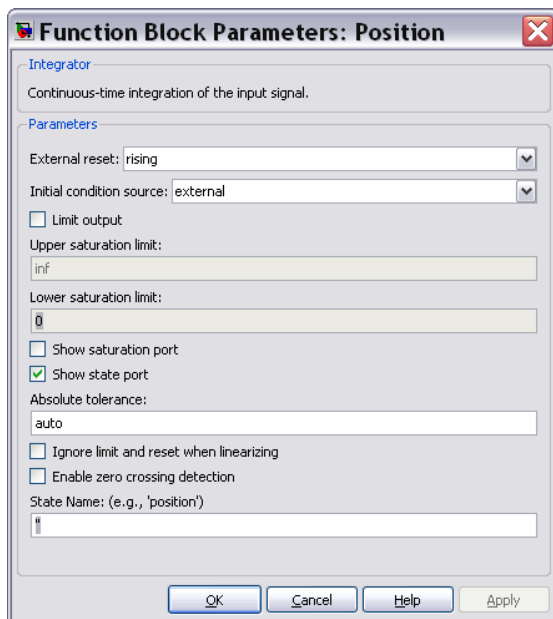
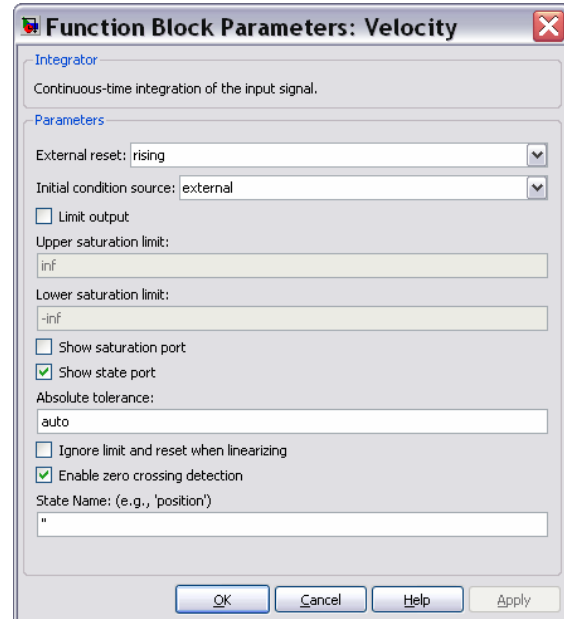
(де g - прискорення вільного падіння; x - поточна висота м'яча над підлогою) у проміжках між моментами часу, в яких відбувається зіткнення м'яча з підлогою.

Інтегрування здійснюється за допомогою двох блоків-інтеграторів - **Velocity** і **Position**. На виході першого з них одержують значення поточної швидкості руху м'яча, а на виході другого – висоти м'яча над підлогою.

Більш детальне знайомство з інтеграторами (блок **Position** і блок **Velocity**) (рис. 8.5 і 8.6) дає можливість впевнитися, що в обох випадках введено зовнішнє керування (встановлений елемент *rising* (при збільшуванні) у списку *External reset*), що приводить до появи другого зверху вхідного порту. В обох інтеграторах введено зовнішнє введення початкових умов (встановлений елемент *external* (зовнішнє) у списку *Initial condition source* (джерело початкової умови)). При цьому на лівому боці кожного блоку виникають треті зверху вхідні порти.

У блоці *Position* встановлена ніжня границя (нуль) змінювань висоти м'яча (див. рис. 8.5), а у блоці *Velocity* встановлений прапорець *Enable zero-crossing detection* (Дозволити визначати перетинання нуля) (див. рис. 8.6).

Окрім того, в обох блоках встановлені прапорці *Show state port* (Показати порт стану), у зв'язку з чим на зображенні цих блоків виникли (знизу) додатково допоміжні вихідні порти (див. рис. 8.4). На ці порти виводяться поточні значення вихідних величин інтеграторів: швидкості на порт блоку *Velocity* і висоти м'яча – на порт блоку *Position*.

Рис. 8.5. Вікно блоку *Position*Рис. 8.6. Вікно блоку *Velocity*

Як бачимо з рис. 8.4, початкові умови встановлені такими: зі швидкості 15 м/с, з положення – 10 м.

До нижніх входів блоків-інтеграторів під'єднані блоки *Initial Position* (Початкове положення) і *Initial Velocity* (Початкова швидкість), побудовані на основі стандартних блоків *IC* (*Initial Condition* – початкова умова) з поділу *Signal Attributes* бібліотеки Simulink. Призначення їх – встановлювати початкові умови для інтегратора, показані на зображенні цих блоків.

Моделювання здійснюється у такий спосіб. Інтегрування починається при початкових умовах, вказаних на зображенні блоків *Initial Position* і *Initial Velocity*. У момент, коли на другому інтеграторі (*Position*) фіксується перетинання м'ячем нуля висоти, здійснюється точне (з машинною точністю) обчислення моменту часу, в який м'яч контактує з підлогою, перераховуються значення швидкості і висоти на момент перетинання на першому і другому інтеграторах і цей момент встановлюється як новий початковий момент часу.

Найдене значення швидкості через додатковий вихідний порт інтегратора *Velocity* поступає на вхід блоку *Elasticity* (Еластичність), зміню свій знак на протилежний і зменшується на 20% (цим враховується зменшення швидкості при відскоку за рахунок втрати енергії внаслідок неідеальної пружності м'яча) і використовується як нова початкова швидкість. Початкове значення висоти пі-

сля контакту м'яча з підлогою встановлюється таким, яким є вихідний сигнал блоку *Ground*, який з'єднаний з блоком *Initial Position*. А цей сигнал завжди дорівнює нулеві. Тому початкова висота після контакту з підлогою завжди буде встановлена рівною нулю. Потім інтегрування продовжується при нових початкових умовах. Результат роботи моделі відображується в оглядовому вікні блоку *Scope* (рис. 8.7).

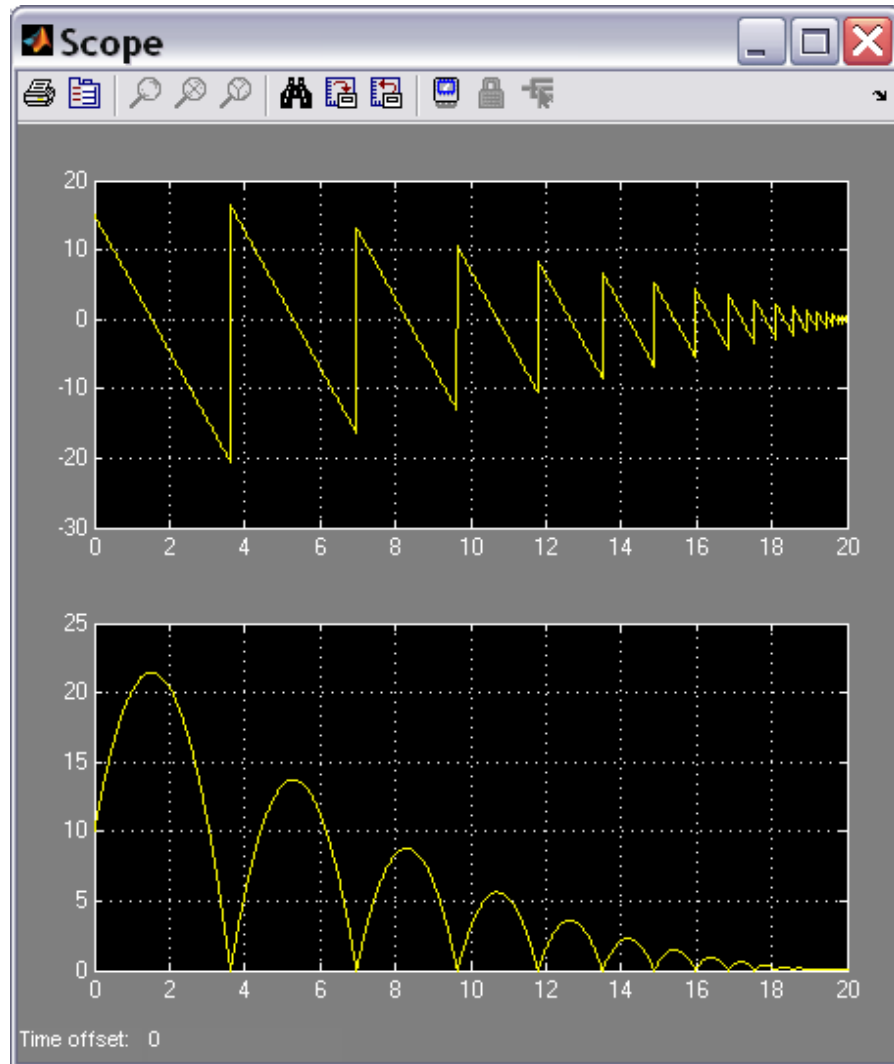


Рис. 8.7. Результат роботи S- моделі *Bounce*

Варто відмітити, що керування процесом переривання інтегрування і його продовження при нових початкових умовах здійснюється другим інтегратором *Position* при перетинанні величини на його виході встановленої нижньої границі (нуля). При цьому значення величини висоти й швидкості на виході обох інтеграторів не можуть бути використані для розрахунку нового їхнього значення. для цієї мети необхідно застосовувати додаткові вихідні порти інтеграторів, а тому потрібно встановити прапорці *Show state port* (Показати порт стану), і подати це розраховане значення не безпосередньо на вхідний порт початкової умови інтегратора, а обов'язково через блок початкової умови *IC*.

Описані вище засоби пакета Simulink дозволяють моделювати вельми важливі і важкопрограмовані особливості поведінки суттєво неінійних систем., таких як "зчеплення" і "розчеплення" рухомих частин механізмів під дією сил сухого тертя, удароподібні процеси і пов'язані з ними режими, "ковзні" режими, автоколивання, різкі переходи від одного режиму до іншого.

8.1.3. Обмін даними між середовищем Matlab і S-моделлю

Робочий простір середовища Matlab є досяжним для використовуваної S-моделі. Це означає, що якщо у якості значень параметрів у вікнах настроювання блоків S-моделі використані ймення змінних, а значення цих змінних були попередньо встановлені у робочому просторі, то ці значення одразу передадуться відповідним блокам S-моделі. Тому, щоб організувати зручне змінювання параметрів блоків S-моделі (наприклад, у діалоговому режимі), достатньо зробити наступне:

- 1) у вікнах настроювання блоків S-моделі у якості параметрів вказати ідентифікатори (ймення) замість чисел;
- 2) організувати засобами середовища Matlab (наприклад, програмно) присвоювання числових значень цим ідентифікаторам, а також (у випадку необхідності) – їхнє змінювання у діалоговому режимі;
- 3) після присвоювання числових значень усім ідентифікаторам (наприклад, через запуск відповідної M-програми) провести запуск S-моделі на моделювання.

Деякі засоби обміну даними були вже розглянуті раніше. Це блок *From Workspace* поділу *Sources* і блок *To Workspace* поділу *Sinks* стандартної бібліотеки *Simulink*. Першій прислуговується для включення сигналів, попередньо одержаних (обчислених) і записаних у робочий простір Matlab (наприклад, в результаті обчислень в середовищі Matlab), у процес моделювання S-моделі. Другий забезпечує можливість запису результатів, одержаних при моделюванні з використанням S-моделі, у робочий простір середовища Matlab.

Для записи одержаного в результаті моделювання процесу у робочий простір (див. рис. 3.19, п 3.6.1) слід вставити у блок схему S-моделі блок *To Workspace*, подати на його вхід потрібний для запису сигнал і вказати у полі *Variable name* (Імя змінної) вікна настроювання блоку ймення, під яким цей процес потрібно зберегти у робочому просторі системи Matlab. Відповідні моменти модельного часу при цьому не записуватимуться.

Для визначення процесу, який буде використаний у S-моделі, користуючись даними, записаними у робочий простір, слід вставити у блок-схему S-моделі блок *From Workspace*, з'єднати його вихід з одним з входів інших блоків, розкрити вікно настроювання блоку (рис. 4.8), і у полі введення *Data* вказати вектор, складений з двох імен – ймення масиву значень аргументів (моментів часу, у які визначений цей процес) і ймення масиву значень процесу при вказа-

них значеннях аргументу, наприклад: $[T, D]$ (рис. 4.8). У цьому випадку з масиву T робочого простору будуть зчитані усі значення, які будуть відігравати в S -моделі роль значень модельного часу, а вихідна величина блоку при моделюванні у моменти часу, що відповідають записаним у масиві T , прийматиме значення, записані у масиві D . Якщо при цьому реальні значення моментів часу при моделюванні не збігатимуться з записаними у масиві T , вдбудеться лінійна інтерполяція значень масиву D , що відповідають попередньому і наступному значенням моментів часу у масиві T .

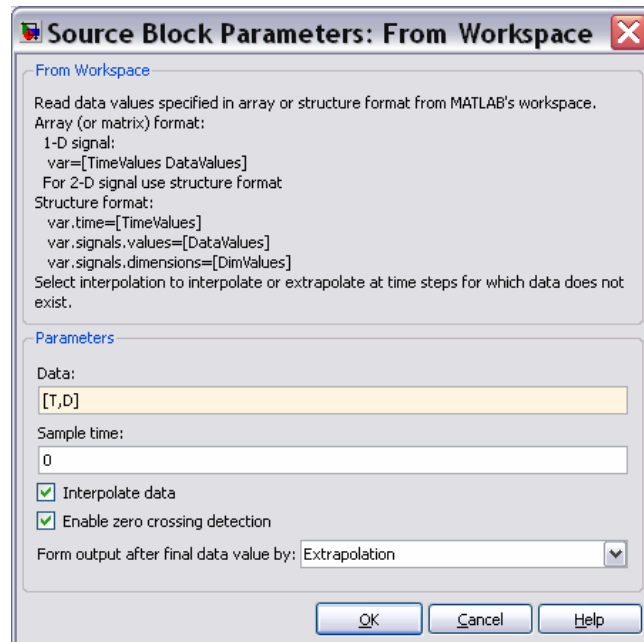


Рис. 8.8. Вікно настроювання блоку *From Workspace*

Але існує й простіший спосіб виконання вищеописаних операцій – без використання зазначених блоків.

Щоб підключити визначений у програмі Matlab процес у S -модель як вхідний, передбачений механізм включення портів входу і виходу. для цього потрібно зробити наступне.

1. Вставити блок вхідного порту *In* у блок-схему S -моделі і з'єднати його з одним з блоків S -моделі.
2. У вікні S -моделі викликати команду Simulation > Configuration Parameters, щоб відчинити вікно *Configuration Parameters*, в якому обрати команду *Data Import/ Export* (Імпорт-експорт даних) (рис. 8.9).
3. В області *Load from workspace* (Завантажити з робочого простору) встановити прапорець Input (Вхід) і у полі праворуч ввести ім'я, що складається з ймення вектор значень аргумента і ймення вектора значень вхідного сигналу при цих значеннях аргумента, наприклад: $[t, u]$.
4. Встановити значення цих векторів у Matlab, наприклад, так:

$$t = (0 : 0.1 : 1)';$$

$$u = [\sin(t), \cos(t), 4 * \exp(t)]$$

5. Запустити S-модель на моделювання.

Щоб вивести деякі сигнали, що формуються в S-моделі, у робочий простір Matlab, потрібно виконати наступні дії.

1. У блок схему моделі вставити блоки портів виходу **Out** і під'єднати до них необхідні вихідні величини інших блоків.
2. У вікні S-моделі викликати команду Simulation > Configuration Parameters, щоб відчинити вікно *Configuration Parameters*, в якому обрати команду *Data Import/ Export* (Імпорт-експорт даних) (рис. 8.9).
3. В області *Save to workspace* (зберегти у робочому просторі) відчиненого вікна встановити прапорці Time і Output, і у полі праворуч ввести ймення, під якими будуть записані значення часу і величин, що подаються на вихідні порти, у робочий простір. За замовчуванням ці імена є **tout** (для модельного часу) і **yout** (для даних з вихідних портів).

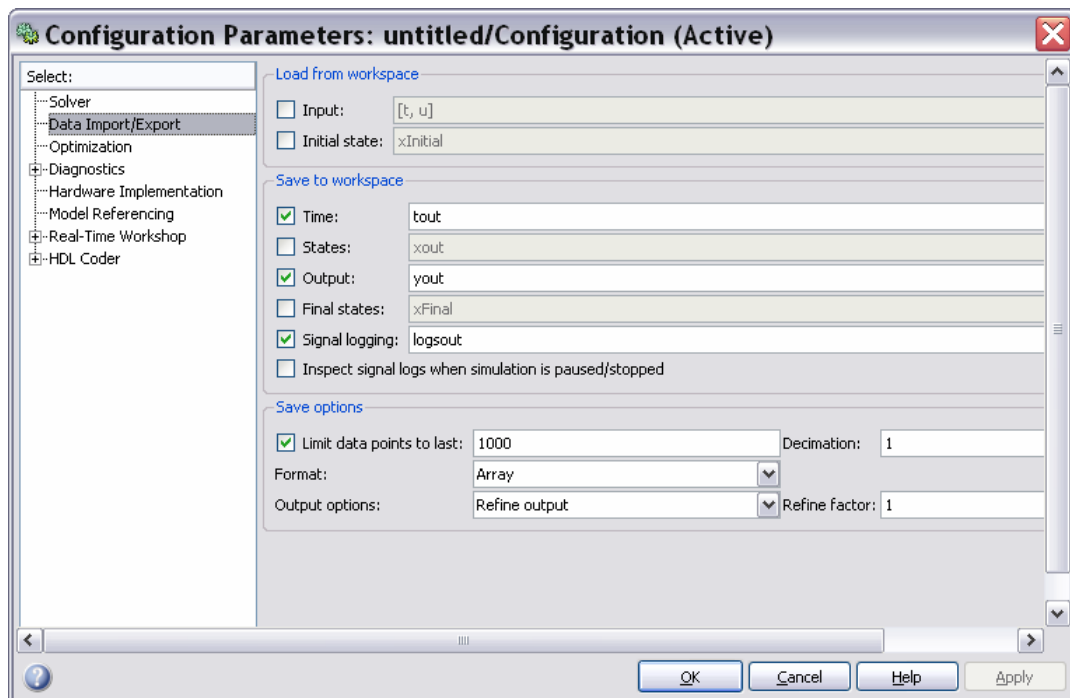


Рис. 8.9. Вкладення Data Import/ Export вікна Configuration Parameters

У цьому випадку значення модельного часу будуть записуватися у робочий простір у масив під ім'ям **tout**, а відповідні значення вихідних сигналів при цих значеннях часу – у стовпці матриці **yout** (у першій стовпець – процес, який поданий на першій вихідний порт **Out1**, у другий стовпець – процес, поданий на другий порт **Out2** і т. д.).

Після встановлення прапорця *Initial state* (Початковий стан) в області *Load from workspace* (Завантажити з робочого простору) можна ввести в S-модель початкові значення змінних стану системи. Встановивши прапорець *States* (Змінні стану) в області *Save to workspace* (Зберегти у робочому просторі),

можна записати поточні значення змінних стану системи у робочий простір під ім'ям **xout** (або під іншим ім'ям, якщо його записати у поле праворуч від прапорця *State*). Нарешті, можна записати й кінцеві значення змінних стану у вектор **xFinal**, якщо встановити прапорець *Final state* (Кінцевий стан).

8.1.4. Запуск процесу моделювання S-моделі з середовища Matlab

Розглянемо засоби, які дозволяють запускати процес моделювання утворених S-моделей із програми Matlab.

S-модель запускається на виконання, якщо у програмі Matlab (або у командному вікні Matlab) викликати процедуру **sim**:

[t,x,y1,y2,...,yn] = sim(model, timespan,options,ut).

Тут **model** – симольний рядок, що містить ім'я MDL-файлу, в якому записана відповідна S-модель; **timespan** – вектор, що складається з двох елементів – значень початкового і кінцевого моментів часу моделювання; **options** – вектор значень параметрів інтегрування, який встановлюється процедурою **simset**

options = simset('Властивість1',Значення1,'Властивість2',Значення2...);

Процедура **sim** повертає наступні значення: **t** – вектор вихідних значень моментів модельного часу; **x** – масив (вектор) змінних стану системи; **y1** – першій стовпець матриці вихідних змінних системи (які надаються до вихідних портів) і т. д.

Змінювати параметри розв'язувача і процесу інтегрування у Matlab можна за допомогою функції **simset**, як це показано вище. У такий спосіб можна задати значення наступних властивостей розв'язувача:

- 'Solver' – назва розв'язувача; значення (вказується між двома апострофами) може бути однією з наступних: ode45, ode23, ode1b, ode15s, ode23s – для інтегрування з автоматично змінюваним кроком; ode5, ode4, ode3, ode2, ode1 – для інтегрування з фіксованим кроком;
- 'RelTol' – відносна припустима похибка; значення може бути додатним скаляром; за замовчуванням встановлюється 1e-3;
- 'AbsTol' – абсолютна припустима похибка; значення може бути додатним скаляром; за замовчуванням встановлюється 1e-6;
- 'FixedStep' – фіксований крок (додатний скаляр);
- 'MaxOrder' – максимальний порядок методу (застосовується лише для методу ode15); може бути одним з цілих чисел 1, 2, 3, 4; за замовчуванням дорівнює 5;
- 'MaxRows' – максимальна кількість рядків у вихідному векторі; невід'ємне ціле;
- 'InitialState' – вектор початкових значень змінних стану;
- 'FinalStateName' – ім'я вектора, в який буде записуватися кінцеве значення вектора змінних стану моделі;

- 'OutputVariables' – вихідні змінні системи; за замовчуванням має значення {txy}; можливі варіанти tx, ty, ху, t, x, y; усі вони неявно вказують, які саме вихідні змінні не будуть виводитися.

8.1.5. Створення S-блоків з використанням програм Matlab. S-функції

У системі Matlab передбачений механізм перетворення деяких процедур, написаних мовами високого рівня, у блок S-моделі. Він реалізується за допомогою S-функцій.

S-функція – це відносно самостійна програма, яка написана користувачем мовою Matlab або C і має візуальне подання у виді блоку Simulink. Застосування S-функцій дозволяє вирішити наступні задачі:

- утворення нових (користувацьких) блоків, які доповнюють бібліотеку пакета Simulink;
- використання опису модельованої системи у виді системи математичних рівнянь;
- включення раніше створених програм, написаних M-мовою або мовою C, у S-модель.

Програмний код S-функції має чітку структуру. У випадку, коли S-функція утворюється на основі M-мови, ця структура наведена у файлі *SfinTMPL.m*, який міститься у папці TOOLBOX\SIMULINK\BLOCKS. заголовок S-функції у загальному випадку має вид:

```
function [sys,x0,str,ts] = <Ім'я S-функції>(t, x, u, flag{,<Параметри>})
```

Стандартними аргументами S-функції є:

t – поточне значення аргументу (модельного часу);

x – поточне значення вектора змінних стану;

u – поточне значення вектора вхідних величин;

flag – цілочислова змінна, яка відбиває етапи дії S-функції;

<Параметри> - перелік додаткових ідентифікаторів, які характеризують модельовану систему і значення яких використовуються у S-функції (їхня наявність не є обов'язковою).

В результаті обчислень, що виконуються при роботі S-функції, присвоюються значення таким змінним:

sys – системна змінна, вміст якої залежить від значення, що набуває змінна flag;

x0 – вектор початкових значень змінних стану;

str – символна змінна стану (зазвичай вона є пустою []);

ts – матриця, що містить інформацію про дискрету часу.

Текст S-функції складається з стандартного тексту самої S-функції і текстів наступних внутрішніх процедур, якими вона використовує:

- *mdlInitializeSizes* – встановлює розміри змінних S-функції і початкові значення змінних стану;

- ***mdlDerivatives*** – процедура обчислення поточних значень правих частин диференційних рівнянь системи, записаних у формі Коші у випадку, коли змінні стану об’явлені як неперервні;
- ***mdlUpdate*** – процедура оновлення на наступному інтервалі дискрету часу значень змінних стану, об’явлених як дискретні;
- ***mdlOutputs*** – процедура обчислення значення вектора вихідної змінної блоку S-функції;
- ***mdlGetTimeOfNextVarHit*** – допоміжна функція для визначення моменту часу, коли конкретна змінна перетинає заданий рівень;
- ***mdlTerminate*** – функція переривання роботи S-функції.

Деякі з вказаних процедур можуть не використовуватися. Це залежить від типу рівнянь (алгебричні, диференційні або різницеві), якими описується модельована S-функцією система. Так, якщо поведження системи описується лише алгебричними рівняннями, то не використовуються майже усі вказані внутрішні процедури, за виключенням процедур ***mdlInitializeSizes*** і ***mdlOutputs***. В останній й обчислюються відповідні алгебричні співвідношення, що визначають зв’язок між вхідними змінними **u** і вихідними змінними **y**. У тому випадку, коли поведження системи визначено системою диференційних рівнянь, не використовується функція ***mdlUpdate***, якщо рівняння системи є різницевими - процедура ***mdlDerivatives***.

Головна процедура S-функції містить, у головному, звернення до тієї чи іншої внутрішньої процедури у відповідності до значення змінної *flag*. Наприклад:

```
switch flag,
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
  case 1,
    sys=mdlDerivatives(t,x,u);
  case 2,
    sys=mdlUpdate(t,x,u);
  case 3,
    sys=mdlOutputs(t,x,u);
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
    sys=mdlTerminate(t,x,u);
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
```

У залежності від значення змінної *flag*, виконуються наступні дії (через звернення до відповідної внутрішньої процедури):

- 0 – ініціалізація блоку;
- 1 – звернення до процедури правих частин диференційних рівнянь у формі Коші;
- 2 – обчислення нових значень змінних стану на наступному кроці дискретизації (для різницевих рівнянь);

- 3 – формування значення вектора вихідних величин;
- 4 – формування нового значення модельного часу, яке відліковується від моменту перетинання заданого рівня певною змінною стану;
- 9 – припинення роботи блоку.

Встановлювання і змінювання значень змінної *flag* відбувається автоматично, без втручання користувача, у відповідності до логіки функціонування блоків Simulink при моделюванні.

Отже, використання S-функції дозволяє моделювати роботу як звичайних алгебричних, так і динамічних (неперервних або дискретних) ланок.

Щоб утворити S-блок на основі використання S-функції, виконайте наступні дії.

1. Напишіть текст S-функції, наприклад у виді М-файлу, користуючись файлом-шаблоном *SfunTMPL.m*.
2. Перетягніть стандартний блок S-функції (рис. 8.10) з поділу **User-Defined Functions** бібліотеки **SIMULINK** у вікно блок-схеми, в який буде створюватися новий S-блок.

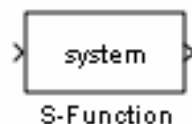


Рис. 8.10. Заготівка S- блоку S-функції

3. Двічі клацніть на зображенні блоку **S-function**. Це призведе до виникнення на екрані вікна його настроювання (рис. 8.11). Вікно містить поля введення *S-function name* (Ім'я S-функції), в яке вводиться ім'я файлу з написаним текстом S-функції, і *S-function parameters* (Параметри S-функції), в яке вводяться ймення або значення параметрів блоку, вказаних у розділі <Параметри> М-файла, що містить написаний текст S-функції.

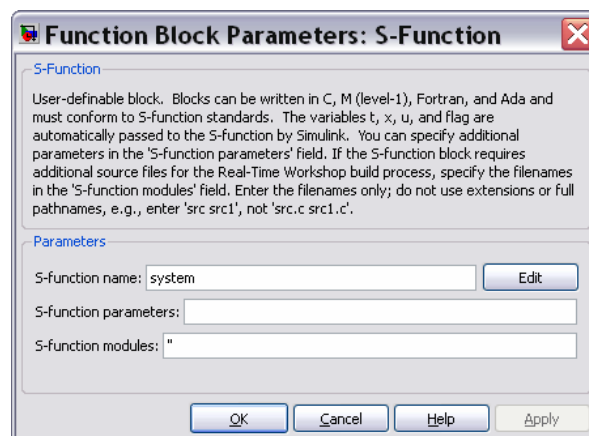


Рис. 8.11. Вікно настроювання блоку S-function

4. Введіть у вказані поля ім'я М-файла, у якому записаний текст S-функції, і список значень параметрів. Якщо, наприклад, у перше поле ввести ім'я S_KA, а у друге – рядок J, Ug0, UgSk0, вікно набуде виду, поданому на рис. 8.16.
5. Клацніть мішкою на кнопці ОК. Якщо система виявить М-файл з вказаним ім'ям у папках які досяжні Matlab, вікно, що подане на рис. 8.12, зникне, а на зображенні блоку у вікні блок-схеми виникне введене ім'я S-функції (точніше, написаного М-файла) (рис. 8.13).

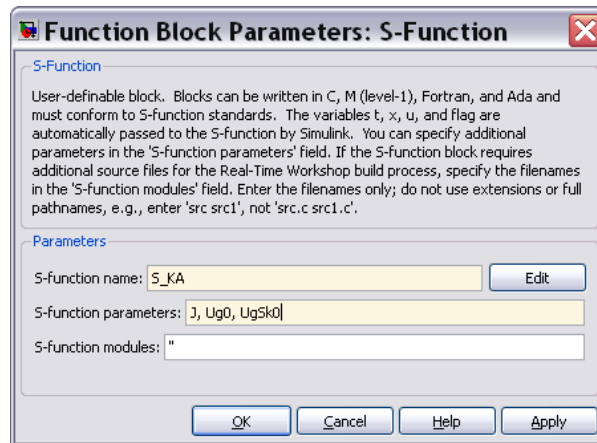


Рис. 8.12. Вікно S-function після введення даних

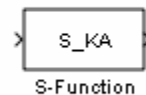


Рис. 8.13. Блок S-function після введення даних

S-блок на основі S-функції, що міститься у М-файлі S_KA.m буде створений. Тепер його можна використовувати як звичайний S-блок у блок-схемі S-моделі. До входу цього блоку має надходити векторний сигнал \mathbf{u} . Виходом блоку стане векторний сигнал \mathbf{y} , який сформований S-функцією у внутрішній процедурі *mdlOutputs*.

8.1.6. Приклад утворення і роботи з S-функцією

Створимо S-функцію, яка релізує динамічні властивості твердого тіла при його обертальному русі. Для опису динаміки тіла скористаємося динамічними рівняннями Ейлера у матричній формі:

$$\mathbf{J} \frac{d\boldsymbol{\omega}}{dt} + (\boldsymbol{\omega} \times) \mathbf{J} \boldsymbol{\omega} = \mathbf{M}. \quad (8.1)$$

Тут \mathbf{J} - матриця моментів тіла відносно декартових осей, зв'язаних з тілом; $\boldsymbol{\omega}$ - матриця-стовпець з проекцій абсолютної кутової швидкості тіла на ті самі осі; $(\boldsymbol{\omega} \times)$ - косиметрична матриця виду

$$(\boldsymbol{\omega} \times) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \quad (8.2)$$

складена з тих самих проєкцій; \mathbf{M} - матриця-стовпець з проєкцій моменту зовнішніх сил на пов'язані осі.

Такими рівняннями описується, наприклад, обертальний рух космічного апарату. Тому у подальшому тіло іноді будемо ототожнювати з космічним апаратом.

Утворимо М-файл відповідної S-функції. Назвемо його `S_DUE.m`.

```
function [sys,x0,str,ts] = S_DUE(t,x,M,flag,J,UgSk0)
% S-функция S_DUE Динамических Уравнений Ейлера
% Реализует динамику вращательного движения твердого тела,
% отыскивая вектор абсолютной угловой скорости тела
% по заданному вектору моментов внешних сил,
% действующих на тело
% ВХОД блока:
%   M - вектор проекций момента внешних сил на оси
%   X, Y i Z связанной с телом системы координат
% ВЫХОД блока:
%   y - вектор из шести элементов: первые три - проекции
%       абсолютной угловой скорости от тела на указанные оси,
%       последние три - проекции на те же оси
%       углового ускорения тела
% Входные ПАРАМЕТРЫ S-функции:
%   J - матрица моментов инерции тела в указанных осях;
%   UgSk0 - вектор начальных значений проекций
%           угловой скорости тела

% Лазарев Ю.Ф., Украина, 18-12-2001

IJ=inv(J); % вычисление обратной матрицы моментов инерции
switch flag,
case 0
    [sys,x0,str,ts] = mdlInitializeSizes(UgSk0);
case 1,
    sys = mdlDerivatives(t,x,M,J,IJ);
case 3,
    sys = mdlOutputs(x,M,J,IJ);
case 9
    sys = [];
end
% Конец процедуры
%=====
function [sys,x0,str,ts] = mdlInitializeSizes(UgSk0,Ug0)
sizes = simsizes;
sizes.NumContStates = 3;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 6;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = UgSk0;
str = [];
ts = [0 0];
```

```

% Конец процедуры mdlInitializeSizes
%=====
function z = mdlDerivatives(t,x,M,J,IJ)
% ВХОДНОЙ вектор M является вектором проекций моментов внешних сил,
% действующих на космический аппарат соответственно по осям X Y Z
% x(1)=om(1); x(2)=om(2); x(3)=om(3);
% z(1)=d(om(1))/dt; z(2)=d(om(2))/dt; z(3)=d(om(3))/dt;
% |Jx Jy Jz|
% J = |Jxy Jy Jyz| - матрица моментов инерции КА
% |Jxz Jz Jz|
om=x(1:3); omx=vect2ksm(om);
z=IJ*(M-cross(om,J*om)); % ДИНАМИЧЕСКИЕ уравнения Ейлера
% Конец процедуры mdlDerivatives
%=====
function y = mdlOutputs(x,M,J,IJ)
y(1:3)=x;
om=x(1:3);
zom=IJ*(M-cross(om,J*om)) ; % Определения УСКОРЕНИЙ
y(4:6)=zom;
% Конец процедуры mdlOutputs

```

В якості вхідного вектора утворюваного S-блоку прийнятий вектор M , який складається з трьох значень поточних проекцій вектора моменту зовнішніх сил, що діють на тіло, на осі системи декартових координат, пов'язаної з тілом. Утворимо вихідний вектор y із шести елементів: перші три – поточні значення проекцій абсолютної кутової швидкості тіла, другі три – проєкції на ті самі осі вектора абсолютного кутового прискорення тіла:

$$y = [\text{omx}, \text{omy}, \text{omz}, \text{epsx}, \text{epsy}, \text{epsz}]$$

Утворюваний S-блок розглядається як неперервна система (з трьома неперервними змінними стану $x = [\text{omx}, \text{omy}, \text{omz}]$). Тому з тексту головної програми S-функції вилучена процедура *mdlUpdate* і залишена процедура *mdlDerivatives*, яка є фактично підпрограмою правих частин динамічних рівнянь Ейлера.

Утворимо нове (пусте) вікно блок-схеми. Перетягнемо в нього стандартний блок S-функції з поділу *User-Defined Functions* бібліотеки *SIMULINK*.

Подвійним клацанням мишкою на зображенні блоку викличем його вікно настроювання і запишемо в нього назву M-файла створеної S-функції і його параметри J , $UgSk0$ (рис. 8.14). Клацнемо на кнопці ОК. Внаслідок цього вікно настроювання зникне, а на зображенні блоку виникне ім'я **S_DUE**.

Тепер у цьому ж вікні з S-блоком створимо блок-схему для перевірки правильності роботи цього блоку. Але для цього попередньо потрібно продумати умови тестового прикладу.

Розглянемо такий випадок:

- на тіло не діють моменти зовнішніх сил, тобто тіло вільно обертається у просторі;
- осі декартової системи координат, яка жорстко пов'язана з тілом, спрямовані вздовж головних осей інерції тіла; за цих умов матриця моментів інерції буде діагональною;
- тіло є динамічно симетричним, а його вісь фігури спрямована вздовж другої осі тіло є динамічно симетричним, а його вісь фігури спрямована вздовж другої осі (Y) зв'язаної системи коор-

динат; це означає, що матриця моментів інерції матиме такий вид:

$$\mathbf{J} = \begin{bmatrix} J_e & 0 & 0 \\ 0 & J & 0 \\ 0 & 0 & J_e \end{bmatrix},$$

де J_e - екваторіальний момент інерції; J - момент інерції тіла відносно його осі фігури (осьовий момент інерції тіла);

- тіло попередньо приведено у обертання з кутовою швидкістю Ω навколо своєї осі фігури і має незначну (у порівнянні з Ω) початкову кутову швидкість ω_0 навколо осі X.

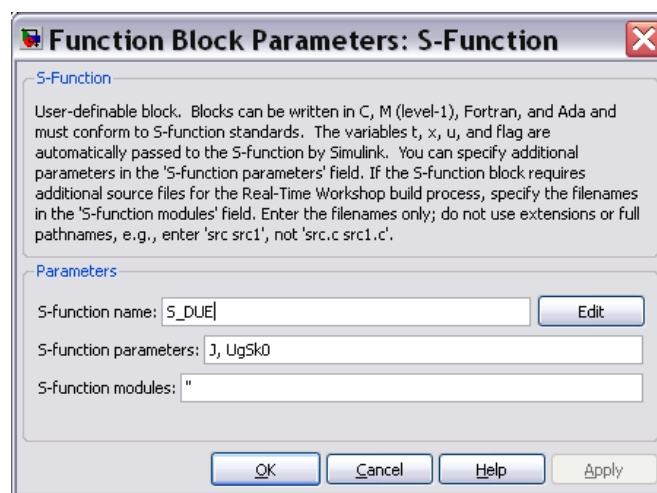


Рис. 8.14. Вікно настроювання блоку S_DUE

За цих умов рівняння (8.1) набудуть такого вигляду:

$$\begin{cases} J_e \frac{d\omega_X}{dt} = (J - J_e)\omega_Y\omega_Z \\ J \frac{d\omega_Y}{dt} = 0 \\ J_e \frac{d\omega_Z}{dt} = -(J - J_e)\omega_Y\omega_X \end{cases}$$

і при заданих початкових умовах матимуть такий розв'язок:

$$\omega_X = \omega_0 \cos(k\Omega t); \quad \omega_Y = \Omega; \quad \omega_Z = \omega_0 \sin(k\Omega t), \quad (8.3)$$

де позначено

$$k = \frac{J - J_e}{J_e}. \quad (8.4)$$

Отже, якщо утворена модель є правильною (адекватною процесові, що описується рівнянням (8.1)) і для неї забезпечені задані умови, при моделюванні маємо отримати результати, що відповідають формулам (8.3).

Додамо у блок-схему блок констант, який формує нульовий вектор моментів зовнішніх сил, а також блоки Scope, які дозволять проконтролювати ре-

зультати моделювання у виді графіків залежностей проєкцій кутової швидкості і кутового прискорення тіла від часу. В результаті одержимо блок схему, подану на рис. 8.15.

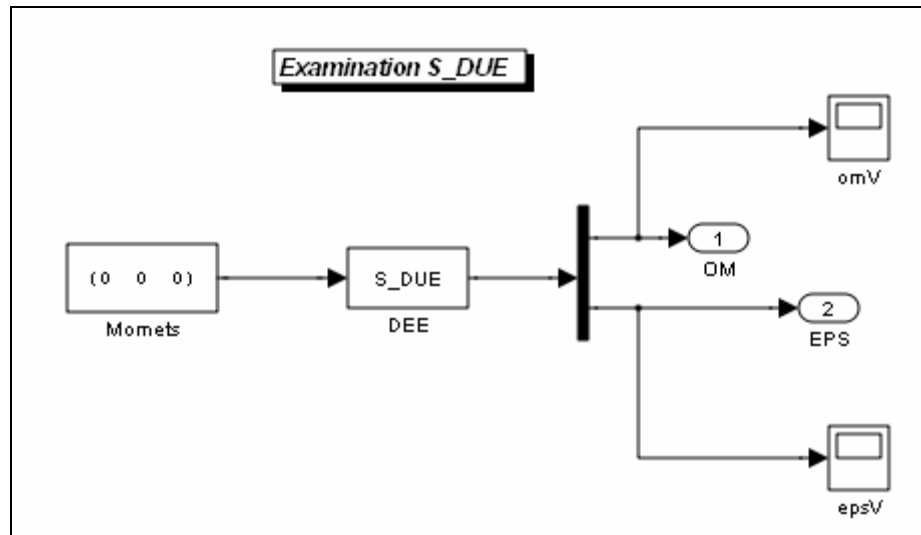


Рис. 8.15. Блок-схема S-моделі Prov

Перед запуском цієї S-моделі, якій дамо ім'я Prov, встановимо кінцевий час моделювання у 5000 с через команду Simulation > Configuration Parameters > Solver > Stop time, а також введемо у командному вікні Matlab оператори

```
>> J=diag([400 600 400])
J =
  400    0    0
    0   600    0
    0    0   400
>> UgSk0=[0.001 0.01 0]
UgSk0 =
  0.0010  0.0100  0
```

які задають матрицю моментів інерції і початкові умови.

Тепер слід перейти у вікно блок-схеми і запусити блок-схему на моделювання.

Звернувшись до оглядових блоків по завершенні процесу моделювання, можна впевнитися, що утворена модель дає результати, що повністю збігаються з одержаними з формул (8.3).

Показати у графічній формі результати роботи створеної моделі досить важко з огляду на наступні обставини. Блоки Scope виводять графіки на чорному полі. Тому при копіюванні відповідного графічного вікна на папір виходить неякісне зображення. Можна скопіювати його на папір за допомогою команди друку графічного вікна блоку Scope, але тоді відповідне зображення займе цілий аркуш – його неможливо зменшити засобами текстового редактору. Лінії на графіку після друку на чорно-білому принтері не відрізнятимуться одна від одної. На них неможливо нанести написи, щоб вказати особливості кривих, до того ж складно змінити стиль лінії.

Виходячи з зазначеного можна висновувати, що найраціональнішим рішенням буде передати результати у робочий простір шляхом введення у блок схему виходних портів (див. рис. 8.15) і відправки на них тих сигналів, які потрібно подати графічно. Потім слід побудувати необхідні графіки, використовуючи графічні засоби Matlab. Останнє можна зробити бкзпосередньо, за допомогою команд Matlab, у командному вікні, але доцільніше виконати цю процедуру програмно, причому бажано об'єднати в ній усі дії:

- 1) введення значень параметрів, початкових умов тощо;
- 2) встановлення параметрів інтегрування;
- 3) звернення до S-моделі і запуск її на моделювання;
- 4) обробку одержаних результатів, побудову і оформлення графіків.

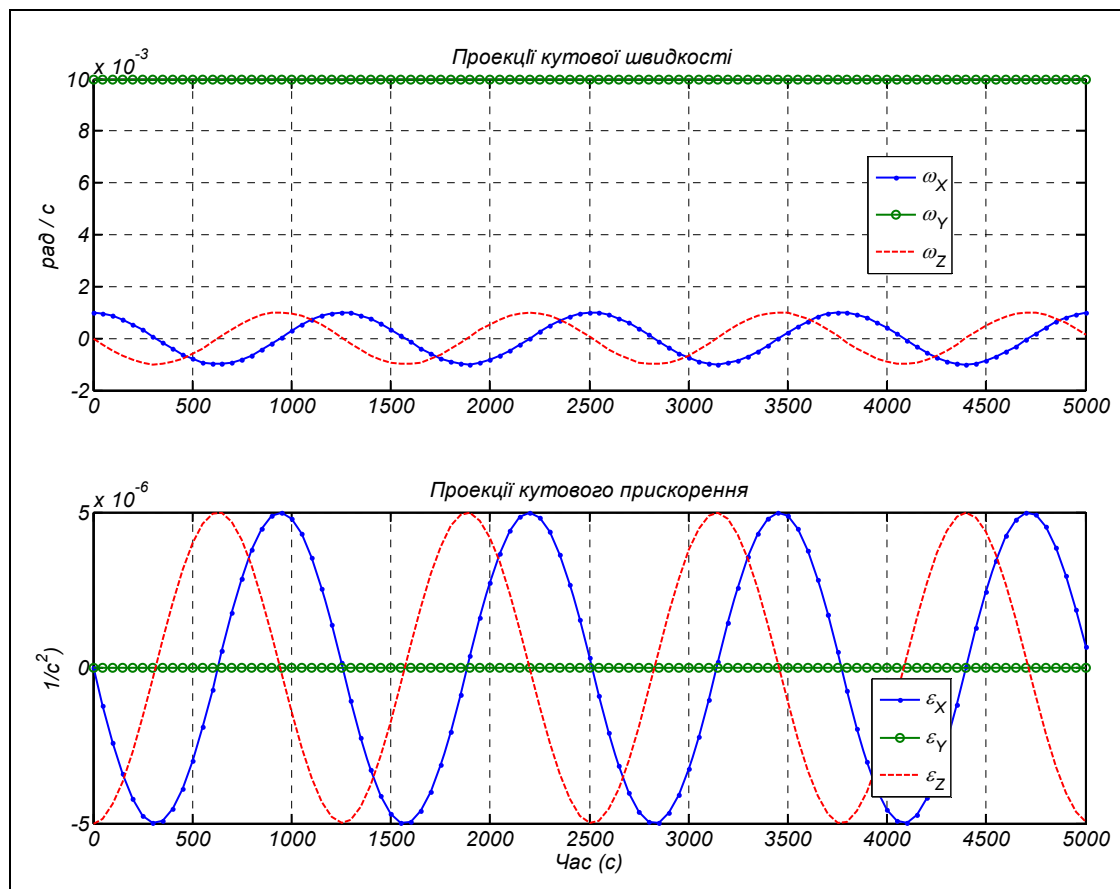


Рис. 8.16. Результати роботи програми *Prov_upr*

Приклад такої програми за ім'ям **Prov_upr** наведений нижче.

```
% Prov_upr
% Керуюча програма для запуску моделі Prov.mdl

% Лазарєв Ю.Ф. 23-07-2009
J=[400 0 0; 0 600 0; 0 0 400]; % Введення значень матриці інерції
UgSk0=[0.001 0.01 0]; % Введення початкових значень проекцій кутової швидкості
% Встановлення параметрів моделювання
options=simset('Solver','ode4','FixedStep',5e1);
% МОДЕЛЮВАННЯ на S-моделі
sim('Prov',5000,options);
```

```

% Формування даних і виведення графіків
tt=tout; omx=yout(:,1); omy=yout(:,2); omz=yout(:,3);
epsx=yout(:,4); epsy=yout(:,5); epsz=yout(:,6);

subplot(2,1,1), h=plot(tt,omx,'-',tt,omy,'o-',tt,omz,'--');grid
set(h,'LineWidth',2); set(gca,'LineWidth',2)
set(gca,'FontAngle','italic','FontSize',16)
title('Проекції кутової швидкості'), ylabel('рад / с'),
legend('\omega_X','\omega_Y','\omega_Z',0)

subplot(2,1,2), h=plot(tt,epsx,'-',tt,epsy,'o-',tt,epsz,'--');grid
set(h,'LineWidth',2); set(gca,'LineWidth',2)
set(gca,'FontAngle','italic','FontSize',16)
title('Проекції кутового прискорення'), ylabel('1/с^2'), xlabel('Час (с)')
set(gcf,'color','white'), legend('\epsilon_X','\epsilon_Y','\epsilon_Z',0)

```

Запуск цієї програми дозволяє одержати графіки, показані на рис. 8.16. Тепер читач може наочно впевнитися у адекватності моделі.

Зауважимо, що смодельований рух відповідає вільному рухові симетричного гіроскопа – його нутаційним коливанням. У гіроскоп тіло перетворює надання йому порівняно швидкого обертання навколо однієї з його осей ($\omega_Y = \Omega = 0,01 \text{ с}^{-1}$). Матриця моментів інерції прийнята діагональною, тобто припускається, що тіло динамічно збалансовано відносно осей X , Y і Z . Нарешті, моменти інерції відносно осей X і Z прийняті рівними. Це означає, що тіло є динамічно симетричним з віссю фігури Y .

8.1.7. Запуск M-програм із S-моделі

Слід вказати ще один, більш зручний спосіб поєднання S-моделі з програмами Matlab. Він полягає у виклику M-файлів безпосередньо з S-моделі за допомогою спеціально передбачених для цього засобів.

Припустимо, що перед початком завантажування S-моделі MODEL.mdl потрібно викликати M-файл, наприклад за ім'ям PERVdan, який містить операції присвоювання первісних значень усім даним. Це можна здійснити, якщо при створенні S-моделі з вказаним ім'ям у командному вікні Matlab ввести наступну команду:

```
set_param('MODEL', 'PreLoadFcn', 'PERVdan')
```

Вона пов'яже файл PERVdan.m з S-моделлю MODEL.mdl у такий спосіб, що він буде автоматично викликатися при виклику цієї S-моделі. Якщо після виконання цієї команди записати на диск дану S-модель, то при подальших її викликах спочатку автоматично буде викликаний файл PERVdan.m і лише потім на екрані виникне блок-схема S-моделі, готова для моделювання.

Перевірити, який саме M-файл використовується в даній S-моделі у якості попередньо виконаного, можна шляхом виклику команди

```
get_param('ім'я S-моделі', 'PreLoadFcn')
```

За допомогою функції *set_param* можна встановити в S-моделі значення багатьох її параметрів, у тому числі і параметрів окремих блоків моделі. У загальному виді звернення до функції може мати такий вид:

`set_param('ім'я S-моделі / Ім'я блоку', 'Параметр1', Значення1, 'Параметр2', Значення2 ...)`

Якщо вказаний параметр Ім'я блоку, то наступні значення присвоюються параметрам цього блоку. Наприклад, звернення

`set_param('MODEL', 'Solver', 'ode15s', 'StopTime', 3000)`

приведе до встановлення у S-моделі розв'язувача *ode15s* і часу завершення процесу моделювання – 3000.

При використанні звернення до цієї функції виду

`set_param('MODEL / Rivnyannya', 'Gain', '1000')`

у блоці *Rivnyannya* S-моделі **MODEL** параметру **Gain** буде присвоєно значення 1000.

Команда

`set_param('MODEL / Fcn', 'Position', [50 100 110 120])`

встановить зображення блоку *Fcn* у S-моделі **MODEL** у прямокутник з координатами **[50 100 110 120]** у вікні блок-схеми. При зверненні

`set_param('MODEL / Compute', 'OpenFcn', 'my_open_fcn')`

блок **Compute** S-моделі **MODEL** буде пов'язаний з M-програмою Matlab, яка записана у файлі **my_open_fcn.m**. Після цього файл **my_open_fcn.m** буде викликатися кожного разу після подвійного клацання на зображенні блоку **Compute**.

Якщо потрібно викликати деякий M-файл перед проведенням власне моделювання на S-моделі або після нього (наприклад, потрібна викликати програму, яка дозволяє змінити значення параметрів моделі у діалоговому режимі, або використати програму виведення результатів моделювання у графічній формі), можна встановити на вільному місці блок-схеми порожні блоки *Subsystem* (з поділу *Ports & Subsystems*). Кожен з них здійснюватиме виклик відповідного M-файла.

Порожні блоки *Subsystem* блок-схеми можа зв'язати з певними M-програмами, набравши у командному вікні команду, аналогічну приведеній раніше. Спробуємо організувати таку форму керування процесом моделювання моделі *Prov*.

Для цього створимо на основі раніше створеного файла *Prov_upr* три окремих файли:

- *Prov_DUE_Pred.m*, який виконує присвоювання значень первісним величинам;
- *Menu_DUE.m*, який здійснює змінювання первісних даних у діалоговому режимі;
- *Graf_DUE.m*, який забезпечує виведення результатів моделювання у графічне вікно.

Тексти цих програм наведені нижче.

Програма *Prov_DUE_Pred*

```
% Prov_DUE_Pred
```

```
% Програма встановлення первісних (вшитих) значень параметрів моделі Prov.mdl
```



```
% Лазарєв Ю.Ф. 23-07-2009
clc, clear all
J=[400 0 0; 0 600 0; 0 0 400]; % Введення значень матриці інерції
UgSk0=[0.001 0.01 0]; % Введення початкових значень проєкцій кутової швидкості
% Встановлення параметрів чисельного інтегрування
hi=10; TK=10000;
```

Програма Menu_DUE

```
% MENU_DUE
% Програма змінювання первісних дазначень параметров модели Prov.mdl

% Лазарєв Ю.Ф. 23-07-2009
k=1;
while k<12
    k=menu('Дані для моделі Prov. Що змінити?',...
        sprintf('Jx = %g',J(1,1)),    sprintf('Jy = %g',J(2,2)),    sprintf('Jz = %g',J(3,3)),...
        sprintf('Jxy = %g',J(1,2)),    sprintf('Jxz = %g',J(1,3)),    sprintf('Jyz = %g',J(2,3)),...
        sprintf('OMx(0) = %g',UgSk0(1)), sprintf('OMy(0) = %g',UgSk0(2)),...
        sprintf('OMz(0) = %g',UgSk0(3)), sprintf('hi = %g',hi),sprintf('TK = %g',TK),...
        'Нічого не змінювати');
    if k==1
        J(1,1)=input([sprintf('Поточне значення Jx=%g; ',...
            J(1,1)),'Встановіть нове значення Jx=']);
    end
    if k==2
        J(2,2)=input([sprintf('Поточне значення Jy=%g; ',...
            J(2,2)),'Встановіть нове значення Jy=']);
    end
    if k==3
        J(3,3)=input([sprintf('Поточне значення Jz=%g; ',...
            J(3,3)),'Встановіть нове значення Jz=']);
    end
    if k==4
        J(1,2)=input([sprintf('Поточне значення Jxy=%g; ',...
            J(1,2)),'Встановіть нове значення Jxy=']);
    end
    if k==5
        J(1,3)=input([sprintf('Поточне значення Jxz=%g; ',...
            J(1,3)),'Встановіть нове значення Jxz=']);
    end
    if k==6
        J(2,3)=input([sprintf('Поточне значення Jyz=%g; ',...
            J(2,3)),'Встановіть нове значення Jyz=']);
    end
    if k==7
        UgSk0(1)=input([sprintf('Поточне значення OMx(0)=%g; ',...
            UgSk0(1)),'Встановіть нове значення OMx(0)=']);
    end
    if k==8
        UgSk0(2)=input([sprintf('Поточне значення OMy(0)=%g; ',...
            UgSk0(2)),'Встановіть нове значення OMy(0)=']);
    end
    if k==9
        UgSk0(3)=input([sprintf('Поточне значення OMz(0)=%g; ',...
            UgSk0(3)),'Встановіть нове значення OMz(0)=']);
    end
    if k==10
        hi=input([sprintf('Поточне значення hi=%g; ',hi), ...
            'Встановіть нове значення hi=']);
    end
    if k==11
```

```

    TK=input(sprintf('Поточне значення ТК=%g; ',TK),...
            'Встановіть нове значення ТК=');
end
end
J(2,1)=J(1,2); J(3,1)=J(1,3); J(3,2)=J(2,3);

```

Програма Graf_DUE

```

% Graf_DUE
% Програма побудови в графічному вікні графіків результатів роботи моделі
Prov.mdl

% Лазарєв Ю.Ф. 23-07-2009
% Формування даних
tt=tout;
omx=yout(:,1); omy=yout(:,2); omz=yout(:,3);
epsx=yout(:,4); epsy=yout(:,5); epsz=yout(:,6);
StrJ=[sprintf('Jx= %g; ',J(1,1)),sprintf('Jy= %g; ',J(2,2)),sprintf('Jz= %g; ',J(3,3))];
StrU=[sprintf('OMx= %g; ',UgSk0(1)),sprintf('OMy= %g; ',UgSk0(2)),sprintf('OMz= %g; ',
'UgSk0(3))];
StrJ1=[sprintf('Jxy= %g; ',J(1,2)),sprintf('Jxz= %g; ',J(1,3)),sprintf('Jyz= %g; ',J(2,3))];
tt=tout; omx=yout(:,1); omy=yout(:,2); omz=yout(:,3);
epsx=yout(:,4); epsy=yout(:,5); epsz=yout(:,6);
% Виведення графіків
figure
subplot(2,1,1), h=plot(tt,omx,'-',tt,omy,'o-',tt,omz,'--');grid
set(h,'LineWidth',2); set(gca,'LineWidth',2)
set(gca,'FontAngle','italic','FontSize',16)
title('Проекції кутової швидкості'), ylabel('рад / с'),
xlabel([StrJ,' ',StrJ1,' ',StrU])
legend('\omega_X','\omega_Y','\omega_Z',0)

subplot(2,1,2), h=plot(tt,epsx,'-',tt,epsy,'o-',tt,epsz,'--');grid
set(h,'LineWidth',2); set(gca,'LineWidth',2)
set(gca,'FontAngle','italic','FontSize',16)
title('Проекції кутового прискорення'), ylabel('1/с^2'), xlabel('Час (с)')
set(gcf,'color','white'), legend('\epsilon_X','\epsilon_Y','\epsilon_Z',0)

```

Тепер введемо у блок-схему Prov.mdl два блоки Subsystem. Здійснимо у цих блоках наступні перетворення.

1. Змінимо імена блоків (підписи під їхніми зображеннями): перший блок назвемо **MENU**, а другий – **GRAFIKI** (рис. 8.17).

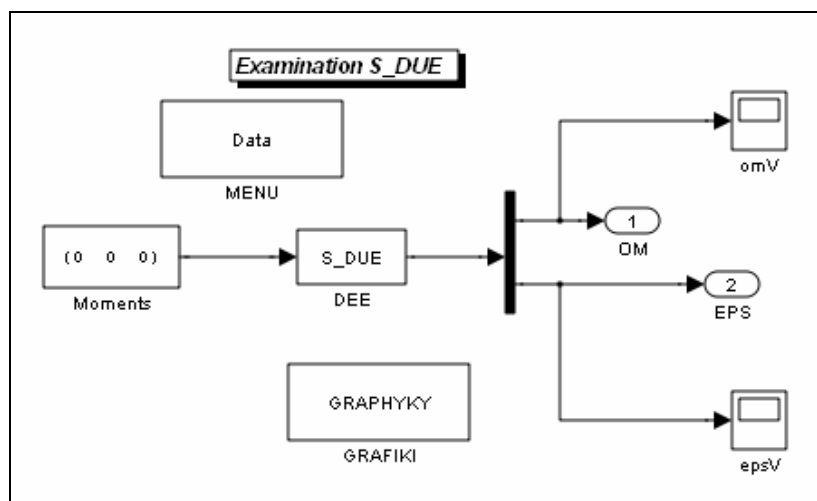


Рис. 8.17. Блок-схема S-моделі Prov1

2. Подвійно клацнувши на кожному з цих двох блоків, відчинимо їхні блок-схеми і видалимо увесь вміст – зробимо блоки порожніми.
3. У вікні блок схеми клацнемо на зображенні цих блоків правою кнопкою миші; з контекстного меню, що виникне після цього, оберемо команду *Mask Subsystem* (Створити маску підсистеми); в результаті відчиниться вікно *Mask Editor* (Редактор маски) (рис. 8.18).

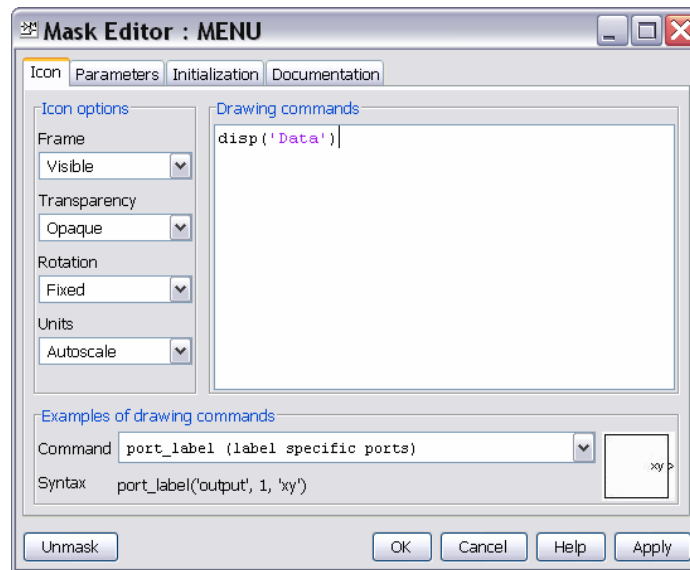


Рис. 8.18. Вікно Mask Editor

4. У полі *Drawing commands* (Команди рисування) вкладинки *Icon* (Значок) цього вікна (див. рис. 8.18) введемо команду **disp** з вказанням у якості аргумента тексту, якого потрібно розмістити на зображенні блоку; для першого блоку це буде текст *Data*, для другого – *ГРАФІКУ*.
5. Зачинемо вікно *Mask Editor* клацанням по кнопці **OK**.

Примітка. В усіх написах усередині блок-схем слід використовувати тільки латиницю. Інакше відповідні моделі можуть не сприйнятися пакетом *Simulink*.

Введемо у командному вікні команди, що пов'язують складені М-програми з S-моделлю:

```
set_param('Prov1', 'PreLoadFcn', 'Prov_DUE_Pred')
set_param('Prov1 / MENU', 'OpenFcn', 'Menu_DUE')
set_param('Prov1 / GRAFIKI', 'OpenFcn', 'Graf_DUE')
```

Після виконання цих команд керування усіма діями з моделювання буде здійснюватися з вікна самої S-моделі. Назвемо цю модифікацію S-моделі *Prov1*.

Тепер моделювання можна робити у наступному порядку.

1. Спочатку викликаємо S-модель *Prov1* шляхом введення у командному вікні Matlab команди
>> Prov1
після її виконання первісні (вшиті) значення даних вже будуть записані у робочий простір, оскільки перед появою вікна блок-схеми на екрані буде запущена на виконання програма **Prov_DUE_Pred**.
2. Двічі клацнемо у вікні блок-схеми на блоці *MENU* (Data). При цьому відчиниться вікно меню (рис. 8.19).

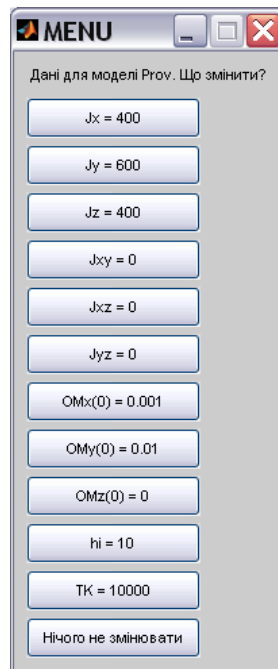


Рис. 8.19. Вікно меню змінювання даних

3. Встановивши потрібні значення параметрів, завершимо роботу з меню, клацнувши на кнопці *Нічого не змінювати*.
4. Встановимо параметри інтегрування через меню блок-схеми **Simulation > Configuration Parameters > Solver**, а саме (рис. 8.20):

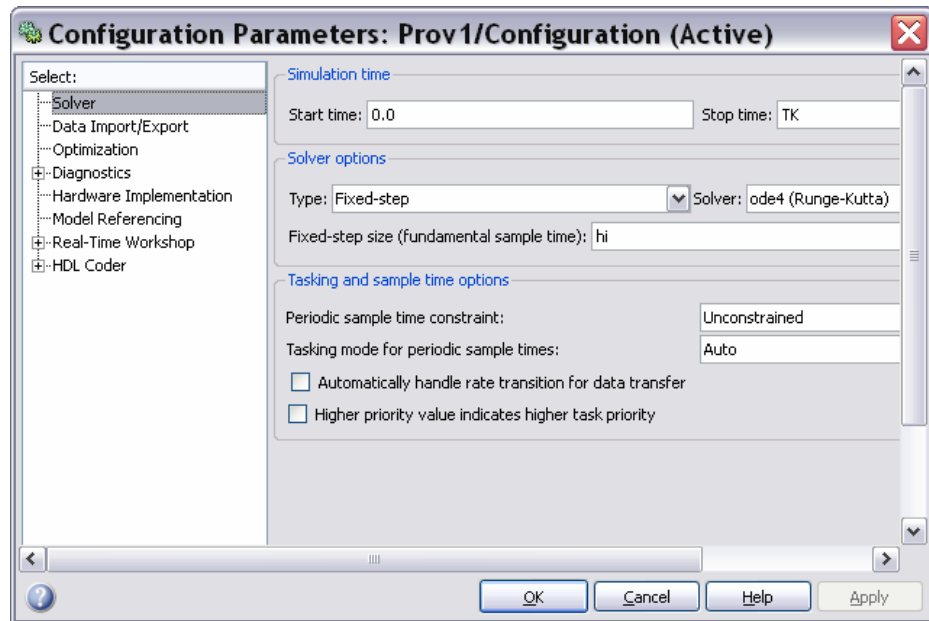


Рис. 8.20. Вікно Configuration Parameters

- а) у полі Type встановимо Fixed-step;
 - б) у полі Solver встановимо ode4;
 - в) у полі Fixed-step size введемо: hi;
 - г) у полі Stop time введемо: TK.
5. Запустимо S-модель на моделювання, клацнувши на кнопці з трикутною стрілкою на панелі інструментів вікна блок-схеми.
 6. По закінченні процесу моделювання для виведення графіків у графічне вікно двічі клацнемо на блоці **ГРАФІКИ**

Такий спосіб зв'язування S-моделі з існуючими M-файлами є, мабуть, найбільш зручним, бо, по-перше, дозволяє викликати M-файли лише у випадку потреби і у довільному порядку, а по-друге, керування моделюванням і викликом програм здійснюється лише з блок-схеми S-моделі.

На рис. 8.21-8.23 наведені результати моделювання вільного обертання тіла при різних сполученнях параметрів. В усіх випадкаху тілу попередньо надано обертання навколо осі Y з кутовою швидкістю $\omega_Y = 0,01$ рад/с.

Графіки на рис. 8.21 відповідають динамічно несиметричному, але динамічно збалансованому тілу ($J_{XY} = J_{XZ} = J_{YZ} = 0$; $J_X = 400$; $J_Y = 600$; $J_Z = 200$ Н м с²). Тіло має початкову швидкість $\omega_X = 0,002$ рад/с навколо осі X .

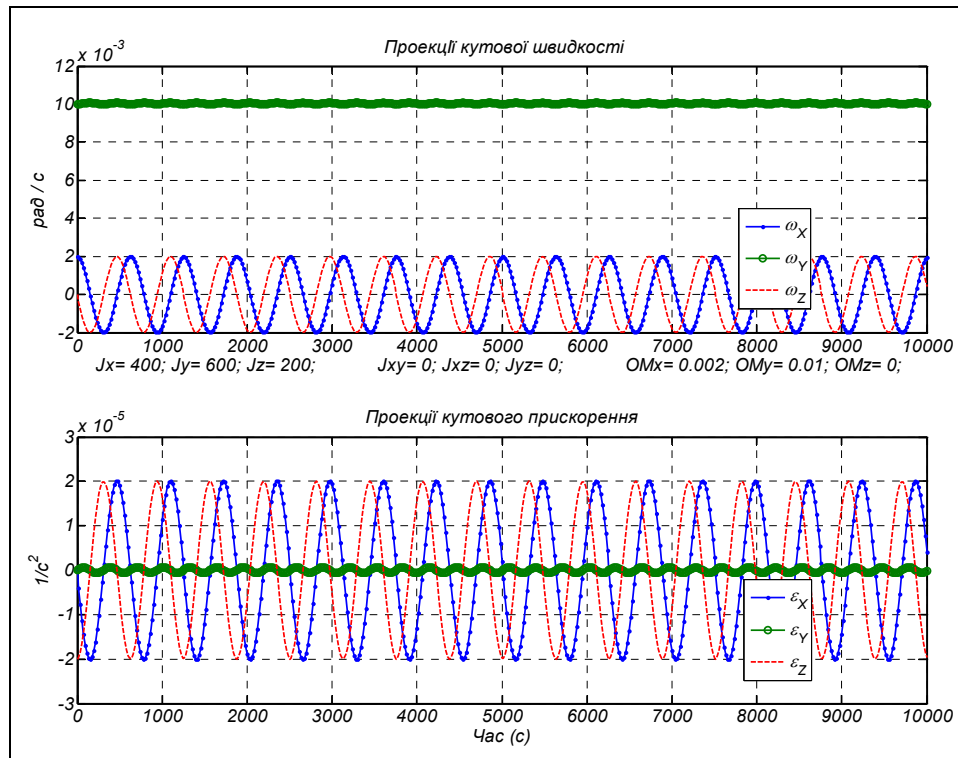


Рис. 8.21. Нутаційні коливання несиметричного гіроскопа

У наступних двох випадках $\omega_x = 0$, тобто тіло у початковий момент часу обертається лише навколо осі Y .

У другому випадку тіло є динамічно незбалансованим ($J_{xy} = -40$; $J_{xz} = 100$; $J_{yz} = 0$; $J_x = 400$; $J_y = 600$; $J_z = 400$ Н м с²).

Третій варіант відповідає випадку, коли тіло є динамічно і несиметричним, і незбалансованим ($J_{xy} = -40$; $J_{xz} = 100$; $J_{yz} = 0$; $J_x = 400$; $J_y = 600$; $J_z = 200$ Н м с²).

Як можна впевнитися по результатах цих "еспериментів", вільний обертальний рух тіла суттєво залежить від його інерційних характеристик.

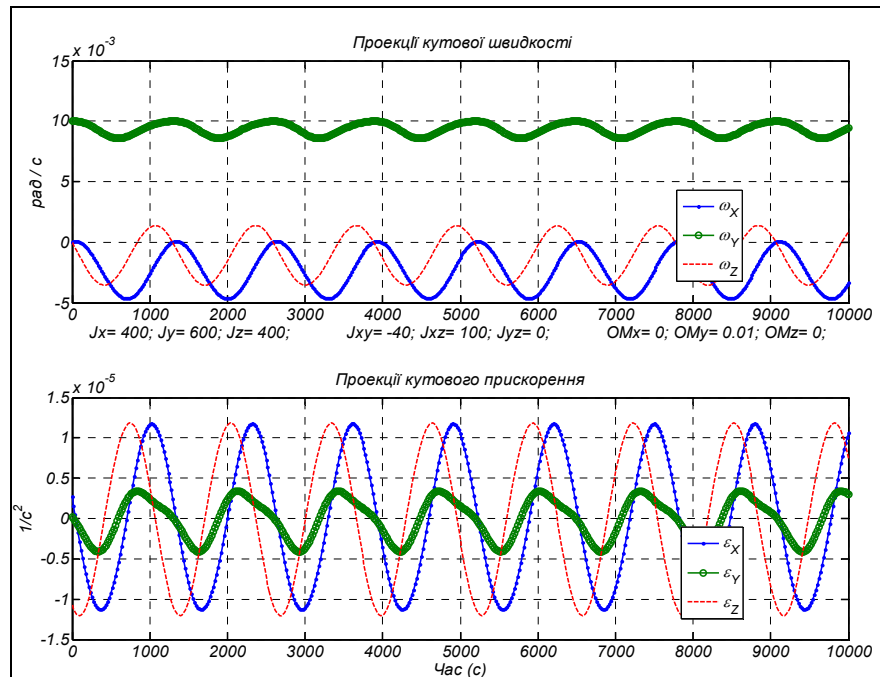


Рис. 8.26. Вільний рух динамічно незбалансованого гіроскопа

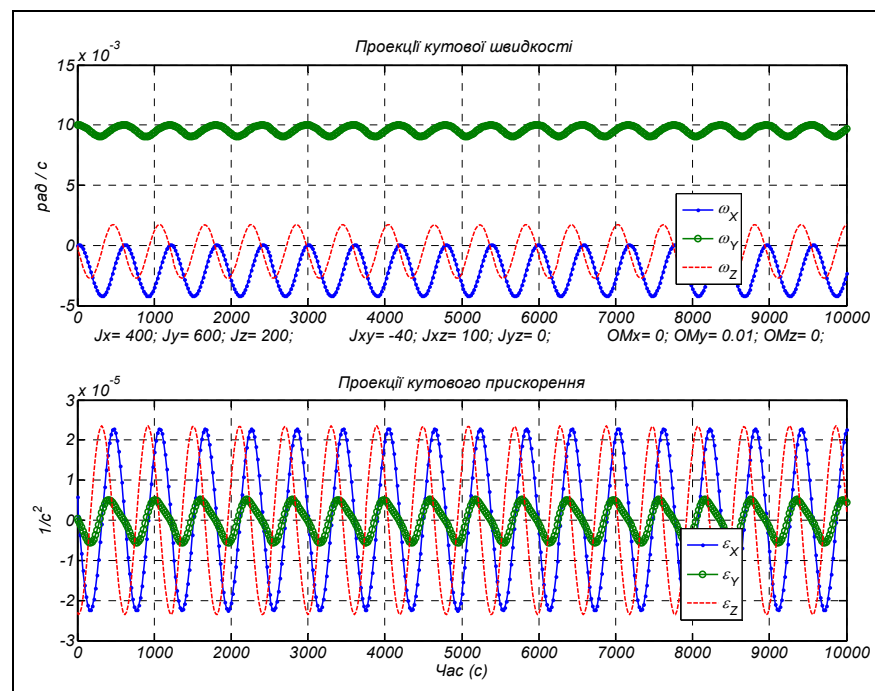


Рис. 8.23. Вільний рух несиметричного і незбалансованого гіроскопа

8.2. Користувацькі бібліотеки S-блоків

Ті, хто займається моделюванням систем, рано чи пізно нашоувуються на необхідність підготовки власних блоків, що мають властивості стандартних бібліотечних блоків пакету Simulink. Потреба у цьому виникає, коли користувач при виконанні різних задач моделювання в власній предметній області змушений неодноразово застосовувати створені їм елементарні блоки, які є оригінальними і не входять у склад стандартних бібліотек Simulink, або багаторазово використовувати ті самі блоки у певних стійких їхніх сполученнях. У таких випадках, розмістивши нові блоки у бібліотеці, можна значно скоротити час утворення нових моделей того самого типу і запобігти появі похибок.

Перевага розміщення власних блоків у бібліотеці користувача полягає у тому, що їх можна застосовувати неодноразово, перетягуючи зображення блоку з бібліотеки у вікно блок-схеми. Користуватися такими блоками зручніше за все через спеціальні вікна настроювання, аналогічних тим, які розглядалися при опису стандартних блоків Simulink.

Створення вікон настроювання блоків здійснюється через формування так званої маски блоку, яка й відіграє роль вікна настроювання.

8.2.1. Утворення бібліотеки

Розглянемо процес створення бібліотеки S-блоків на конкретних прикладах.

Формування нової бібліотеки починається з відкриття вікна нової блок-схеми моделі. У цьому вікні слід викликати команду File > New > Library (Файл > Новий > Бібліотека). В результаті на екрані виникне порожнє вікно бібліотеки (рис. 8.24) з ім'ям Library: untitled1. У цьому вікні можна утворювати S-блоки, можна також перетягувати в нього блоки, які вже створені.

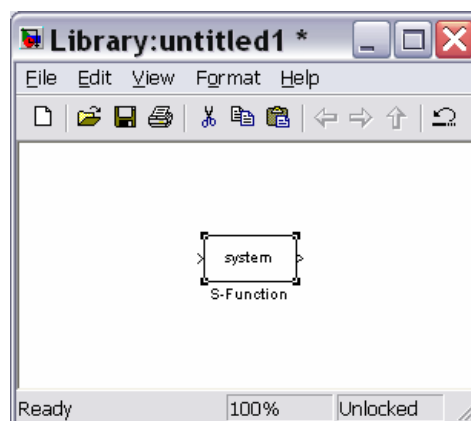


Рис. 8.24. Вікно новоутворюваної бібліотеки

У загальному випадку сформувати S-блок можна на основі стандартних блоків двох видів: блоку *S-function* з поділу *User-defined Function* бібліотеки Simulink і блоку *Subsystem* з поділу *Ports & Subsystems* той самої бібліотеки.

При утворенні S-блоку на ґрунті блоку **S-function** використовуються файли S-функцій, написані мовою Matlab; такий блок має лише один вхід (можливо, векторний) і один вихід (векторний). S-блок, утворюваний на основі блоку **Subsystem**, являє собою блок-схему, яка вміщує вже існуючі блоки, і може мати довільну кількість входів і виходів різного виду.

Утворимо у відчиненій нами бібліотеці S-блок, який назвемо **S_DUE**, на основі створеної раніше однойменної S-функції.

1. Перетягнемо у вікно утворюваної бібліотеки блок **S-function** з поділу *User-defined Function* бібліотеки Simulink. Вікно бібліотеки набуде виду, наведеному на рис. 8.24.

2. Двічі клацнувши на зображенні цього блоку, викличемо вікно його налаштування (рис. 8.25).

3. У полі *S-function name* (Ім'я S-функції) вікна налаштування введемо ім'я **S_DUE**, а у полі *S-function parameters* (Параметри S-функції) – параметри $J, UgSk0$, потім клацнемо на кнопці ОК. В результаті (за умови, що відповідний файл міститься у папках, досяжних для Matlab, а список введених параметрів відповідає списку параметрів, вказаних у S-функції), вікно налаштування зникне, і зображення блоку у вікні бібліотеки зміниться (рис. 8. 26).

4. Щоб точніше відобразити сутність перетворень, які здійснює блок, надамо йому назву **Euler dynamic equations**.

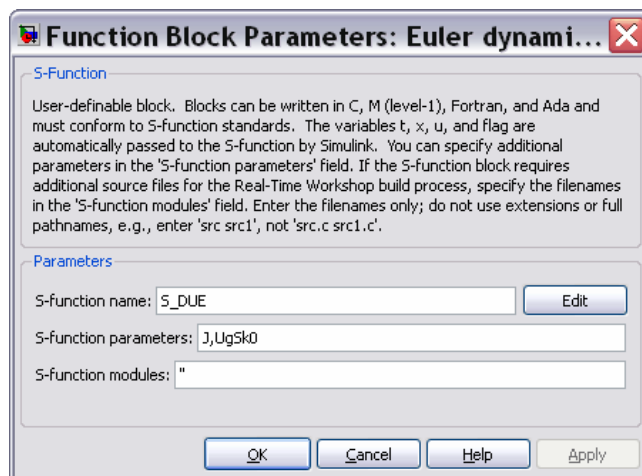


Рис. 8.25. Вікно налаштування блоку **S_DUE**

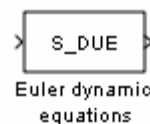


Рис. 8.26. Зображення блоку **S_DUE**

У подальшому для моделювання процесу керування орієнтацією, наприклад, космічного апарату (КА), що рухається навколо планети за певною замкненою орбітою, знадобиться ще один блок, який здійснює інтегрування кінематичних рівнянь орієнтації. В результаті обчислюються значення параметрів,

які визначають поточне кутове положення корпусу КА відносно орбітальної системи координат. Якщо у якості такого параметру обрати кватерніон повороту \mathbf{Q} , який описує перехід від орбітальної системи координат до зв'язаною з корпусом КА, відповідні кінематичні рівняння набудуть наступних кватерніонної форми:

$$\frac{d\mathbf{Q}}{dt} = \frac{1}{2}(\mathbf{Q} \circ \boldsymbol{\omega} - \boldsymbol{\Omega} \circ \mathbf{Q}),$$

де $\boldsymbol{\omega}$ - вектор-кватерніон абсолютної кутової швидкості КА; $\boldsymbol{\Omega}$ - вектор-кватерніон абсолютної кутової швидкості орбітальної системи координат (жорстко зв'язаної з положенням КА на орбіті); \circ - знак кватерніонного добутку.

Кватерніонне кінематичне рівняння не вельми зручно використовувати для проведення обчислень, з огляду на те, що дії над кватерніонами суттєво відрізняються від дій над матрицями і не передбачені у Matlab. Доцільніше перетворити це рівняння у систему матричних рівнянь:

$$\begin{cases} \frac{dq_0}{dt} = -\frac{1}{2} \mathbf{q}^t (\boldsymbol{\omega} - \boldsymbol{\Omega}) \\ \frac{d\mathbf{q}}{dt} = \frac{1}{2} [q_0 (\boldsymbol{\omega} - \boldsymbol{\Omega}) + (\mathbf{q} \times) (\boldsymbol{\omega} + \boldsymbol{\Omega})] \end{cases} \quad (8.5)$$

У цих рівняннях величини \mathbf{q} , $\boldsymbol{\omega}$ і $\boldsymbol{\Omega}$ є векторами-стовпцями з проекцій, відповідно, векторної частини кватерніону, вектора абсолютної кутової швидкості КА на осі зв'язаної з корпусом КА системи координат, і вектора кутової швидкості орбітальної системи координат на її ж осі; q_0 - скалярна частина кватерніона повороту; $(\mathbf{q} \times)$ - кососиметрична матриця, складена з проекцій вектора \mathbf{q} .

Створимо М-файл S-функції, що здійснює інтегрування цих кінематичних рівнянь. Нижче наведений текст М-файла за ім'ям S_KUqwat.

```
function [sys,x0,str,ts] = S_KUqwat(t,x,u,flag,OM0,Qw0)
% S-функція S_KUqwat Кінематичні Рівнянь орієнтації у кватерніонах
% Реалізує перехід від заданого вектора абсолютної
% кутової швидкості орбітального космічного апарату
% до кватерніону поворота КА відносно орбітальної системи координат
% ВХІД блоку:
%   u = [omx,omy,omz]- вектор проекцій абсолютної кутової
%   швидкості КА на осі СК, жорстко звязаної з ним
% ВИХІД блоку:
%   y=[qw0,qw1,qw2,qw3] - вектор компонентів кватерниона повороту
%   відносно орбітальної декартової системи координат
% Вхідні ПАРАМЕТРИ S-функції:
%   OM0 - орбітальна кутова швидкість;
%   Qw0 - вектор початкового кватерніона повороту

% Лазарєв Ю.ф. Україна 28-07-2009
switch flag,
case 0
    [sys,x0,str,ts] = mdlInitializeSizes(Qw0);
case 1,
```

```

sys = mdlDerivatives(t,x,u,OM0);
case 3,
  sys = mdlOutputs(x);
case 9
  sys = [];
end
% Кінець процедури
%
%=====
% Далі йдуть тексти внутрішніх процедур
%=====
%
function [sys,x0,str,ts] = mdlInitializeSizes(Qw0)
sizes = simsizes;
sizes.NumContStates = 4;    sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;      sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;  sizes.NumSampleTimes = 1;
sys = simsizes(sizes);     x0 = Qw0;          str = [];      ts = [0 0];
% Кінець процедури mdlInitializeSizes

%=====
function z = mdlDerivatives(t,x,u,OM0)
% ВХІДНИЙ вектор "u" - вектор проєкцій абсолютної кутової швидкості КА
% Вектор змінних стану X - складові кватерніона повороту
% x(1)=qw0 - скалярна частина кватерніона;
% x(2:4)=qw(1:3) - векторна частина кватерніона;
% Вектор похідних змінних стану
% z(1)=d(qw0)/dt; z(2:4)=d(qw)/dt; ;
% Формування векторів кутових швидкостей и кватерніона
om=u;    OM = [0 OM0 0]'; v=x(2:4);
omMOM=om-OM; omPOM=om+OM;
z(1)=-(v'*omMOM)/2; % уравнения скалярной части кватерніона
z4=(x(1)*omMOM+cross(v,omPOM))/2;% уравнения векторной части кватерніона
z(2:4)=z4;
% Кінець процедури mdlDerivatives

%=====
function y = mdlOutputs(x)
y=x;
% Кінець процедури mdlOutputs

```

За аналогією з попереднім утворимо S-блок за ім'ям *S_KUquat*. Його входом є вектор проєкцій абсолютної кутової швидкості КА, а виходом – вектор з чотирьох компонентів кватерніона поворота КА відносно орбітальної системи координат. Перший компонент являє собою скалярну частину, решта три – проєкції векторної частини цього кватерніона. В результаті отримуємо вікно бібліотеки у виді, поданому на рис. 8. 27.



Рис. 8.27. Вид бібліотеки з доданим блоком *S_KUEquat*

Нарешті, досить важливо утворити S-блок, який здійснював би операцію векторного множення двох векторів, аналогічну M-функції *cross*.

Для цього зручніше використати інший стандартний блок *SubSystem* з поділу *Ports & Subsystems*. Перетягнемо його мишкою у вікно нової бібліотеки (рис. 8.28).

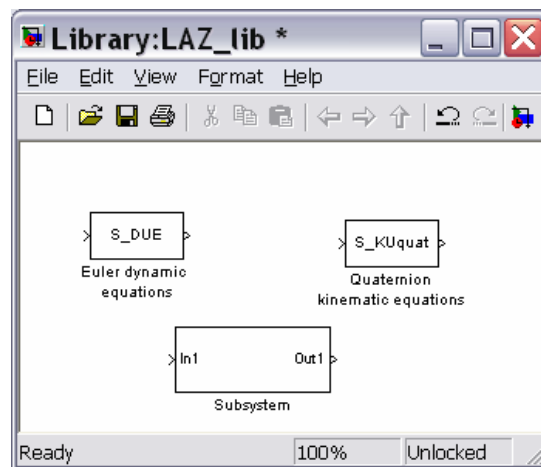


Рис. 8.28. Включення у бібліотеку блоку *Subsystem*

Двічі клацнувши на зображенні цього блоку, одержимо порожнє вікно, у якому складемо блок-схему підсистеми, наведену на рис. 8.29.

В ній використаний блок *MATLAB Function* з поділу *User-Defined Functions*, вікно настроювання якого подано на рис. 8.30. Саме він, власне, й виконує операцію векторного множення двох вхідних векторів, використовуючи для цього стандартну функцію **cross** системи MatLab.

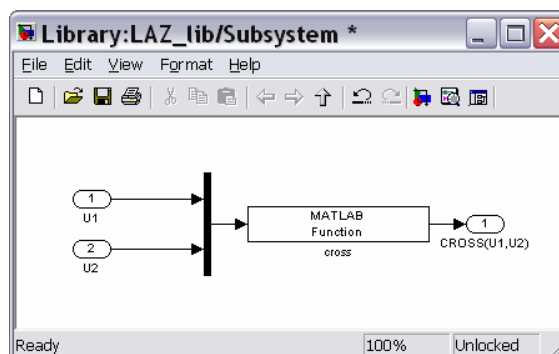


Рис. 8.29. Блок-схема підсистеми векторного добутку

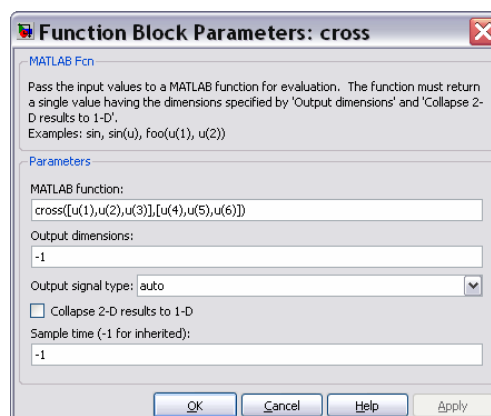


Рис. 8.30. Вікно настроювання блоку *Cross* (*MATLAB Function*)

У підсумку нами створена бібліотека, яка складається з трьох нових власних S-блоків. Запишемо її як `LAZ_lib` (рис. 8.31).

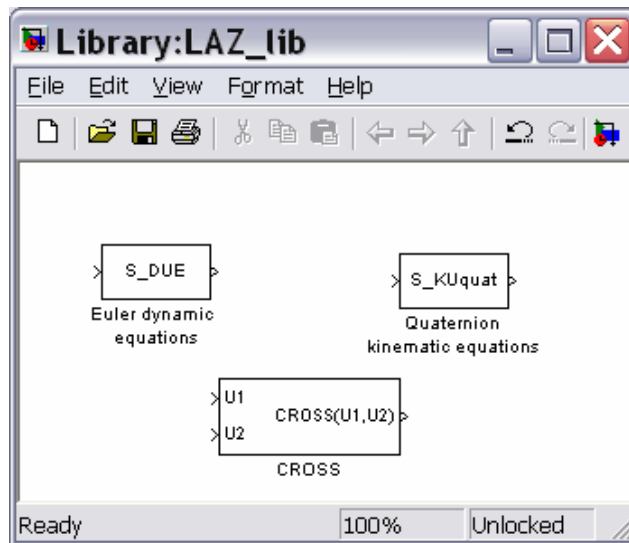


Рис. 8.31. Бібліотека користувача `LAZ_lib`

8.2.2. Утворення вікна настроювання (маски) блоку

Розглянемо процес створення вікна настроювання новоутвореного S-блоку. Це вікно називається по іншому маскою блоку і є зручним засобом встановлення і змінювання параметрів блоку.

Перш за все, потрібно виділити у бібліотеці той блок, для якого бажано утворити маску. Нехай це буде блок `S_DUE` нової бібліотеки. Потім слід виконати команду `Edit > Mask S-Function` (Редагування > Маскування S-функції) вікна бібліотеки, де міститься виділений блок. На екрані виникне вікно `Mask editor` редактора маски, подане на рис. 8.36.

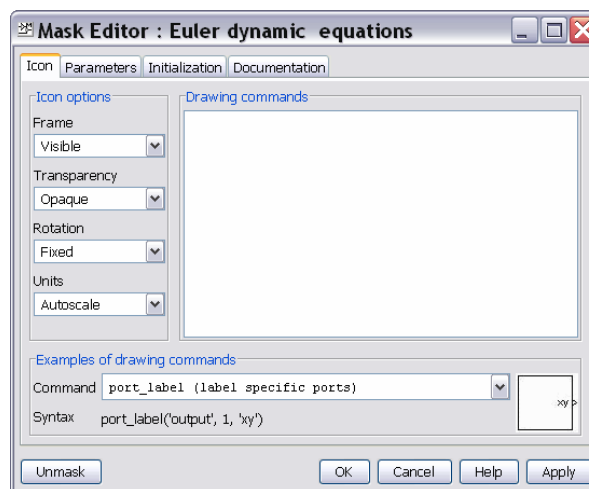


Рис. 8.36. Вікно `Mask Editor`

Примітка. Можливо, що при повторному виклику бібліотеки команда `Edit > Mask S-Function` буде не досяжною (не активованою). У цьому випадку зверніть увагу

на команду *Unlock Library* (Розблокувати бібліотеку) у тому ж меню – вона має бути активною. Після клацання на ній команда *Mask S-Function* має також стати активною.

Вікно *Mask Editor* (рис. 8.32) має чотири вкладинки:

- Icon* для створення і редагування зображень на блоці;
- Parameters* для створення і редагування діалогової частини (введення параметрів) вікна настроювання;
- Initialization* для введення деяких команд середовища MatLab при ініціалізації блоку;
- Documentation* для оформлення і редагування тексту і довідкової частини маски.

Вікна настроювання (маски) мають, в загальному випадку, три частини. Верхня частина містить інформацію про призначення блоку, його головних параметрах і правилах, яких слід дотримуватися при встановленні параметрів блоку і цього застосуванні. У середній частині маски містяться поля введення параметрів блоку, написи над ними пояснюють сенс параметрів. Нижня частина вікна настроювання містить стандартні кнопки *OK*, *Cancel*, *Help* і *Apply*.

Редактор маски призначений для оформлення перших двох частин вікна настроювання, а також утворення довідки, яка викликається при клацанні на кнопці *Help*.

Перейдемо до вкладинки *Documentation* (рис. 8.33). У вікно *Mask type* (Тип маски) слід ввести ім'я блоку. У вікно *Mask description* (Опис маски) записується інформація, яка має бути відображеною у довідковій верхній частині вікна настроювання, а у полі *Mask help* (Довідка маски) – додаткова довідкова інформація, яка має виникати після клацання на кнопці *Help*.

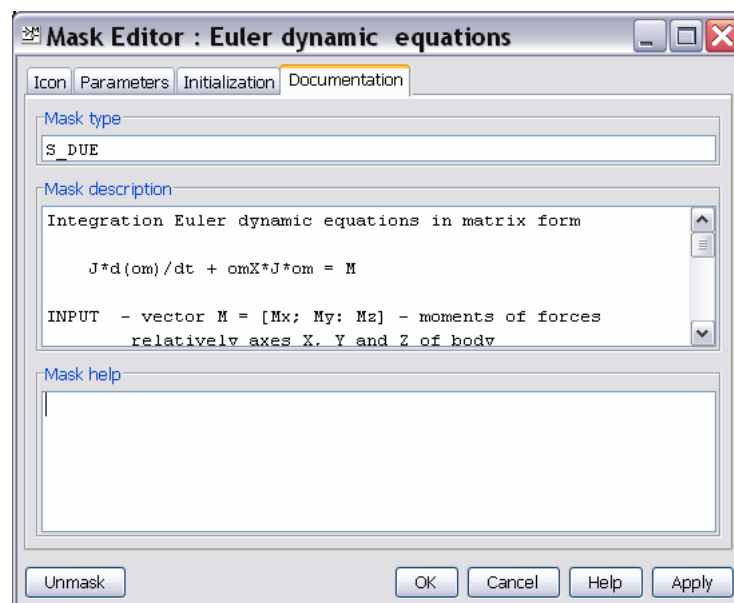


Рис. 8.33. Вкладка *Documentation* вікна *Mask Editor*

Увага! Поява у тексті документації маски знаків кирилиці зазвичай приводить до порушень роботи маски і блоки. Тому бажано при написанні тексту документів маски застосовувати виключно знаки латиниці.

За допомогою вкладки Parameters (Параметри) (рис. 8.34) можна сконструювати найважливішу частину маски – діалогову. Як видно з рисунку, головний простір цієї вкладки займає область (поки що недосяжна) Dialog Parameters (Параметри діалога). В ній вводяться параметри, які визначають кількість полів введення у вікні настроювання, написи над ними і ймення, за якими вони фігуруватимуть у блоці. Ліворуч від вказаної області знаходиться єдина досяжна (активна) кнопка (вгорі) з зображенням стрілки. Це кнопка Add (Додати). Клацання на цій кнопці приводить до активізації області Dialog Parameters – в ній виникає рядок, в який слід ввести параметри.

У блоці *S_DUE* є два параметри, значення яких потрібно вводити у діалоговому режимі, - матриця моментів інерції тіла J розміром 3×3 і вектор $UgSk0$ початкових значень трьох проекцій кутової швидкості тіла. Тому потрібно створити два поля для введення значень вказаних параметрів і зробити написи до них.

Додавання чергового поля у маску здійснюється через клацання мишкою на кнопці Add (Додати). Сам запис виконується в області *Dialog Parameters* (Параметри діалога).

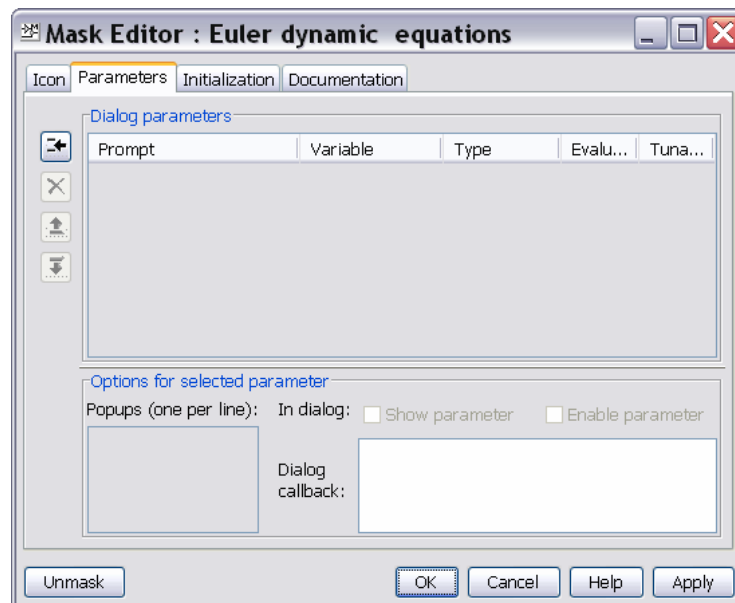
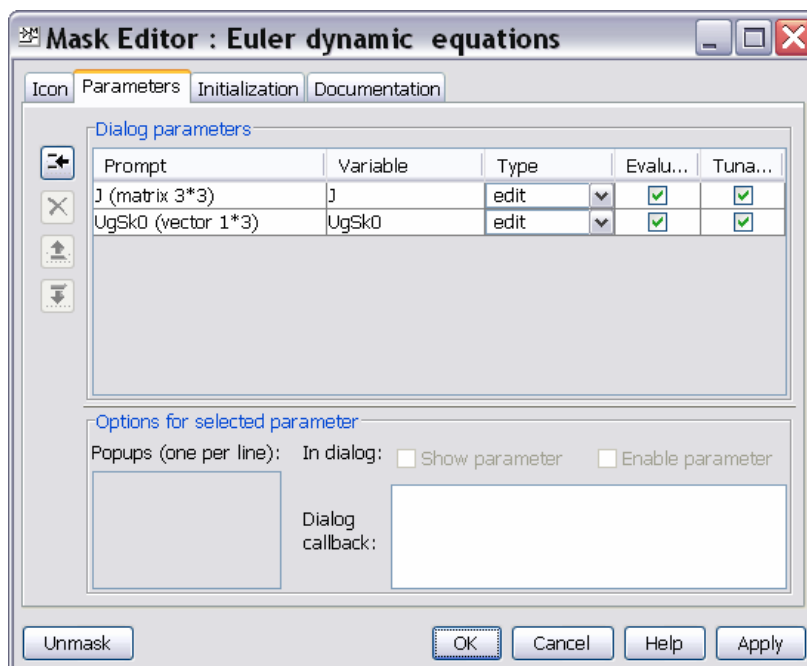
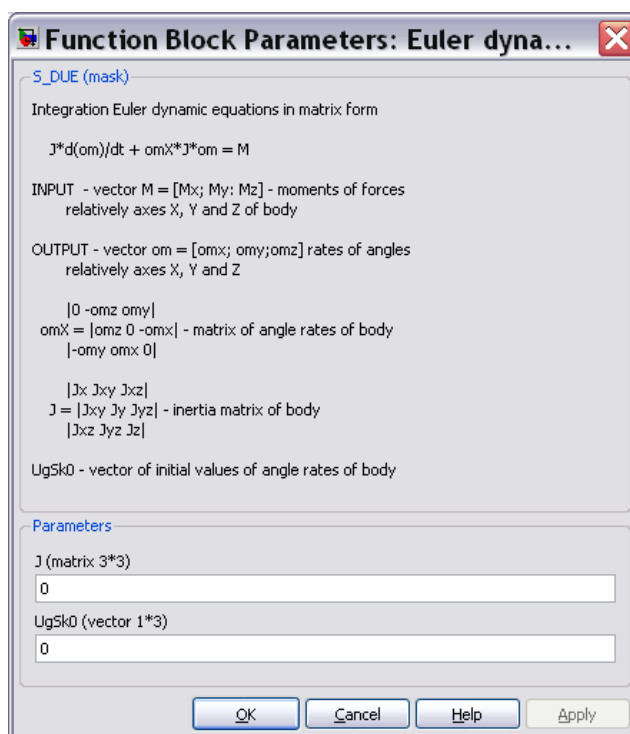


Рис. 8.34. Вкладка Parameters вікна Mask Editor

У стовпець Prompt (Підказка) записується текст напису, що міститься над полем введення, а у стовпець Variable – ім'я, під яким введена величина фігуруватиме у блоці (рис. 8.35).

Рис. 8.35. Введення діалогових параметрів маски блоку S_DUE

Для завершення процесу створення маски, клацніть на кнопці ОК у вікні редактора маски, перейдіть у вікно бібліотеки і двічі клацніть на зображенні блоку S_DUE . На екрані виникне вікно настроювання цього блоку (рис. 8.36).

Рис. 8.36. Вікно настроювання (маска) блоку S_DUE

У такий самий спосіб утворюється вікно настроювання блоку S_KUquat (рис. 8.37)

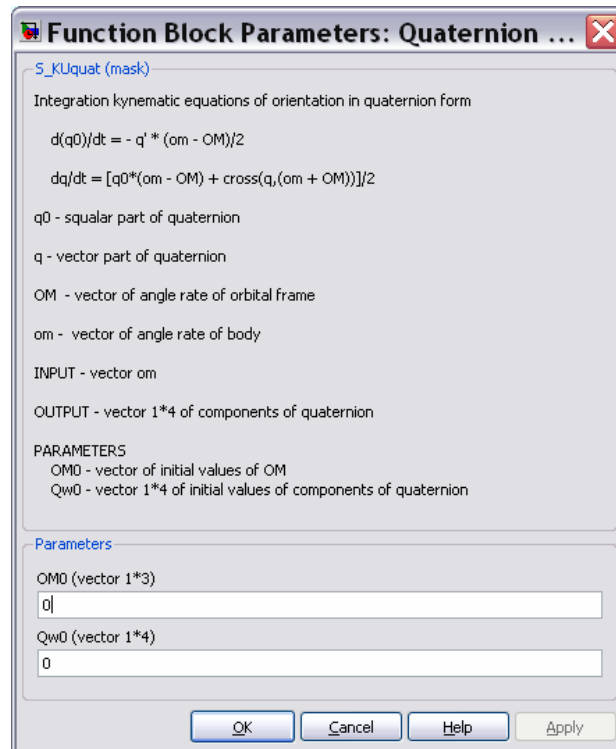


Рис. 8.37. Вікно настроювання (маска) блоку S_KUquat

8.3. Приклади застосування користувацької бібліотеки

8.3.1. Орієнтування космічного апарату

Як приклад застосування блоків власної бібліотеки розглянемо процес утворення S-моделі і комплексу M-програм, призначених для моделювання процесу автоматичного керування орієнтацією космічного апарату (КА), зокрема, штучного супутника Землі (ШСЗ).

Задля спрощення будемо припускати космічний апарат твердим тілом, яке обертається навколо Землі за замкненою орбітою. Керування орієнтацією (тобто кутовим положенням відносно орбітальної системи координат) здійснюється за допомогою трьох маховичних двигунів, осі яких збігаються з осями декартової системи координат, жорстко зв'язаної з корпусом космічного апарату. Маховичні двигуни, з одного боку, виконують розгін відповідного ротора у відповідності до рівнянь

$$\frac{dH_k}{dt} = M_k, \quad (k = x, y, z); \quad (8.6)$$

а, з іншого боку, накладають відповідний моменту сил (але у протилежному напрямку) навколо осі обертання ротора на корпус самого космічного апарату. Останній момент викликає змінювання кутового руху КА навколо цієї осі, тобто здійснює керування орієнтацією.

Змінювання кутової орієнтації космічного апарату у просторі підпорядковується основним законам механіки, з яких випливають рівняння, воплощенные в блоках **S_DUE** и **S_KUquat**.

Складемо блок-схему S-моделі системи орієнтації і збережемо її у файлі **SUO_KA.mdl** (рис. 8.38). Она приведена на рис. 8.38.

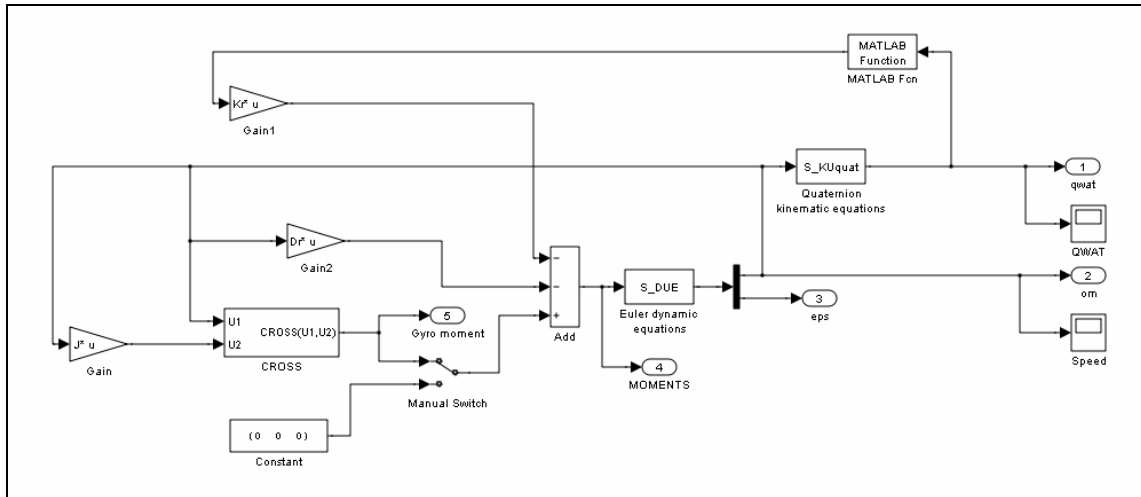


Рис. 8.38. Блок-схема S-моделі системи керування орієнтацією КА

Блок-схема складається з послідовно з'єднаних блоків **S_DUE** і **S_KUquat**, охоплених трьома ланцюгами зворотного зв'язку, які забезпечують керування космічним апаратом по кватерніону, кутовій швидкості і компенсацію гіроскопичного моменту, що виникає при поворотах КА. Отримане на виході блоку **S_KUquat** значення кватерніону поворота надходить до входу блоку **MATLAB Function**, який виділяє векторну частину цього кватерніону, необхідну для формування вектора моменту керування кутовим положенням КА (рис. 8. 39).

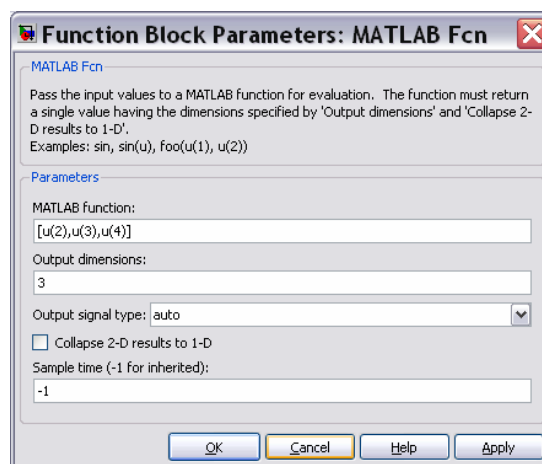


Рис. 8.39. Вікно налаштування блоку MATLAB Fcn

У цілому момент сил керування формується за наступним матричним законом:

$$\mathbf{M} = -\mathbf{K}_r \cdot \mathbf{q} - \mathbf{D}_r \cdot \boldsymbol{\omega} + (\boldsymbol{\omega} \times) \cdot (\mathbf{J} \cdot \boldsymbol{\omega}). \quad (8.7)$$

У ньому можна розрізнити три складові:

- складову, пропорційну векторній частині кватерніона відхилення поточного положення КА від заданого його положення (у цій програмі задане положення відповідає нульовому значенню векторної частини кватерніона); саме ця складова моменту керування змушує КА наближатися до заданого кутового положення;

- складову, пропорційну вектору кутової швидкості КА; вона забезпечує демпфірування процесу наближення КА до заданого положення;

- третя складова моменту вводиться для того, щоб компенсувати виникаючий при кутовому русі КА гіроскопічний момент, який прагне повернути КА навколо осі, перпендикулярної осі дії моменту сил керування; введення цієї складової змушує корпус КА повертатися до заданого положення за найкоротшим шляхом, тобто зменшує енергетичні витрати на переорієнтацію КА.

Матриці K_r і D_r визначають закон керування і мають бути відомими заздалегідь.

Формування першої складової моменту керування на блок-схемі здійснюється верхнім зворотним ланцюгом з матричним підсилювачем K_r . Друга складова формується ланцюгом з матричним підсилювачем D_r . Третя складова забезпечується третім зворотним ланцюгом, у склад якого входять матричний підсилювач J і утворений раніше блок **CROSS** (см. рис. 8. 38).

Ці три складові підсумовуються на суматорі і надходять до входу блоку **S_DUE**. У такий спосіб утворюється замкнена система керування.

Поставимо задачу промоделювати поведінку системи орієнтації КА, керованого за компонентами кватерніона при різних законах регулювання у відповідності з даними, наведеними у статті [6], тобто за наступних значень матриці інерції КА

$$J=[1200 \ 100 \ -200; 100 \ 2200 \ 300; -100 \ 300 \ 3100]$$

J =

1200	100	-200
100	2200	300
-100	300	3100

матриці моментів демпфірування

$$D_r=0.315*\text{diag}([1200 \ 2200 \ 3100])$$

D_r =

378.0000	0	0
0	693.0000	0
0	0	976.5000

і за чотирьох значень матриці позиційного керування:

1) матриця керування є пропорційною оберненій матриці моментів інерції

$$K_r=k/J=\text{diag}([201,110,78]);$$

K_r =

201	0	0
0	110	0
0	0	78

2) матриця керування є пропорційною одиничній матриці E

$$K_r=k*E=\text{diag}([110,110,110]);$$

K_r =

```
110  0  0
  0 110  0
  0  0 110
```

3) матриця керування являє собою комбінацію матриць інерції і одиничної

$K_r = (\alpha J + \beta E) = \text{diag}([72, 110, 204]);$

```
Kr =
  72  0  0
  0 110  0
  0  0 204
```

4) матриця керування є пропорційною матриці J моментів інерції

$K_r = k \cdot J = \text{diag}([60, 110, 155])$

```
Kr =
  60  0  0
  0 110  0
  0  0 155
```

Припустимо, що орбітальна кутова швидкість дорівнює нулю, а початкове відхилення положення КА від заданого визначається кватерніоном $Q_w0 = [0.159 \ 0.57 \ 0.57 \ 0.57]$

що відповідає початковому відхиленню від потрібного положення у $161,7^\circ$.

Щоб почати моделювання, потрібно присвоїти первісні значення усім параметрам. По закінченні моделювання необхідно на основі отриманих даних побудувати ряд графіків, які відбивали би процес переорієнтації КА. Виконаємо ці процедури (а також саме моделювання) за допомогою спеціального М-файла *SUO_KAupr.m*. Його текст наведений нижче

```
% SUO_KAupr.m
% Керуюча програма для запуску моделі SUO_KA.mdl

% Лазарєв Ю.Ф. 18-12-2001
% Останні змінювання 5-08-2009
clear all, clc
K=[3,1,2];  OM0=0;
% Введение значений матрицы инерции
J=[1200 100 -200; 100 2200 300; -200 300 3100]
% Введение начальных значений:
% 1) проекций угловой скорости тела
UgSk0=[0 0 0];
% 2) компонентів кватерніона поворота
Qw0=[0.159,0.57,0.57,0.57];  qw0=Qw0(1),  qw=Qw0(2:4)
U0=2*acos(qw0); Cs0=qw/sin(U0/2);  Zk=[1 0 0 0];
% Матрицы моментов УПРАВЛЕНИЯ
Dr=0.315*diag(diag(J))
V=[201,110,78;110 110 110;72 110 204;60 110 155]
for k=1:4
    % Установление параметров моделирования
    Kr=diag(V(k,:))
    options=simset('Solver','ode45','RelTol',1e-6);
    sim('SUO_KA',[0,100],options);          % МОДЕЛИРОВАНИЕ на S-модели
    % Формирование данных для вывода ГРАФИКОВ
    tt=tout;
    q0=yout(:,1); qx=yout(:,2);  qy=yout(:,3);          qz=yout(:,4);
    omx=yout(:,5);          omy=yout(:,6); omz=yout(:,7);
    Mx=yout(:,11);          My=yout(:,12); Mz=yout(:,13);
    if k==1
        t1=tt;
```

```

    q01=q0;    qx1=qx;    qy1=qy;    qz1=qz;
    omx1=omx;    omy1=omy;    omz1=omz;
    Mx1=Mx;    My1=My;    Mz1=Mz;
elseif k==2
    t2=tt;
    q02=q0;    qx2=qx;    qy2=qy;    qz2=qz;
    omx2=omx;    omy2=omy;    omz2=omz;
    Mx2=Mx;    My2=My;    Mz2=Mz;
elseif k==3
    t3=tt;
    q03=q0;    qx3=qx;    qy3=qy;    qz3=qz;
    omx3=omx;    omy3=omy;    omz3=omz;
    Mx3=Mx;    My3=My;    Mz3=Mz;
elseif k==4
    t4=tt;
    q04=q0;    qx4=qx;    qy4=qy;    qz4=qz;
    omx4=omx;    omy4=omy;    omz4=omz;
    Mx4=Mx;    My4=My;    Mz4=Mz;
end
clear tt q0 qx qy qz omx omy omz Mx My Mz
end
A=180/pi;
D1=2*acos(q01); D2=2*acos(q02); D3=2*acos(q03); D4=2*acos(q04);
dt1=[0;diff(t1)];
dHx1=Mx1.*dt1;    dHy1=My1.*dt1; dHz1=Mz1.*dt1;
domx1=[0;diff(omx1)]; domy1=[0;diff(omy1)]; domz1=[0;diff(omz1)];
DEx1=cumsum(abs(dHx1.*domx1)); DEy1=cumsum(abs(dHy1.*domy1));
DEz1=cumsum(abs(dHz1.*domz1));
d1=DEx1+DEy1+DEz1; dt2=[0;diff(t2)];
dHx2=Mx2.*dt2;    dHy2=My2.*dt2; dHz2=Mz2.*dt2;
domx2=[0;diff(omx2)]; domy2=[0;diff(omy2)]; domz2=[0;diff(omz2)];
DEx2=cumsum(abs(dHx2.*domx2)); DEy2=cumsum(abs(dHy2.*domy2));
DEz2=cumsum(abs(dHz2.*domz2));
d2=DEx2+DEy2+DEz2; dt3=[0;diff(t3)];
dHx3=Mx3.*dt3;    dHy3=My3.*dt3; dHz3=Mz3.*dt3;
domx3=[0;diff(omx3)]; domy3=[0;diff(omy3)]; domz3=[0;diff(omz3)];
DEx3=cumsum(abs(dHx3.*domx3)); DEy3=cumsum(abs(dHy3.*domy3));
DEz3=cumsum(abs(dHz3.*domz3));
d3=DEx3+DEy3+DEz3; dt4=[0;diff(t4)];
dHx4=Mx4.*dt4;    dHy4=My4.*dt4; dHz4=Mz4.*dt4;
domx4=[0;diff(omx4)]; domy4=[0;diff(omy4)]; domz4=[0;diff(omz4)];
DEx4=cumsum(abs(dHx4.*domx4)); DEy4=cumsum(abs(dHy4.*domy4));
DEz4=cumsum(abs(dHz4.*domz4)); d4=DEx4+DEy4+DEz4;
% Графики проекцій компонентів кватерніона на площину

subplot(2,2,1)
plot(qx1,qy1, qx2,qy2, ':',qx3,qy3, '--',qx4,qy4, '.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('    Проекції КОМПОНЕНТІВ кватерніонів')
xlabel('Q_x'), set(gca,'FontName','Helvetica'), ylabel('Q_y')

subplot(2,2,2)
plot(qy1,qz1,qy2,qz2, ':',qy3,qz3, '--',qy4,qz4, '.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('    на координатні площини    ')
xlabel('Q_y'), set(gca,'FontName','Helvetica'), ylabel('Q_z')

subplot(2,2,3)
plot(qx1,qz1, qx2,qz2, ':',qx3,qz3, '--',qx4,qz4, '.'), grid
set(gca,'FontSize',14)
xlabel('Q_x'), set(gca,'FontName','Helvetica'), ylabel('Q_z')
legend('Kr = k/J','Kr = kE','Kr = k/(\alphaJ+\betaE)','Kr = kJ',4)

```

```

subplot(2,2,4)
c1=D1./sin(D1/2)*A;
cx1=c1.*qx1; cy1=c1.*qy1; cz1=c1.*qz1; c2=D6./sin(D2/2)*A;
cx2=c6.*qx2; cy2=c6.*qy2; cz2=c6.*qz2; c3=D3./sin(D3/2)*A;
cx3=c3.*qx3; cy3=c3.*qy3; cz3=c3.*qz3; c4=D4./sin(D4/2)*A;
cx4=c4.*qx4; cy4=c4.*qy4; cz4=c4.*qz4;
plot3(cx1,cy1,cz1,cx2,cy2,cz2,':',cx3,cy3,cz3,'--',cx4,cy4,cz4,':'),grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('Вектор ЕЙЛЕРОВОГО повороту у просторі')
xlabel('E_x (градуси)')
ylabel('E_y (градуси)'), set(gca,'FontName','Helvetica'), ylabel('E_z')
% Графики зависимостей компонентов кватерніону от времени
figure
subplot(2,2,1)
plot(t1,qx1,t2,qx2,':',t3,qx3,'--',t4,qx4,':'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('Залежність КОМПОНЕНТІВ кватерніонів від ЧАСУ')
xlabel('Час (c)'), set(gca,'FontName','Helvetica'), ylabel('Q_x')

subplot(2,2,2)
plot(t1,qy1,t2,qy2,':',t3,qy3,'--',t4,qy4,':'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
xlabel('Час (c)'), set(gca,'FontName','Helvetica'), ylabel('Q_y')

subplot(2,2,3)
plot(t1,qz1,t2,qz2,':',t3,qz3,'--',t4,qz4,':'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
xlabel('Час (c)'), set(gca,'FontName','Helvetica'), ylabel('Q_z')

subplot(2,2,4)
plot(t1,D1*A,t2,D2*A,':',t3,D3*A,'--',t4,D4*A,':'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('Поворот навколо осі Ейлера')
xlabel('Час (c)'), set(gca,'FontName','Helvetica'), ylabel('Kut (gradus)')
legend('Kr = k/J','Kr = k','Kr = k/(alphaJ+betaE)','Kr = kJ')
% Графики зависимостей проекций момента сил от времени
figure
subplot(2,2,1)
plot(t1,Mx1,t2,Mx2,':',t3,Mx3,'--',t4,Mx4,':'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('Залежність проекцій МОМЕНТУ КЕРУВАННЯ від часу')
xlabel('Час (c)'),set(gca,'FontName','Helvetica'), ylabel('M_x'),

subplot(2,2,2)
plot(t1,My1,t2,My2,':',t3,My3,'--',t4,My4,':'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
xlabel('Час (c)'), set(gca,'FontName','Helvetica'), ylabel('M_y')

subplot(2,2,3)
plot(t1,Mz1,t2,Mz2,':',t3,Mz3,'--',t4,Mz4,':'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
xlabel('Час (c)'), set(gca,'FontName','Helvetica'), ylabel('M_z')

subplot(2,2,4)
plot(t1,d1,t2,d2,':',t3,d3,'--',t4,d4,':'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('Загальні витрати ЕНЕРГІЇ')
xlabel('Час (c)'), set(gca,'FontName','Helvetica')
ylabel('\Sigma\Delta H\Delta\omega')
legend('Kr = k/J','Kr = k','Kr = k/(alphaJ+betaE)','Kr = kJ',0)

```

Запуск цього М-файла приводить до результатів, поданих на рис. 8.40-8.46.

На рис. 8.40 наведені проекції траєкторій вектора кватерніона на усі три координатні площини, а також траєкторії у просторі вектора ейлерового поворота.

На рис. 8.41 показані графіки залежностей від часу компонентів кватерніону, а також проекцій вектора ейлерова поворота.

Рис 8.42 подає залежності проекцій моменту керування від часу, а також загальні (сумма по трьох ортогональних осях) прирости кінетичних моментів двигунів-маховиків, які забезпечують виконання цих розворотів КА. Останні характеризують у певній мірі витрати енергії на поворот КА.

Розглядаючи отримані графіки, можна дійти висновку, що найменші витрати енергії забезпечує керування за законом, коли матриця позиційного керування є пропорційною матриці моментів інерції КА. Цей закон дозволяє зекономити більш ніж у два рази у порівнянні з випадком, коли матриця коефіцієнтів позиційного керування є обернено пропорційною матриці моментів інерції.

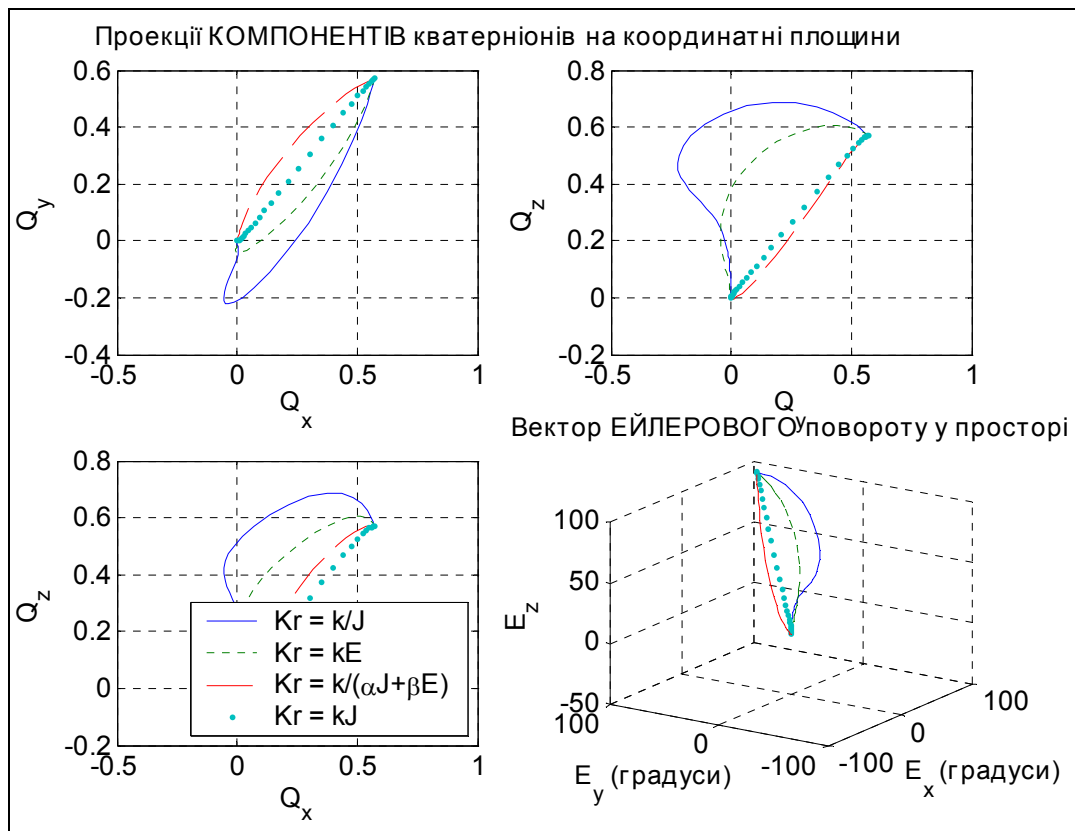


Рис. 8.40. Проекції кватерніону поворота КА на координатні площини

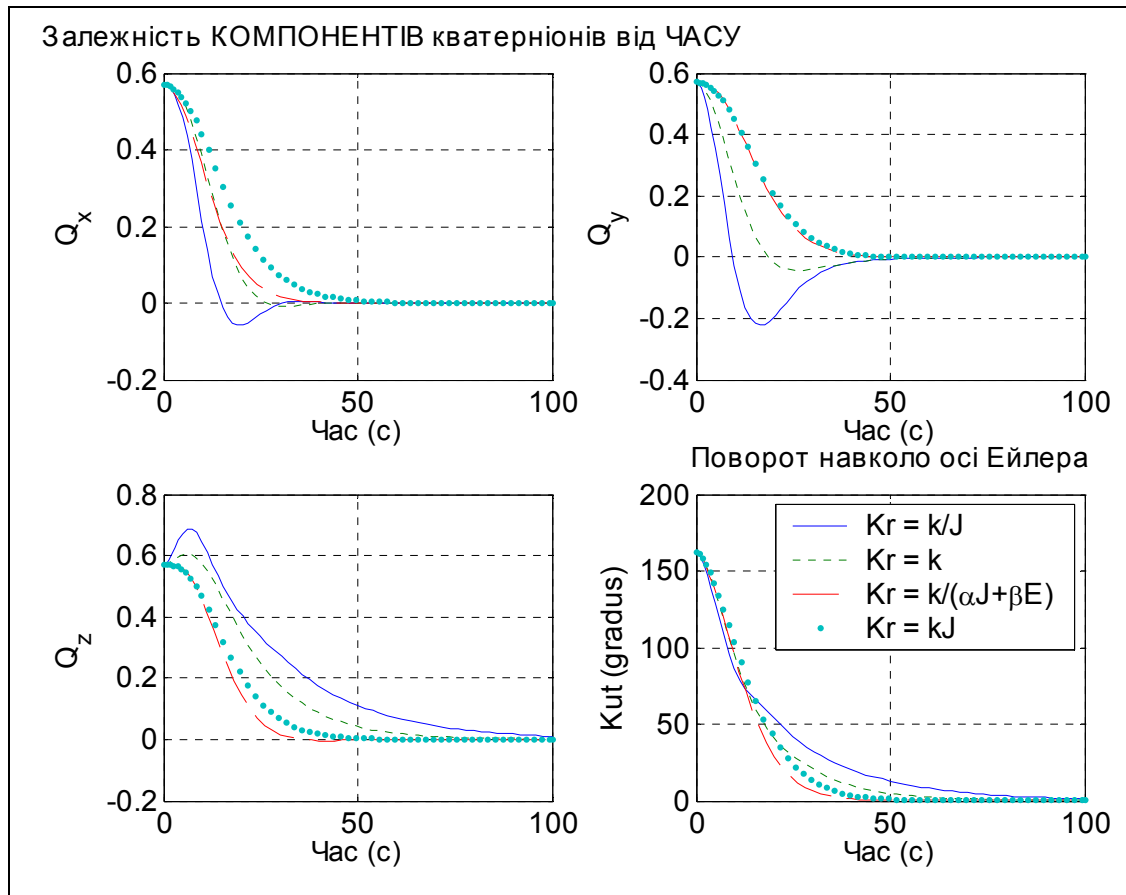


Рис. 8.41. Залежності компонентів кватерніона поворота КА від часу

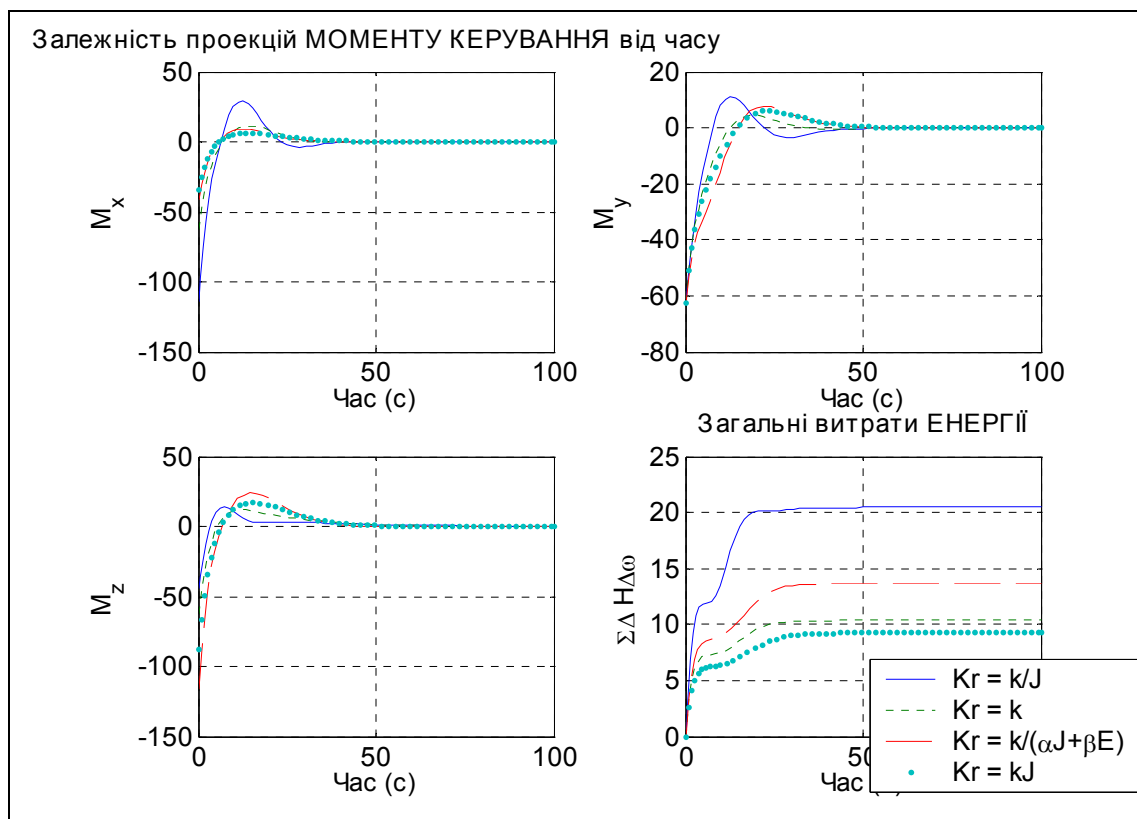


Рис. 8.46. Залежності компонентів моменту керування орієнтацією КА від часу

До того самого висновку можна дійти й суґубо теоретичним шляхом, якщо підставити вираз (8.7) моменту керування у рівняння (8.1) руху КА з врахуванням останньої залежності матриці Kr від матриці J . Якщо припустити також, що й матриця демпфірування Dr є також пропорційною матриці J з коефіцієнтом пропорційності f , то неважко впевнитися, що векторне рівняння руху у цьому випадку матиме вид:

$$\frac{d\omega}{dt} + f \cdot \omega + k \cdot q = 0,$$

і воно розпадається на три однакові незалежні рівняння руху КА відносно трьох його координатних осей.

8.3.2. Маятник під дією сил сухого тертя

Як відомо, основні властивості сили сухого тертя, що виникає при відносному переміщенні двох тіл, що труться один по одному, є наступними:

- сила тертя завжди скерована у бік, протилежний відносній швидкості руху тіл;

- величина сили тертя не залежить від величини цієї відносної швидкості.

Зазначені властивості достатньо добре описуються математично, якщо скористуватися сигнум-функцією:

$$F_{tr} = -F_T \cdot \text{sign}(V),$$

де F_T - деяка додатна величина, що дорівнює величині сили сухого тертя, а V - швидкість тіла, з боку якого діє сила тертя, відносно тіла, на яке ця сила діє.

Однак нам відома ще одна властивість сили сухого тертя:

- якщо тіла, що труться, нерухомі одне відносно одного, то прикладання зовнішньої сили до одного з них, не призведе до відносного руху тіл доти, поки діюча сила (назвемо її "активною" - F_a) не перебільшить за величиною так звану силу тертя спокою $F_p > F_T > 0$.

У цьому випадку величина сили тертя вже визначається не величиною і напрямком швидкості, а величиною і напрямком прикладеної активної сили, приймаючи таке значення і напрямком, що вона повністю компенсує дію цієї сили:

$$F_a + F_{tr} = 0, \text{ якщо } V = 0 \text{ і } |F_a| \leq F_p.$$

Ця особливість сил сухого тертя зумовлює цілу низку дивних властивостей систем, в яких діють подібні сили. Це, зокрема, таке явище, як "захоплення" або "зчеплення" одного тіла з другим, коли обидва тіла починають рухатися як одне, залишаючись нерухомими одне відносно одного.

Теоретичне досліджування цієї властивості сил сухого тертя пов'язане зі значними труднощами, оскільки залежність сили тертя від швидкості має ривний характер, а також існує складна залежність сили чсухого тертя від швидкості і активної сили.

Сформулюємо задачу опису руху механічної системи, яка знаходиться під дією сил сухого тертя. Нехай q - узагальнена координата (це може бути лінійне переміщення або кут при обертальному русі), яка відповідає відносному переміщенню тіл, що труться. Складемо узагальнене рівняння руху з цієї координати і виділимо у ньому три частини.

Узагальнена сила інерції. До цієї частини віднесемо лише члени рівняння, які є пропорційними відносному узагальненому прискоренню. Цю частину можна подати у наступному виді: $(-M_q \ddot{q})$, де M_q має сенс узагальненої маси і може залежати від узагальненої координати q .

Узагальнена сила тертя. Віднесемо до цієї частини усі члени рівняння, які визначають вплив сил сухого тертя: $Q_{tr}(\dot{q}, Q_a)$.

Активна узагальнена сила Q_a . До цієї сили віноситимемо усі решту членів рівняння.

Тоді рівняння руху з цієї координати можна подати у наступному виді:

$$\ddot{q} = \frac{1}{M_q} [Q_a + Q_{tr}(\dot{q}, Q_a)]. \quad (8.8)$$

Лише після цієї операції можна сформулювати математичну залежність узагальненої сили сухого тертя від усіх чинників, що впливають на неї у відповідності з встановленими властивостями тертя:

$$Q_{tr}(\dot{q}, Q_a) = \begin{cases} -Q_{Td}, & \text{якщо } \dot{q} > 0 \\ Q_{Td}, & \text{якщо } \dot{q} < 0 \\ -Q_a, & \text{якщо } \dot{q} = 0 \text{ і } |Q_a| < Q_{Tp} \\ Q_{Tp}, & \text{якщо } \dot{q} = 0 \text{ і } |Q_a| \geq Q_{Tp} \end{cases} \quad (8.9)$$

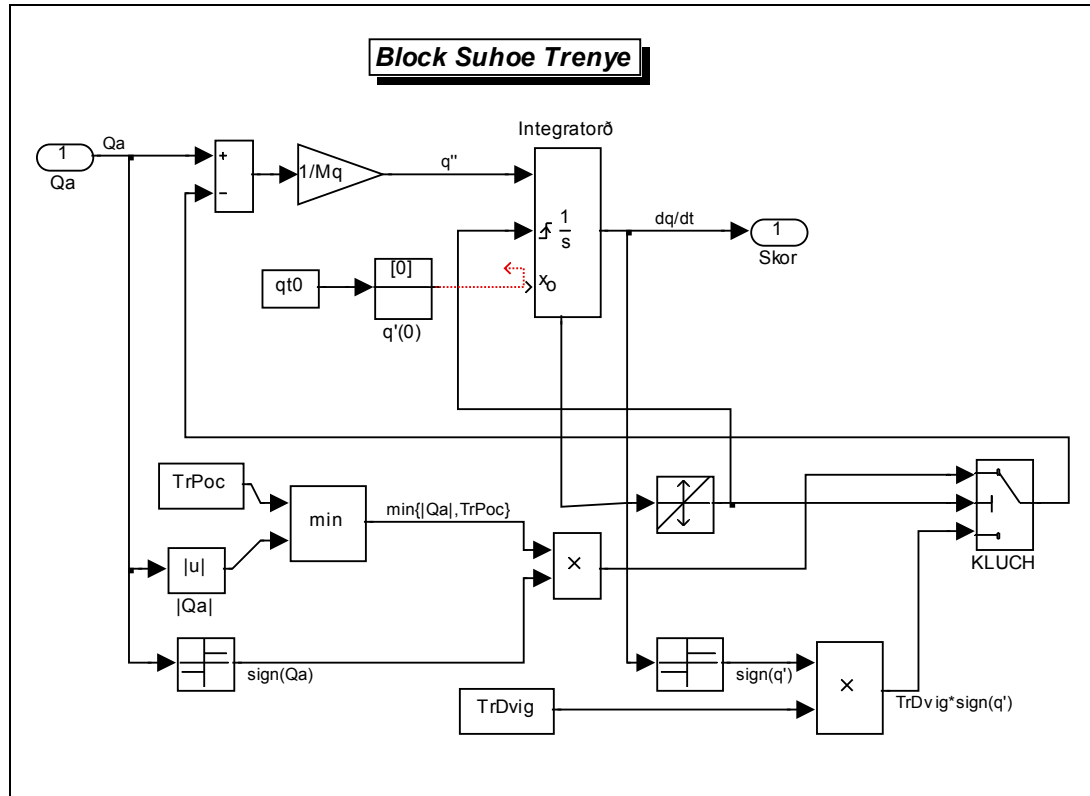
де постійні додатні величини Q_{Td} і Q_{Tp} визначають величини узагальнених сил тертя руху і спокою відповідно. При цьому зазвичай виконується співвідношення $Q_{Tp} \geq Q_{Td}$.

Як бачимо, повністю описати усі вказані особливості сил сухого тертя можна лише після того, як задані (відомі) рівняння руху і виділена так звана активна сила.

Створимо універсальний блок, що здійснює однократне інтегрування рівняння (8.8). Входом цього блоку має бути поточне значення активної сили, а виходом – поточне значення узагальненої швидкості \dot{q} . Параметри блоку приймемо наступні:

- Mq - узагальнена маса M_q ;
- $TrDvig$ - величина узагальненої сили тертя руху Q_{Td} ;
- $TrPoc$ - величина узагальненої сили тертя спокою Q_{Tp} ;
- $qt0$ - початкове значення узагальненої швидкості \dot{q}_0 .

Оформимо блок у виді підсистеми (рис. 8. 43) і назвемо його **Suhoe Trenye**.

Рис. 8.43. Блок-схема блоку *Suhoe Trenye*

Головний елемент блоку – інтегратор (*Integrator*). Він здійснює інтегрування сигналу відносного прискорення, що надходить до нього, видаючи сигнал, що дорівнює поточному значенню узагальненої швидкості. Початкове значення узагальненої швидкості задається зовнішнім блоком *IC*, до якого надається постійний сигнал з блоку *Constant*, що дорівнює встановленому в ньому початковому значенню узагальненої швидкості.

Сигнал узагальненого прискорення формірується у такий спосіб. Спочатку на суматорі (блок *Sum0* активна сила сумується з силою тертя. Результат подається на блок *Gain*, який здійснює ділення сумарного сигналу на узагальнену масу. Блоки, що формірують силу тертя, розташовані у нижній частині блок-схеми. Якщо швидкість \dot{q} не дорівнює нулеві, то перемикач у блоці *Kluch* знаходиться у нижньому положенні, і сигнал узагальненої швидкості проходить через блок *Sign* (нижня права частини схеми), помножується на постійний коефіцієнт *TrDvig* і передається до суматора як сила тертя з протилежним знаком. У такий спосіб реалізуються перші два співвідношення (8.9)

Значно складніше виконати дві останні умови (8.9). Для цього перш за все потрібно якнайточніше визначити момент часу, коли відносна швидкість проходить через нуль. Необхідно зробити наступне.

1. У блоці інтегратора потрібно відкрити порт стану *Show state port*. При цьому на зображенні блоку внизу виникне додатковий вихід – порт стану. Окрім того, потрібно підключити зовнішнє керування роботою інтегратора, встановивши для параметра *External*

reset значення *rising* (рис. 8.44). З лівого боку блоку виникне зображення ще одного вхідного порту (керуючого).

2. Порт стану інтегратора потрібно з'єднати зі входом блоку **Hit Crossing**, який здійснює фіксування точного моменту переходу швидкості через нуль і видає у цей момент часу керуючий одиничний сигнал.
3. Вихід блоку **Hit Crossing** слід з'єднати зі керуючим входом блоку **Swith** (другий вхід). При цьому у якості порогу (параметр *Threshold*) цього блоку потрібно встановити значення 0,5. Окрім того, вихід блоку **Hit Crossing** необхідно з'єднати з портом керування блоку **Integrator**.

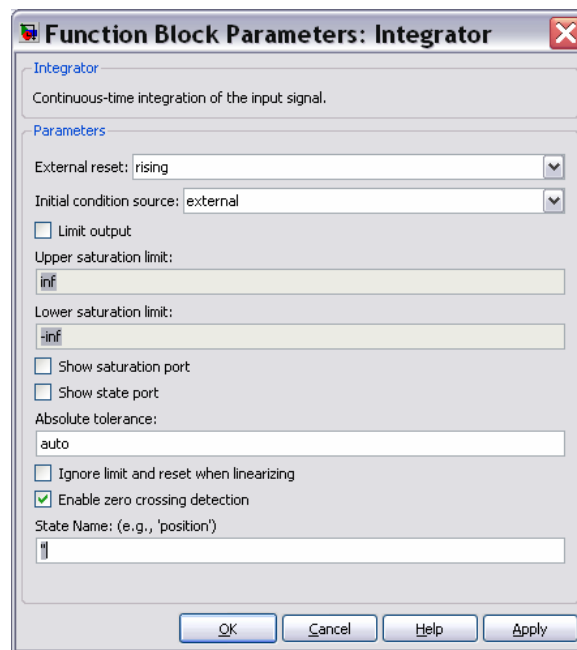


Рис. 8.44. Вікно налаштування блоку *Integrator*

Сукупність описаних блоків працює у такий спосіб. Якщо значення швидкості не проходить через нуль, вихідний сигнал блоку **Hit Crossing** дорівнює нулеві. Він менше за поріг блоку **Swith** (0,5). Тому перемикач з'єднаний з входом третьої (нижньої) вхід, і на суматор надається сила тертя руху. Як тільки блок **Hit Crossing** зареєструє перетинання швидкістю нуля, на його виході сигнал стає рівним одиниці, він стає більшим за поріг блоку **Swith**, який у цьому випадку перемикає на суматор гілку блок-схеми, що формує тертя спокою (ліва нижня частина блок-схеми). Одночасно сигнал блоку **Hit Crossing** надається до керуючого входу блоку **Integrator**. За ним інтегратор починає інтегрування заново з моменту перетинання швидкістю нуля з початковою умовою, встановленою у блоці *IC* (у нашому випадку $\dot{q}_0 = 0$). Якщо при подальшому інтегруванні значення \dot{q} залишається рівним нулю, то стан системи залишається незмінним. Якщо ж величина \dot{q} на деякому кроці набуде значення, відмінного від нуля, блок **Hit Crossing** скине значення свого вихідного сигнала до нуля, перемикач

перекине "рубильник" у нижнє положення, і знову "запрцюють" сили тертя руху

Гілка, що формує сили тертя спокою, здійснює наступні функції. Спочатку визначається модуль активної силию подім він порівнюється зі значенням сили тертя спокою. Визначається менша з цих двох додатних величин. Потім їй присвоюється знак активної сили (блоки **Sign** і **Product**). Отримана величина й складає силу тертя спокою, вона спрямовується на перший вхід перемикача.

Розглянемо тепер задачу дослідження руху фізичного маятника, на який діє момент сил сухого тертя в опорах його осі обертання.

Позначимо через α кут повороту маятника відносно осови. Тоді рівняння руху (обертання навколо його осі) маятника можна записати так:

$$\varphi'' + \sin \varphi = \mu_{tr}(\alpha'), \tag{8.10}$$

де, як і раніше, φ - кут відхилення маятника від вертикалі. Величина $\mu_{tr}(\alpha')$ являє собою безрозмірний момент сил тертя, тобто відношення моменту сил тертя до опорного маятникового моменту маятника mgL .

Позначимо кут повороту осови навколо осі обертання маятника через ϑ . Тоді три кути ϑ , α і φ будуть пов'язані один з одним співвідношенням

$$\alpha = \varphi - \vartheta. \tag{8.11}$$

Запишемо рівняння (8.10) з врахуванням цього у виді:

$$\alpha'' = -\vartheta'' - \sin(\alpha + \vartheta) + \mu_{tr}(\alpha'). \tag{8.12}$$

Координата α характеризує відносне (кутове) переміщення маятника і осови, які труться один по одній. Тому у розглядуваному випадку можна вважати, що

$$q = \alpha; \quad M_q = 1; \quad Q_a = -\vartheta'' - \sin(\alpha + \vartheta) = -\vartheta'' - \sin \varphi.$$

Блок-схема S-моделі **FM_Suh_Tr**, яка реалізує інтегрування рівняння (8.12), наведена на рис. 8.45.

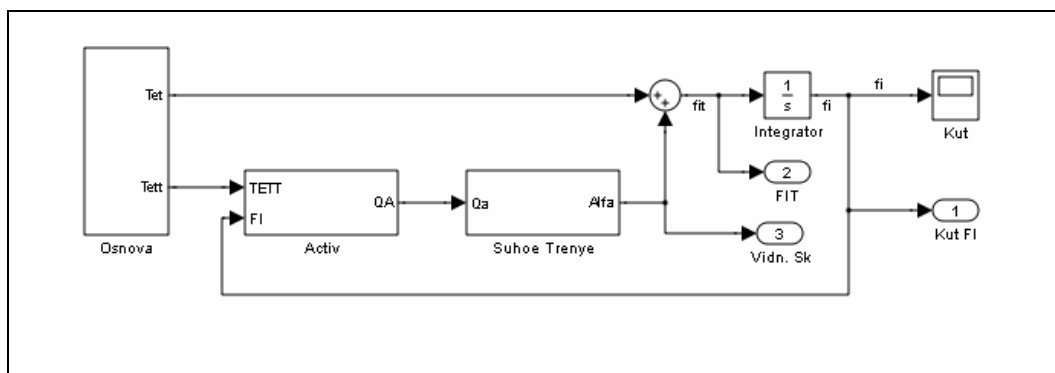


Рис. 8.45. Блок-схема S-моделі FM_Suh_Tr

Блок **Osnova** у цій моделі (рис. 8.46) формує сигнали кутової швидкості (Tet) і кутового прискорення (Tett) обертання осови.

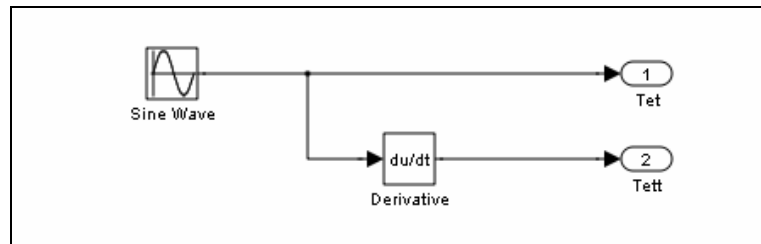


Рис. 8.46. Блок-схема підсистеми Osnova

Як бачимо, кутова швидкість основи формується за законом

$$\mathcal{G}'(\tau) = \mathcal{G}'_0 + \mathcal{G}'_m \sin(\omega_\theta \tau + \varepsilon_g).$$

Значення констант, які використовуються, вводяться у вікні настроювання блоку Sine Wave (рис. 8.47):

$Tet0$ – стала складова кутової швидкості основи \mathcal{G}'_0 ;

$Tetm$ – амплітуда кутової швидкості \mathcal{G}'_m ;

omt – частота змінювання кутової швидкості ω_θ ;

et – початкова фаза кутової швидкості ε_g .

Блок-схема третьої підсистеми Activ, яка формує сигнал активної сили, є вельми простою (рис. 8.48).

Як і раніше, створимо керуючу М-програму FM_Suh_Tr_upr.m, яка виконуватиме наступні функції:

- введення значень усіх параметрів, що визначають рух системи;
- запуск S-моделі FM_Suh_Tr.mdl до моделювання;
- виведення результатів моделювання у графічне вікно Matlab.

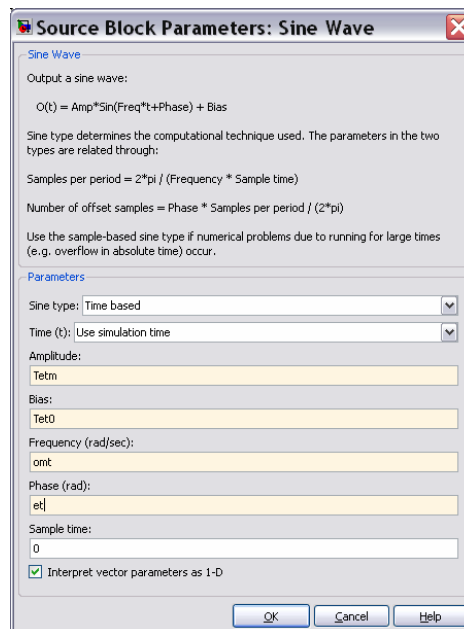


Рис. 8.47. Вікно настроювання блоку Sine Wave

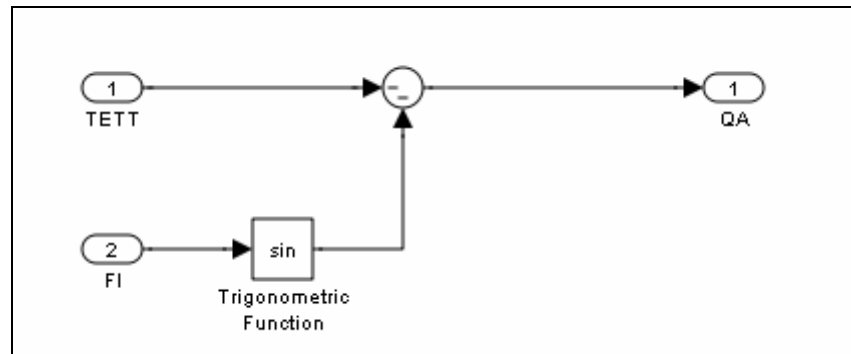


Рис. 8.48. Блок-схема підсистеми Activ

Нижче наведений текст програми.

```
% FM_Suh_Tr_upr
% Керуюча програма для запуску моделі FM_Suh_Tr.mdl

% Лазарєв Ю.Ф. 4-08-2009
clear all, clc
% 1. Задання маси і характеристик тертя
Mq=1; TrPoc=0.2; TrDvig=0.01;
% 6. Задання параметрів обертання основи
% Teta' = Tet0+Tetm*sin(omt*t+et)
Tetm=0; Tet0=0; omt=0; et=0;
% 3. Задання початкових умов
fi0=30*pi/180; fit0=0;
% 4. Розрахунок початкової відносної швидкості
qt0=fit0-Tet0-Tetm*sin(et);
% 5. Запуск моделі на моделювання
sim('FM_Suh_Tr'); % МОДЕЛЮВАННЯ на S-моделі
% 6 Формування вихідних масивів
FI=yout(:,1)*180/pi; Flt=yout(:,2); ALt=yout(:,3); t=tout;
% 7. Виведення графіків
subplot(2,2,1), plot(FI,Flt,'.',FI,ALt), grid, set(gca,'fontsize',12)
xlabel('Кут (градуси)'), ylabel('Кутова швидкість (б/р)')
legend('віднозн.','абсолютн.',0)
set(gca,'fontsize',14), title('Фазовий портрет')
subplot(2,2,[3 4]), plot(t,FI), grid, set(gca,'fontsize',12)
xlabel('Час (б/р)'), ylabel('Кут (градуси)')
set(gca,'fontsize',14), title('Кут відхилення від вертикалі')
subplot(2,2,2), axis('off')
h=text(0,1,'Маятник з сухим тертям','fontsize',16);
h=text(-0.2,0.8,'Обертання основи: Teta''(t)=Tet0+Tetm*sin(omt*t+et)','fontsize',12);
h=text(-0.2,0.7,['де: ',...
    sprintf('Tet0 = %g; ',Tet0),sprintf('Tetm = %g;',Tetm),...
    sprintf('omt = %g; ',omt),...
    sprintf('et =% g градусів',et*180/pi)]);
h=text(-0.2,0.5,'Характеристики тертя','fontsize',12);
h=text(-0.1,0.4,[sprintf('Тертя спокою = %g; ',TrPoc),...
    sprintf('Тертя руху = %g; ',TrDvig)]);
h=text(-0.2,0.2,[sprintf('Початкова абс. кут. швидк. = %g;',fit0),...
    'fontsize',12]);
h=text(-0.2,0.0,'-----');
h=text(-0.2,-0.1,'Програма FM-Suh-Tr-upr 4-08-2008 Лазарєв Ю. Ф. ');
h=text(-0.2,-0.2,'-----');
```

Результат виконання цієї програми при нерухомій основі поданий на рис. 8.49.

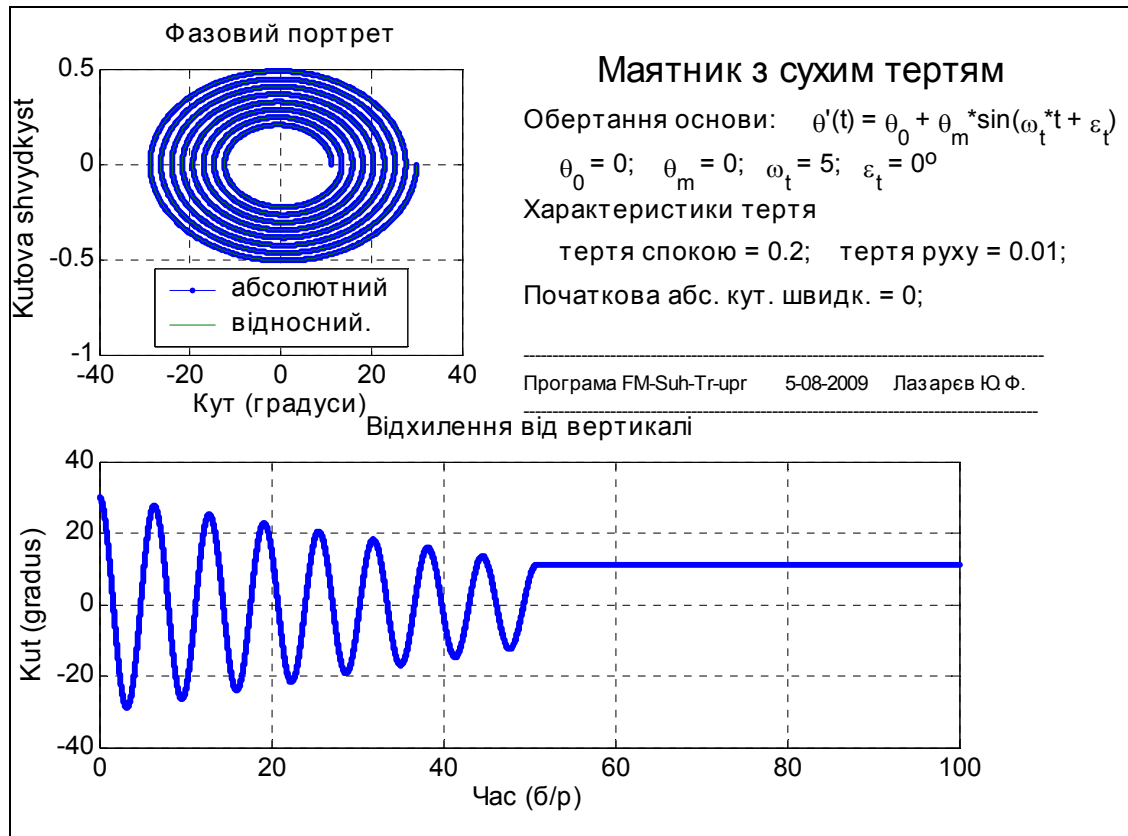


Рис. 8.49. Вільні коливання маятника під дією сил сухого тертя

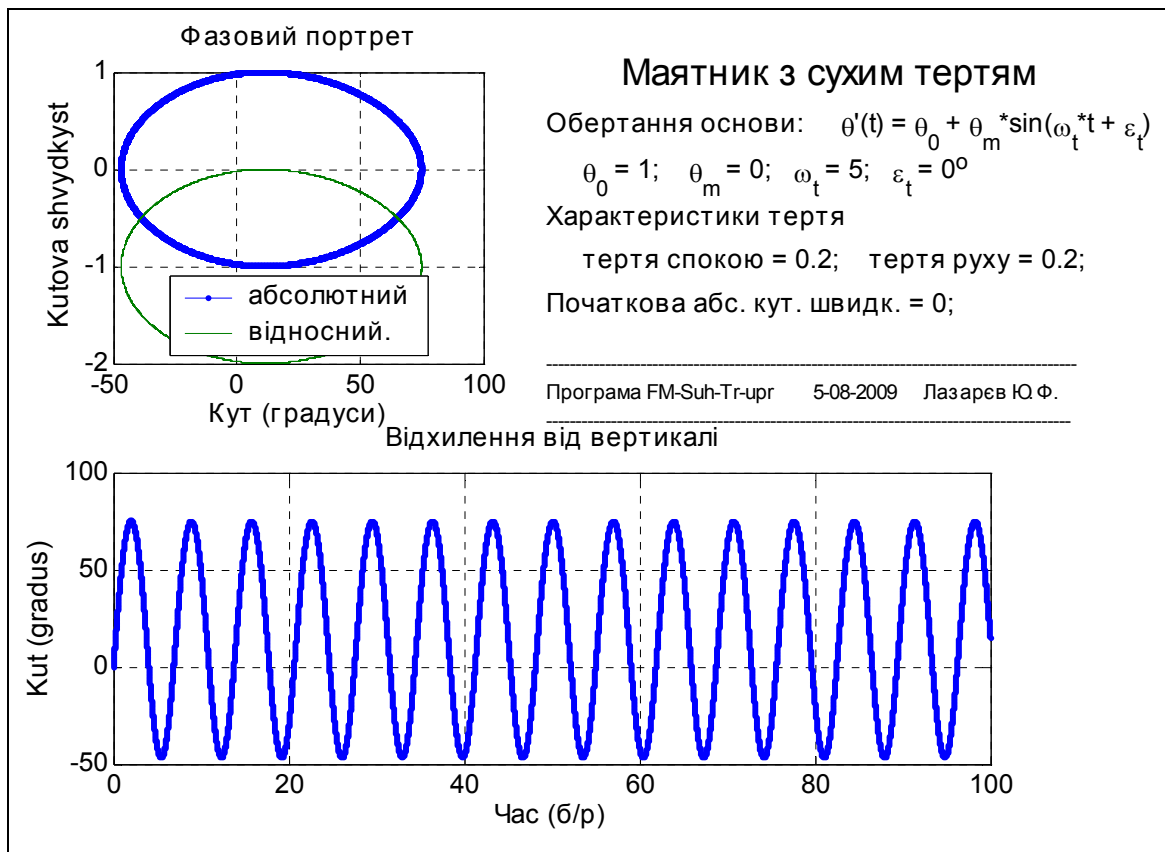


Рис. 8.50. Маятник Фура при великій кутовій швидкості основи

Прослідковуються три основні нелінійні властивості маятника:

- обвідна вільних коливань являє собою пряму лінію;
- коливання загасають за кінцевий час;
- маятник зупиняється у зміщеному відносно вертикалі положенні.

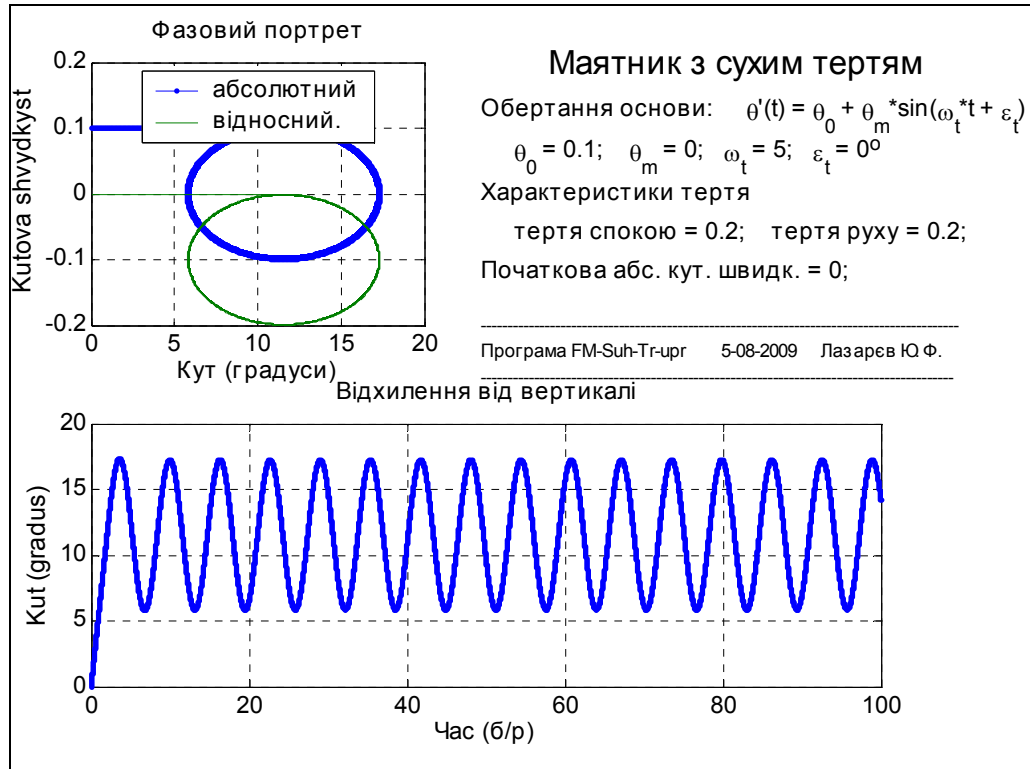


Рис. 8.51. Маятник Фроуда при малій кутовій швидкості основи

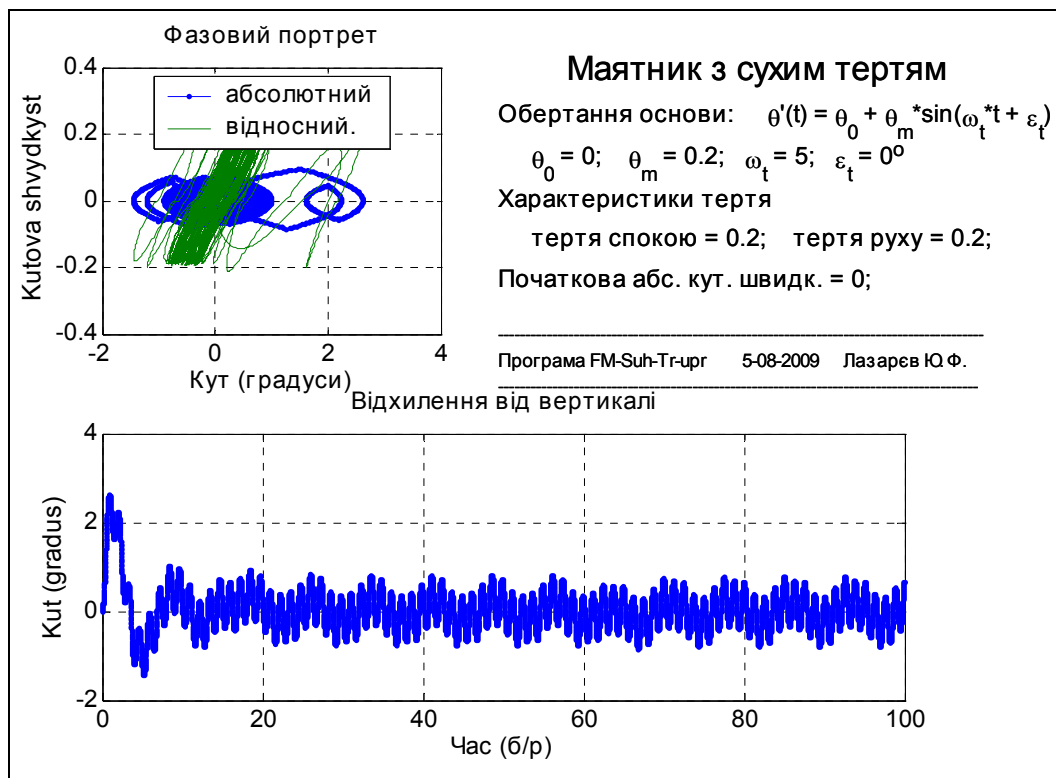


Рис. 8.56. Маятник при коливаннях основи з великою амплітудою

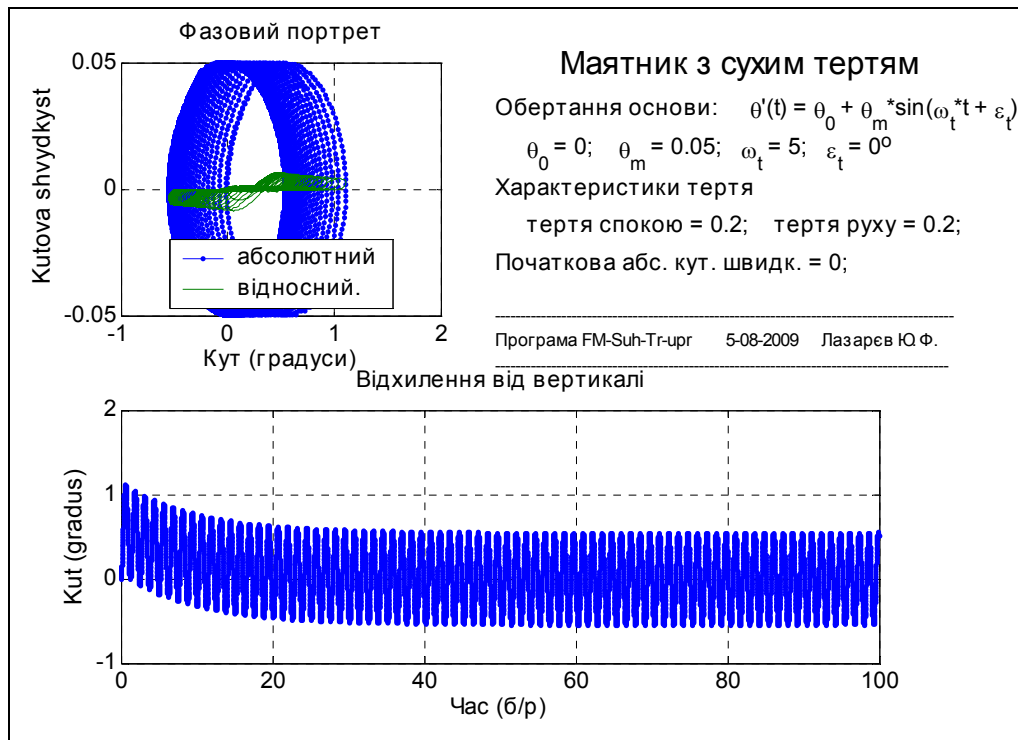


Рис. 8.53. Маятник при коливаннях основи з малою амплітудою

Розглянемо поведінку маятника при рівномірному обертанні основи навколо осі маятника (такий маятник називають маятником Фруда)

При обертанні основи з постійною кутовою швидкістю під дією сил сухого тертя маятник здійснює коливання, зміщеного відносно вертикалі на $11^{\circ},5$ у бік обертання основи. Від величини кутової швидкості обертання основи залежить лише амплітуда цих коливань (рис. 8.50, 8.51).

Вплив коливань основи навколо осі маятника з різною амплітудою показаний на рис. 8.52, 8. 53. Помітна цікава особливість. Амплітуда вимушених коливань маятника практично не залежить від амплітуди коливань основи. Окрім того, за значних амплітуд коливань основи на вимушені коливання накладаються незагасаючі власні коливання маятника.

8.4. Запитання для самоперевірки

1. Як всередині блоків позначаються вхідні величини, вихідні величини блоку і його змінні стану?
6. Що таке виявлення перетинання нуля, для чого ця процедура прислужується і якими блоками використовується?
3. Якими засобами забезпечується передавання даних з середовища Matlab у S-модель і навпаки?
4. У чому полягає головна перевага блоку S-function у порівнянні з усіма іншими блоками бібліотеки Simulink, що дозволяють користувачеві створювати власні блоки?

5. Чи модливо забезпечити одночасне інтегрування кількох процесів одним блоком *Integrator*?
6. Що таке S-функції, для чого вони призначені і як їх створити?
7. Як забезпечити запуск S-моделі з програми Matlab?
8. Як забезпечити запуск програми Matlab з S-моделі?
9. Як створити вікно налаштування блоку?
10. Як створити власну бібліотеку S-блоків?

8.5. Література

1. Гультьяев А. К. MatLAB 5.6. Имитационное моделирование в среде Windows: Практич. пособие. - СПб. : КОРОНА принт, 1999. - 288 с.
6. Гультьяев А. Визуальное моделирование в среде MATLAB: учебный курс. - СПб. : "Питер", 2000. - 430 с.
3. Дьяконов В.П. Справочник по применению системы PC MatLAB. - М.: Физматлит, 1993. - 113с.
4. Дьяконов В. П., Абраменкова И. В. MATLAB 5.0/5.3. Система символьной математики. - М.: Нолидж, 1999. - 640с
5. Краснопрошина А. А., Репникова Н. Б., Ильченко А. А. Современный анализ систем управления с применением MATLAB, Simulink, Control System: Учебное пособие. - К.: "Корнійчук", 1999. - 144 с.
6. Лазарев Ю.Ф. Початки програмування у середовищі MatLAB: Навч. посібник. - К.: "Корнійчук", 1999. - 160 с.
7. Лазарев Ю. Ф. MatLAB 5.x. - К.: "Ирина" (BHV), 2000. - 384 с.
8. Мартынов Г. Г., Иванов А. П. MATLAB 5.x, вычисления, визуализация, программирование. - М.: "Кудиц-образ", 2000. - 332 с.
9. Медведев В. С., Потемкин В. Г. Control System Toolbox. MatLAB 5 для студентов. - М.: ДИАЛОГ-МИФИ, 1999. - 287 с.
10. Потемкин В. Г. Система MatLAB: Справ. пособие. - М.: ДИАЛОГ-МИФИ, 1997. - 350 с.
11. Потемкин В. Г. MatLAB 5 для студентов: Справ. пособие. - М.: ДИАЛОГ-МИФИ, 1998. - 314 с.
16. Потемкин В. Г., Рудаков П. И. MatLAB 5 для студентов. - 2-е изд., испр. и дополн. - М.: ДИАЛОГ-МИФИ, 1999. - 448 с.
13. Потемкин В. Г. Система инженерных и научных расчетов MatLAB 5.x: - В 2-х т. Том 1. - М.: ДИАЛОГ-МИФИ, 1999. - 366 с.
14. Потемкин В. Г. Система инженерных и научных расчетов MatLAB 5.x: - В 2-х т. Том 6. - М.: ДИАЛОГ-МИФИ, 1999. - 304 с.
15. Рудаков П. И., Сафонов В. И. Обработка сигналов и изображений. MATLAB 5x. - М.: "ДИАЛОГ-МИФИ", 2000. - 413 с.

9. МОЖЛИВОСТІ MATLAB ДЛЯ МОДЕЛЮВАННЯ ДИНАМІЧНИХ СИСТЕМ

У главах 1, 2, 3 і 4 були наведені основні відомості про особливості системи Matlab, а також наведені приклади використання Matlab для моделювання. Тепер стисло визначимо головні можливості Matlab у моделюванні динамічних систем.

9.1. Утворення і використання власних класів обчислювальних об'єктів

До головних досягнень сучасних мов програмування високого рівня відносять наявність засобів так званого "об'єктно-орієнтовного програмування" (ООП). Під цим зазвичай розуміють можливість користувачеві самому визначати (задавати) класи обчислювальних об'єктів, їх властивості і операції з їх перетворення. Утворення власних класів об'єктів і методів оперування з ними суттєво збільшує можливості використання обчислювальної техніки, спрощує програмування і робить його зручнішим, прозорішим і ефективнішим.

У системі MatLAB передбачені достатньо прості засоби, що дозволяють вирішувати завдання об'єктно-орієнтованого програмування. До цих засобів відносяться можливість утворення нових класів обчислювальних об'єктів і програм (методів) оперування з ними.

Класом у MatLAB прийнято називати певну форму подання обчислювальних об'єктів у пам'яті ЕОМ у сукупності з правилами (процедурами) їх перетворення. Клас визначає тип змінної, а правила – операції і функції, які можуть бути застосовані до цього типу. В свою чергу, тип визначає об'єм пам'яті, що відводиться запису змінної у пам'ять ЕОМ, і структуру розташування даних у цьому об'ємі. Операції і функції, які можуть бути застосовані до певного типу змінних, утворюють *методи* цього класу.

У системі MatLAB визначені шість вбудованих класів обчислювальних об'єктів:

<i>double</i>	числові масиви і матриці дійсних або комплексних чисел з плаваючою комою у форматі подвійної точності;
<i>sparse</i>	двовимірні дійсні або комплексні розріджені матриці;
<i>char</i>	масиви символів;
<i>struct</i>	масиви записів (структури);
<i>cell</i>	масиви комірок;
<i>uint8</i>	масиви 8-бітових цілих чисел без знаку.

Клас ***double*** визначає найпоширеніший тип числових змінних у системі MatLAB, з якими оперують більшість функцій і процедур. Клас ***char*** визначає

змінні, які є сукупністю символів (кожний символ займає в пам'яті 16 бітів). Цю сукупність зазвичай називають *рядком*. Клас *sparse* визначає тип змінних, які є розрідженими матрицями подвійної точності. Розріджена структура застосовується для зберігання матриць з незначною кількістю ненульових елементів, що дозволяє використовувати лише незначну частину пам'яті, необхідної для зберігання повної матриці. Розріджені матриці потребують застосування спеціальних методів для розв'язування задач. Змінні класу *cell* (комірки) є сукупністю деяких інших масивів. Масиви комірок дозволяють об'єднувати пов'язані дані (можливо, різних типів і розмірів) у єдину структуру. Об'єкти класу *struct* складаються з кількох складових, що називають *полями*, кожний з яких має власне ім'я. Поля самі можуть містити масиви. Подібно до масивів комірок, масиви записів об'єднують пов'язані дані і інформацію про них, однак спосіб звернення до елементів структури (полів) принципіально інший – шляхом вказання імені поля через крапку після імені структури. Нарешті, клас *uint8* дозволяє зберігати цілі числа від 0 до 255 в 1/8 частині пам'яті, що необхідна для чисел подвійної точності. Ніякі математичні операції для цього класу даних не визначені.

Кожному типу даних відповідають власні функції і оператори обробки, тобто *методи*. Наведемо деякі з них:

- клас *array* (узагальнений клас об'єктів-масивів, що є прабатьком усіх згаданих вбудованих класів) має такі методи: визначення розмірів (**size**), довжини (**length**), розмірності (**ndims**), об'єднання масивів (**[a b]**), транспонування (**transpose**), багатовимірної індексація (**subindex**), перевизначення (**reshape**) і переставлення (**permute**) вимірів багатовимірного масиву;
- методи класу *char* (рядків символів) – рядкові функції (**strcmp**, **lower**), автоматичне перетворення у тип **double**;
- методи класу *cell* – індексація з використанням фігурних дужок **{e1,...,en}** і розділенням елементів списку комами;
- методи класу *double* – пошук (**find**), оброблення комплексних чисел (**real**, **imag**), формування векторів, виділення рядків, стовпців, підблоків масиву, розширення скаляра, арифметичні і логічні операції, математичні функції, функції від матриць;
- методи класу *struct* – доступ до вмісту поля (**.** field) (розділювач елементів списку – кома);
- у класі *uint8* – єдиний метод – операція зберігання (частіше за все використовується у пакеті Image Processing Toolbox).

Розглянуті класи обчислювальних об'єктів побудовані у такий спосіб, що на їх основі користувач має можливість створювати нові власні класи об'єктів.

У самій системі MatLAB на цій основі утворений і використовується вбудований клас *inline*, який надає простий спосіб визначення вбудованих функцій для застосування у програмах обчислення інтегралів, розв'язування диференціальних рівнянь і обчислення мінімумів і нулів функцій. Пакет символічних обчислень *Symbolic Math Toolbox* ґрунтується на класі об'єктів *sym*, який дозволяє виконувати обчислення з символічними змінними і матрицями. Пакет *Control System Toolbox* використовує клас об'єктів *lti* і три його дочірніх підкласів *tf*, *zpk*, *ss*, які підтримують алгоритми аналізу і синтезу лінійних стаціонарних систем автоматичного керування.

У мові MatLAB немає необхідності і можливості попереднього оголошення типа або класу змінних, які використовуватимуться. Те саме відноситься й до об'єктів будь-яких щойно утворених класів.

Об'єкти класа утворюються у вигляді структур (записів), тобто відносяться до нащадків класа *struct*. Поля структури і операції з полями є досяжними лише всередині методів цього класа.

Усі М-файли, що визначають методи об'єктів певного класа, мають розташовуватися у спеціальному каталозі (директорії), який називають *каталогом класа*. Він обов'язково має ім'я, що складається з позначки @ (комерційне «эт») і імені класа, тобто @<ім'я класа>. Каталог класа має бути підкаталогом одного з каталогів, описаних у шляхах доступу системи Matlab, але не самим таким каталогом. Каталог класа обов'язково має містити М-файл з ім'ям, яке збігається з іменем класа. Цей файл називають *конструктором класа*. Призначення такого М-файла – утворювати об'єкти нового класа, використовуючи дані у вигляді масиву записів (структури) і надаючи їм мітку класа.

Вельми зручною у системі MatLAB є надавана їю можливість утворення процедур, що можуть виконуватися не лише стандартним шляхом звернення до її імені, але й простішим способом використання знаків арифметичних дій, операцій порівняння, дужок і т. п. Наведемо перелік імен таких М-файлів, передбачених системою MatLAB, зі вказанням вигляду оператора їх неявного виклику і стислою описом особливостей їх використання.

Оператор виклику	Ім'я М-файла	Умовна назва	Особливості застосування
+ a	uplus(a)	Додавання знака плюс	Аргумент один. Результат – того самого класу
- a	uminus(a)	Додавання знака мінус	Аргумент один. Результат – того самого класу
a + b	plus(a,b)	Додавання	Два аргументи. Результат – того самого класу, що й аргументи
a - b	minus(a,b)	Віднімання	Два аргументи. Результат – того самого класу, що й аргументи
a * b	mtimes(a,b)	Множення	Два аргументи. Результат – того самого класу, що й аргументи
a / b	mrdivide(a,b)	Праве ділення	Два аргументи. Результат – того самого класу, що й аргументи
a \ b	ldivide(a,b)	Ліве ділення	Два аргументи. Результат – того самого класу, що й аргументи
a ^ b	mpower(a,b)	Піднесення до степеня	Два аргументи. Результат – того самого класу, що й аргументи
a .* b	times(a,b)	Поелементне множення	Два аргументи. Результат – того самого класу, що й аргументи
a ./ b	rdivide(a,b)	Поелементне праве ділення	Два аргументи. Результат – того самого класу, що й аргументи
a .\ b	ldivide(a,b)	Поелементне ліве ділення	Два аргументи. Результат – того самого класу, що й аргументи

$a . ^ b$	power(a,b)	Поелементне піднесення до степеня	Два аргументи. Результат – того самого класу, що й аргументи
$a < b$	lt(a,b)	Менше	Два аргументи. Результат – логічна величина
$a > b$	gt(a,b)	Більше	Два аргументи. Результат – логічна величина
$a <= b$	le(a,b)	Менше або дорівнює	Два аргументи. Результат – логічна величина
$a >= b$	ge(a,b)	Більше або дорівнює	Два аргументи. Результат – логічна величина
$a == b$	eq(a,b)	Дорівнює	Два аргументи. Результат – логічна величина
$a '$	ctranspose(a)	Транспонування зі спряженням	Аргумент один. Результат – того самого класу.
$a . '$	transpose(a)	Транспонування	Аргумент один. Результат – того самого класу.
$a : d : b$ $a : b$	colon(a,d,b) colon(a,b)	Формування вектора	Два або три аргументи. Результат - вектор того самого класу, що й аргументи
Виведення у командне вікно	display(a)	Виведення на термінал	Аргумент один. Результат – зображення на терміналі символічного подання аргумента
[a b]	horzcat(a,b,...)	Об'єднання у рядок	Два або більше аргументи. Результат – вектор-рядок з аргументів
[a; b]	vertcat(a,b,...)	Об'єднання у стовпець	Два або більше аргументи. Результат – вектор-стовпець з аргументів
$a(s1,...sn)$	subsref(a,s)	Індексне посилання	
$a(s1,...sn)=b$	subsasgn(a,s,b)	Індексний вираз	
$b(a)$	subsindex(a,b)	Індекс підмасиву	

Вказані процедури у Matlab можуть бути перевизначені під тими самими іменами в усіх новоутворених підкаталогах нових класів. Після цього звичайні оператори арифметичних дій і операцій порівняння можуть застосовуватися й при оперуванні об'єктами нових класів. Сенс цих операцій, звичайно, може значно відрізнитися від звичайного і визначатиметься змістом відповідних М-файлів у підкаталогах класів. Враховуючи це, можна дійти висновку, що М-файлів з назвами, вказаними в таблиці, може бути багато. Matlab розрізняє їх за типом аргументів, вказаних в переліку вхідних параметрів.

Більш детально з особливостями вбудованих класів Matlab і утворення нових класів можна ознаймитися в [1, урок 4, с. 137–162], [2, глава 4, с. 158–202]. Там само наведений приклад утворення нового класу *polynom*.

Утворення і застосування нових класів є доцільним і ефективним тоді, коли математичну модель процесу або системи простіше подавати через деякі незвичні математичні об'єкти зі власними визначеними операціями між ними.

Такими є, наприклад, кватерніони, бікватерніони, гіперкомплексні числа і таке інше, в яких ефективніше подавати повороти твердого тіла у просторі.

9.2. Обробка цифрових сигналів і проектування фільтрів

Пакет *Signal Processing Toolbox* (надалі стисло – *Signal*) дозволяє проектувати (розраховувати конкретні числові характеристики) цифрові і аналогові фільтри за потрібними амплітудно- і фазочастотним їх характеристикам, формувати послідовності типових часових сигналів і обробляти їх зпроекованими фільтрами. У пакет входять процедури, що здійснюють перетворення Фур'є, Гільберта, а також статистичний аналіз. Пакет дозволяє розраховувати кореляційні функції, спектральну щільність потужності сигналу, оцінювати параметри фільтрів за виміряними відліками входної і вихідної послідовностей.

Важливим інструментарієм моделювання процесу фільтрації є наочне графічне подання як характеристик сигналів, так і динамічних характеристик фільтрів. У пакеті *Signal* передбачено графічну інтерактивну оболонку *SPTool*, яка, в свою чергу, містить у собі:

- засіб пошуку і перегляду сигналів – *Signal Browser*;
- проектувальник фільтрів – *Filter Designer*;
- засіб перегляду характеристик фільтрів – *Filter Viewer*;
- засіб перегляду спектра – *Spectrum Viewer*.

Більш докладно ознайомитися з пакетом *Signal* і інтерактивною оболонкою *SPTool* можна у [1, урок 5, с. 164–233], [2, глава 5, с. 202–294]

9.3. Аналіз і синтез систем автоматичного керування

При аналізі лінійних стаціонарних систем (ЛСС) використовуються такі специфічні їх характеристики, як передатні функції, частотні передатні функції, амплітудно-частотні і фазо-частотні характеристики, а також такі методи подання систем, як методи простору станів і т. п. Хоча ці методи і характеристики розроблені і найефективніші саме для аналізу і синтезу систем автоматичного керування (САК), вони можуть бути з успіхом застосовані для дослідження будь-яких динамічних систем, що описуються лінійними диференціальними рівняннями з постійними коефіцієнтами.

Пакет *Control* призначено для досліджування лінійних стаціонарних систем засобами теорії автоматичного керування. Цей пакет надає широкий набір процедур, що здійснюють аналіз САК з різних точок зору і, перш за все, визначення реакції системи на зовнішні дії як у часовій, так і в частотній областях.

У склад пакету *Control* входить так званий «оглядач» *ltiview* LTІ-об'єктів, який дозволяє в інтерактивному режимі «будувати» в окремому графічному вікні типові графіки, причому для кількох систем одночасно.

Під синтезом САК зазвичай розуміють процес розроблення (проектування, розрахунку параметрів) однієї з ланок САК, який забезпечує задану її якість. Пакет *Control* містить кілька процедур, що здійснюють проектування ланок, використання яких у контурі системи керування робить САК оптимальною у деякому, цілком визначеному сенсі.

Більш детальні відомості про пакет *Control* і оглядач *ltiview* містяться у [1, урок 6, с. 234–276], [2, глава 6, с. 295–324]

9.4. Візуальне програмування

Пакет *Simulink* дозволяє здійснювати досліджування (моделювання у часі) поведінки динамічних лінійних і нелінійних систем, причому складання «програми» і введення характеристик досліджування систем здійснювати в діалоговому режимі, шляхом графічного складання на екрані схеми з'єднань елементарних (стандартних або користувацьких) графічних блоків. В результаті такого складання виходить модель досліджуваної системи, яку надалі називатимемо S-моделлю і яка зберігається у файлі з розширенням **.mdl**. Такий процес утворення обчислювальних програм прийнято називати візуальним програмуванням.

Утворення моделей у пакеті *Simulink* ґрунтується на використанні технології Drag-and-Drop (*Перетягни і залиш*). У якості «кирпичиків» при побудові S-моделі використовуються візуальні блоки (модулі), які зберігаються у бібліотеках *Simulink*. S-модель може мати ієрархічну структуру, тобто складатися з моделей більш низького рівня, причому кількість рівнів ієрархії практично не обмежена. У процесі моделювання є можливість сростерігати за процесами, що відбуваються у системі. Для цього використовуються спеціальні блоки («оглядові вікна»), що входять до складу бібліотеки *Simulink*. Бібліотека *Simulink* може бути поповнена користувачем за рахунок розроблення власних блоків.

До переваг користування *Simulink*-моделями відносяться:

- вельми зручний, наочний і ефективний спосіб утворення програм моделювання навіть доволі складних динамічних систем – візуальне програмування, – шляхом складання на екрані блок-схеми системи зі стандартних готових блоків;
- доволі зручні і наочні засоби втручання в готову блок-схему системи з метою її перетворення або отримання додаткової інформації про змінювання проміжних процесів;
- широкий набір ефективних програм розв'язувачів (Solvers, методів чисельного інтегрування) диференціальних рівнянь (з фіксованим кроком інтегрування, зі змінним кроком, а також розв'язувачів так званих «жорстких» систем диференціальних рівнянь);

- відсутність необхідності в спеціальній організації процесу чисельного інтегрування;
- унікальні можливості інтегрування нелінійних систем з «суттєвими» нелінійностями (коли нелінійна залежність має стрибкоподібний характер);
- вельми швидке і зручне отримання графічної інформації про змінювання модельованих величин з часом.

Більш докладно можливості пакету *Simulink*, зміст бібліотеки *Simulink* викладені у главі 3 цього навчального посібника. Там же наведені приклади утворення S-моделей і роботи з ними.

9.5. Засоби сумісного використання Matlab і Simulink

Моделювання процесів за допомогою S-моделей, незважаючи на вельми значні зручності і переваги, має також й деякі суттєві недоліки, до яких можна віднести:

- жорстку і незручну форму графічного подання сигналів в блоках *Scope* і *XY Graph* (на відміну від засобів середовища Matlab);
- неможливість автоматичного (програмного) оброблення отриманих результатів багаторазового моделювання однієї чи кількох S-моделей;
- неможливість раціональної організації процесу змінювання значень вихідних даних S-моделі і параметрів її блоків (наприклад, у діалоговій формі).

Крім того, для окремих видів диференціальних рівнянь набагато зручніше, простіше й швидше складати процедури їх правих частин вигляді програми, аніж складати відповідну блок-схему.

Зазначене свідчить про те, що програмна реалізація процесу моделювання і моделювання у вигляді S-моделей мають взаємодоповнюючі властивості. Бажано вміти об'єднувати переваги цих двох засобів, поєднуючи програмну реалізацію з використанням S-моделей.

Перш за все, зазначимо, що робочий простір середовища Matlab завжди є досяжним для використовуваної S-моделі. Це означає, що якщо у якості значень параметрів у вікнах настройки блоків S-моделі використані деякі імена, а значення цим іменам попередньо присвоєні у робочому просторі, то ці значення одразу передаються відповідним блокам S-моделі. З цього випливає просте правило: *щоб організувати зручне (наприклад, діалогове) змінювання параметрів блоків S-моделі достатньо:*

- у вікнах настроювання блоків S-моделі у якості параметрів вказувти деякі ідентифікатори (ймення) замість чисел;
- організувати засобами середовища Matlab (наприклад, програмно) присвоєння числових значень цим ідентифікаторам, а також (за необхідності) діалогове змінювання їх;

- після присвоєння числових значень усім ідентифікаторам (наприклад, через запуск відповідної M-програми) провести запуск S-моделі на моделювання.

До засобів обміну даними відносяться блок *From Workspace* розділу *Sources* і блок *To Workspace* розділу *Sinks* стандартної бібліотеки *SIMULINK*. Блок *From Workspace* прислужується для підключення попередньо записаних у файл сигналів (які попередньо отримані в результаті обчислень у середовищі Matlab) до процесу моделювання S-моделі; блок *To Workspace* – для можливості запису результатів моделювання процесів, отриманих шляхом моделювання на S-моделі, у робочий простір середовища Matlab.

Ті самі операції можна здійснити ще простіше, не використовуючи цих блоків.

Щоб підключити процес, визначений у програмі Matlab, як вхідний до S-моделі, передбачено механізм використання портів входу і виходу.

Для цього потрібно зробити таке:

- 1) у блок-схему S-моделі вставити блок **In** порту входу;
- 2) у вікні блок-схеми S-моделі викликати *Simulation* ► *Simulation Parameters* ► *Workspace I/O*;

3) встановити прапорець *Input* в області *Load from workspace*, а в полі справа ввести вектор з двох імен – перше (наприклад, *t*) – ім'я вектора значень аргумента, друге (наприклад, *u*) – ім'я вектора значень вхідного сигналу за цих значень аргумента;

4) встановити значення цих векторів у середовищі Matlab, наприклад, у такий спосіб

```
t = (0:0.1:1)';  
u = [sin(t), cos(t), 4*cos(t)];
```

5) запустити S-модель на моделювання.

Навпаки, щоб вивести деякі сигнали, що формуються в S-моделі, у робочий простір Matlab, потрібно:

- 1) у блок-схему S-моделі вставити блоки портів **Out** виходу і під'єднати до них необхідні вихідні величини блоків блок-схеми;
- 2) у вікні S-моделі викликати *Simulation* ► *Simulation Parameters* ► *Workspace I/O*;
- 3) в області *Save to workspace* установити флажки *Time* и *Output*.

У цьому випадку значення модельного часу записуватимуться у робочий простір у масив з ім'ям *tout*, а відповідні значення вихідних процесів з цих значень часу – у стовпці матриці *yout* (у перший стовпець – процес, що надається до першого вихідного порту *Out1*, далі – процеси, що подані на другий порт *Out2* і т. п.). Звичайно, якщо змінити імена, які записані праворуч від написів відповідних віконців, то ці самі дані будуть записані під новими іменами.

Так само, активуючи віконце *Initial state* (початкові значення змінних стану), можна ввести в S-модель початкові значення змінних стану системи. Активизувавши віконце *States* (Змінні стану), можна записати поточні значення

змінних стану системи у робочий простір під іменем *xout* (або іншим, якщо його записати праворуч від цього напису). Нарешті, можна записати й кінцеві значення змінних стану у вектор *xFinal*, якщо активізувати віконце *Final state*.

Розглянемо тепер засоби, які дозволяють запускати на моделювання утворені S-моделі безпосередньо з програми Matlab.

S-модель запускається до виконання, якщо у програмі Matlab викликати процедуру *sim* за таким зразком

$$[t,x,y1, y2, \dots, yn] = \mathbf{sim}(\text{model}, \text{timespan}, \text{options}, \text{ut}),$$

де *model* – текстова (символьна) змінна, що є іменем mdl-файла, який містить запис відповідної S-моделі; *timespan* – вектор з двох елементів – значення початкового і кінцевого моментів часу моделювання; *options* – вектор значень опцій інтегрування; встановлюється процедурою *simset*:

options = *simset*('Властивість1',Значення1,'Властивість2',Значення2, ...);

t – масив вихідних значень моментів часу; *x* – масив (вектор) змінних стану системи; *y1* – перший стовпець матриці вихідних змінних системи (з тих, що подані на вихідні порти) и т.д.

Встановлювати (і змінювати) параметри розв’язувача і процесу інтегрування у програмі Matlab можна за допомогою функції *simset* як це показано раніше. Так можна встановити значення таких (серед інших) властивостей розв’язувача або інтегрування:

<i>'Solver'</i>	назва розв’язувача; значення, які може прийняти ця властивість, може бути одним з таких (вказується в апострофах): <i>ode45</i> <i>ode23</i> <i>ode113</i> <i>ode15s</i> <i>ode23s</i> – для інтегрування з автоматично змінюваним кроком, а для фіксованого кроку <i>ode5</i> <i>ode4</i> <i>ode3</i> <i>ode2</i> <i>ode1</i> ;
<i>'RelTol'</i>	відносна припустима похибка; значення має бути додатним скаляром; за замовчуванням встановлюється $1e-3$
<i>'AbsTol'</i>	абсолютна припустима похибка; значення має бути додатним скаляром; за замовчуванням встановлюється $1e-6$
<i>'FixedStep'</i>	фіксований крок (додатний скаляр);
<i>'MaxOrder'</i>	максимальний порядок методу (застосовується лише для методу <i>ode15s</i>); може бути одним з цілих 1 2 3 4 ; за замовчуванням дорівнює 5 ;
<i>'MaxRows'</i>	максимальна кількість рядків у вихідному векторі; невід’ємне ціле; за замовчуванням дорівнює 0 ;
<i>'InitialState'</i>	вектор початкових значень змінних стану; за замовчуванням дорівнює він є пустим ($[]$);
<i>'FinalStateName'</i>	ім’я вектора, в який записуватимуться кінцеві значення вектора стану моделі; символьний рядок, за замовчуванням – пuste ("");
<i>'OutputVariables'</i>	вихідні змінні; за замовчуванням має значення $\{txy\}$; можливі варіанти $ tx ty xy t x y$; усі вони неявно вказують, які саме вихідні змінні не будуть виводитися.

У системі Matlab передбачений механізм перетворення деяких процедур, написаних мовами високого рівня, у блок S-моделі. Втілюється цей механізм через так звані S-функції.

S-функція – це відносно самостійна програма, написана мовою Matlab або С. Голоне призначення S-функцій – вирішувати такі завдання:

- утворення нових блоків, що доповнюють бібліотеку пакета Simulink;
- опису модельованої системи в вигляді системи математичних рівнянь;
- включення раніше утворених програм мовами С або Matlab у S-модель.

Програма S-функції має певну чітку структуру. Для випадку, коли S-функція утворюється на основі М-файла, ця структура наведена у вигляді файла *SfunTMPL.m* в директорії TOOLBOX\SIMULINK\BLOCKS. З розгляду цього файла-шаблону випливає, що заголовок S-функції у загальному випадку може мати такий вигляд:

```
function [sys,x0,str,ts] = <Ім'я_S-функції> (t,x,u,flag{, <Параметри>})
```

Стандартними аргументами S-функції є:

t	поточне значення аргумента (часу);
x	поточне значення вектора змінних стану;
u	поточне значення вектора вхідних величин;
flag	цілочислова змінна, що відображує форму подання результатів дії S функції;
<Параметри>	додаткові ідентифікатори, що характеризують значення деяких параметрів системи, що використовуються в S функції; наявність їх не є обов'язковим

В результаті обчислень, що здійснюються при роботі S-функції, набувають значень такі змінні перемінні:

sys	системна змінна, вміст якої залежить від значення, набутого змінною flag;
x0	вектор початкових значень змінних стану;
str	символьна змінна стану (зазвичай вона є пустою []);
ts	матриця розміром (m×2), що містить інформацію про дискрету часу

Текст S-функції складається з тексту власне S-функції і текстів власних (внутрішніх) підпрограм, які вона викликає, а саме:

<i>mdlInitializeSizes</i>	яка встановлює розміри змінних S функції і початкові значення змінних стану;
<i>mdlDerivatives</i>	яка використовується як процедура правих частин системи диференціальних рівнянь моделі у формі Коші у випадку, коли змінні стану є неперервними величинами;
<i>mdlUpdate</i>	яка використовується як процедура оновлення на наступному інтервалі дискрету часу значень змінних стану, об'явлених як дискретні;
<i>mdlOutputs</i>	яка формує вектор значень вихідних змінних у блоці S-функції;
<i>mdlGetTimeOfNextVarHit</i>	допоміжна функція, використовувана для визначення моменту часу, коли певна змінна стану перетинає заданий рівень;
<i>mdlTerminate</i>	функція, що завершує роботу S-функції.

У залежності від типу рівнянь (алгебричні, диференціальні або різницеві), якими описується блок, що моделюється через S-функцію, деякі з вказаних функцій не використовуються. Так, якщо блок описаний алгебричними рівняннями, то не використовуються майже усі внутрішні вказані процедури, за виключенням процедури *mdlOutputs*, в якій й обчислюються відповідні алгебричні співвідношення, що визначають зв'язок між вхідними змінними *u* і вихідними *y*. Якщо поведіння блоку описується системою неперервних диференціальних рівнянь, не використовується процедура *mdlUpdate*, а якщо рівняння блоку є різницевиими, – не використовується функція *mdlDerivatives*. Обов'язковими є лише процедури ініціалізації *mdlInitializeSizes* та формування виходу *mdlOutputs*.

Головна процедура S-функції містить, головним чином, звернення до тієї чи іншої внутрішньої процедури у відповідності до значення змінної *flag* на кшталт такого:

```
switch flag,
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
  case 1,
    sys=mdlDerivatives(t,x,u);
  case 2,
    sys=mdlUpdate(t,x,u);
  case 3,
    sys=mdlOutputs(t,x,u);
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
    sys=mdlTerminate(t,x,u);
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
```

- flag=0* виконується ініціалізація блоку S-функції;
- flag=1* здійснюється звернення до процедури правих частин неперервних диференціальних рівнянь;
- flag=2* обчислюються нові значення змінних стану на наступному кроці дискретизації (для дискретної S-функції, описуваної різницевиими рівняннями);
- flag=3* формується значення вектора вихідних величин блоку S-функції;
- flag=4* формується нове значення модельного часу, яке відлікується від моменту перетинання заданого рівня певною змінною стану;
- flag=9* припиняється робота блоку S-функції.

Встановлення і змінювання значень змінної *flag* здійснюється автоматично, без втручання користувача, у відповідності з логікою функціонування блоків *Simulink* при моделюванні.

Отже, використання S-функції дає змогу моделювати роботу як звичайних алгебричних, так й динамічних (неперервних або дискретних) ланок.

Слід відзначити ще один, більш зручний, спосіб об'єднати S-моделі з програмами мовою Matlab, який полягає у можливості виклику M-файлів безпосередньо з S-моделі спеціально передбаченими для цього засобами.

Нехай потрібно перед початком роботи S-моделі з іменем **MODEL.mdl** завантажити деякий інший M-файл, наприклад, з іменем **PERVdan**, який містить операції присвоювання первинних значень усім даним. Це можна здійснити, якщо при утворенні S-моделі з цим іменем ввести у командному вікні Matlab команду

```
set_param('MODEL','PreLoadFcn','PERVdan')
```

Вона зв'яже файл **PERVdan.m** з S-моделлю **MODEL.mdl** так, що він буде автоматично викликатися при виклику цієї S-моделі. Якщо після виконання вказаної команди записати на диск цю S-модель, то при подальших її викликах спочатку автоматично буде викликаний файл **PERVdan.m** і лише після цього на екрані виникне блок-схема S-моделі, готова до моделювання.

Перевірити, який саме M-файл використовується у даній S-моделі як попередньо виконуваний, можна шляхом застосування команди

```
get_param('ім'я S-моделі','PreLoadFcn')
```

За допомогою функції **set_param** можна встановити в S-моделі значення багатьох її параметрів, в тому числі й параметрів окремих її блоків.

У загальному вигляді звернення до неї може бути таким

```
set_param(Ім'я_S-моделі/Ім'я_блоку,'Параметр1',Значення1,...  
'Параметр2',Значення2,...)
```

Якщо вказано Ім'я_блоку, то наступні значення присвоюються параметрам цього блоку.

Наведемо приклади.

Виклик вигляду

```
set_param('MODEL','Solver','ode15s','StopTime','3000')
```

приведе до встановлення в S-моделі MODEL розв'язувача **ode15s** и часу закінчення процесу моделювання 3000.

Якщо звернутися до цієї функції так:

```
set_param('MODEL/Рівняння','Gain','1000')
```

то в блоці "Рівняння" S-моделі MODEL параметрові *Gain* буде присвоєно значення 1000.

Команда

```
set_param('MODEL/Fcn','Position',[50 100 110 120])
```

встановить зображення блоку "Fcn" у S-моделі MODEL у прямокутник у вікні блок-схеми з координатами [50 100 110 120].

Якщо ж здійснити таке звернення:

```
set_param('mymodel/Compute','OpenFcn','my_open_fcn')
```

то блок Compute S-моделі mymodel буде зв'язано з M-програмою MatLab, що записана у файлі my_open_fcn.m. Після цього файл my_open_fcn.m буде викликатися до виконання кожного разу, коли на зображенні блоку Compute двічі клацнути мишкою.

Коли потрібно викликати деякий М-файл перед або після проведення власне моделювання на S-моделі (наприклад, потрібно викликати програму, яка дозволяє змінити встановлені значення параметрів моделі у діалоговому режимі, або використати програму виведення результатів моделювання у графічній формі), можна встановити на вільному місці блок-схеми пусті блоки Subsystem (з розділу *Ports & Subsystems* бібліотеки Simulink), кожен з яких здійснюватиме виклик відповідного М-файла. Ці пусті блоки Subsystem у полі блок-схеми моделі можна з'єднати з певною М-програмою, набравши у командному вікні команду, аналогічну останній.

Докладніше з засобами взаємодії пакету Simulink з середовищем Matlab можна ознайомитися у главі 4 цього навчального посібника. Там же, а також у [1, урок 8, с. 348–378] наведені приклади застосування описаних засобів для формування програмних моделей.

9.6. Утворення і використання власних бібліотек блоків користувача

Коли користувач поглиблено займається моделюванням систем, він неодмінно, рано чи пізно, стикнеться з необхідністю створення власних блоків, що мають властивості стандартних бібліотечних блоків пакета Simulink. Потреба у цьому виникає, коли користувач при розв'язуванні різних задач моделювання у власній предметній області вимушений або неодноразово використовувати елементарні утворені ним блоки, які є оригінальними і не входять до складу стандартних бібліотек Simulink, або використовувати одні й ті самі блоки багато разово у певних стійких їх з'єднаннях. У таких випадках можна значно зменшити час утворення нової моделі і при цьому запобігти численним помилкам, якщо оформити ці нові блоки або з'єднання блоків як нові блоки і розташувати їх у бібліотеці.

Переваги використання власних блоків у складі бібліотек полягають у такому:

- їх можна використовувати неодноразово шляхом перетягування зображення блоку з бібліотеки у вікно блок-схеми моделі;
- користуватися ними зручніше за все, коли спілкування з ними здійснюється через спеціальні діалогові вікна настроювання блоків, аналогічні тим, що застосовані у стандартних блоках Simulink.

Утворення вікон настроювання блоків здійснюється через так зване *маскування* блоку, тобто створення *маски* блоку.

Приклади утворення і застосування користувацької бібліотеки наведені у главі 4, а також у [1, урок 8, с. 377–402]

10. ВИКОРИСТАННЯ БІБЛІОТЕКИ AEROSPACE

Фахівці різного профілю, ґрунтуючись на можливостях ядра *Simulink*, розробили кілька S-бібліотек, пристосованих для вирішення специфічних задач своєї галузі. Деякі з таких бібліотек включені у комплект поставки пакета *Simulink*. Однією з них є бібліотека *Aerospace Blockset*, призначена для моделювання динаміки польотів аерокосмічних об'єктів.

10.1. Загальна характеристика бібліотеки Aerospace

Увійдіть у браузер *Simulink*, і за допомогою контекстного меню бібліотеки *Aerospace Blockset* викличьте вікно *aerolibv1* бібліотеки (рис. 10.1).

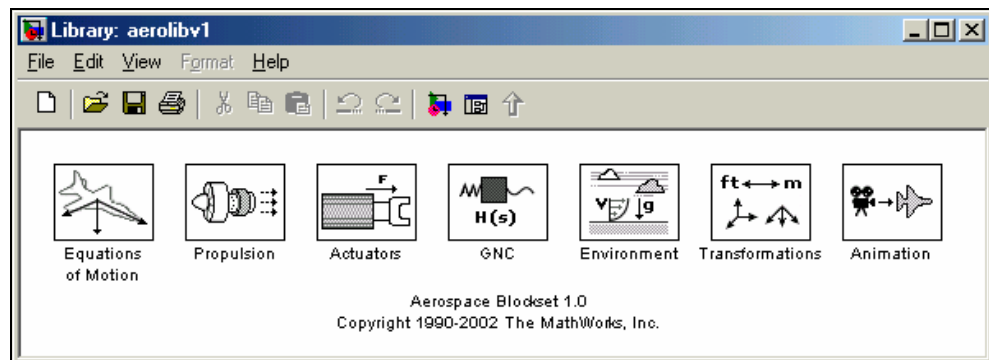


Рис.10. 1. Окно бібліотеки *aerolibv1*

Як бачимо, у склад бібліотеки входять шість розділів:

Equatios of Motion

(Рівняння руху)

Propulsion

(Двигун)

Actuators

(Привод, Рульові машинки)

GNC

(Система керування)

Environment

(Середовище)

Transformations

(Перетворення)

Animation

(Анімація)

містить блоки, що дозволяють скласти модель літального апарату;

містить блоки, що моделюють вплив двигунної установки літального апарату;

містить блоки, що моделюють поведінку приводу рулів літального апарату;

містить блоки моделювання системи керування рухом літального апарату;

складається з блоків, що моделюють вплив оточуючого середовища на рух літального апарату

містить блоки перетворення координат

містить блоки, які дозволяють побудувати анімаційні зображення руху літального апарату у просторі.

10.1.1. Розділ Equations of Motion

У розділі Equatons of Motion розташовані дві групи блоків (рис. 10.2): 6DoF (6 Degree of freedom – 6 ступенів вільності) і 3DoF (3 Degree of freedom – 3 ступеня вільності).

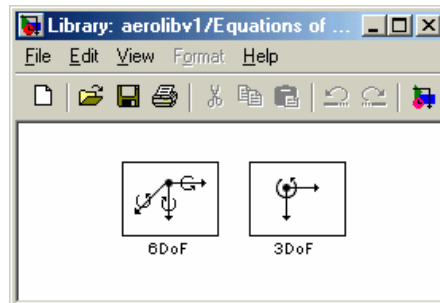


Рис. 10. 2. Вміст розділу Equations of Motion

У першій групі розташовані блоки, що дозволяють задати модель просторового руху (з шістьма ступенями вільності – три переміщення уздовж осей декартової системи координат і три кути повороту літального апарата (ЛА) відносно цієї системи координат). Двічі клацнувши мишею на зображенні групи 6DoF, ви отримаєте на екрані вікно, подане на рис. 10.3.

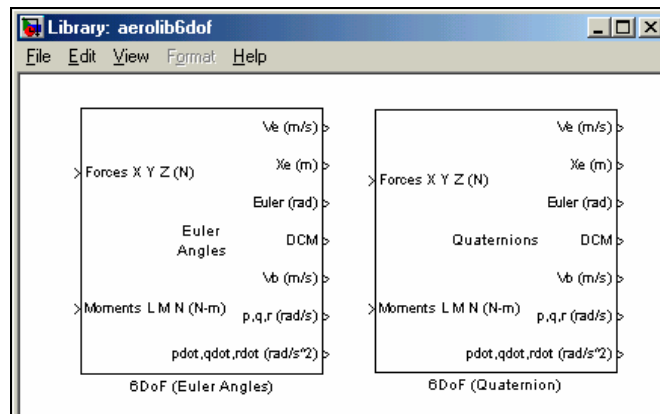


Рис. 10.3. Блоки підрозділу aerolib 6dof групи 6DoF

У ньому ви виявите два блоки - **6DoF (Euler Angles)** і **6DoF (Quaternion)**. Обидва блоки являють собою моделі поведінки твердого тіла з шістьма ступенями вільності. Але перший з них здійснює подання кутового руху тіла в так званих кутах Ейлера, а другий – у вигляді кватерніона повороту.

Вікна налаштування блоків (рис. 10.4 та 10.5) майже не відрізняються.

Перш ніж ознайомитися зі змістом цих вікон, слід обумовити особливості і позначення систем координат, які використовуються в бібліотеці *Aerospace*.

В якості основної (базової) системи координат тут прийнято систему декартових (взаємно ортогональних) осей $X_e Y_e Z_e$, пов'язану з поверхнею Землі. При цьому вісь Z_e припускається вертикальною і спрямованою вниз, до центру Землі.

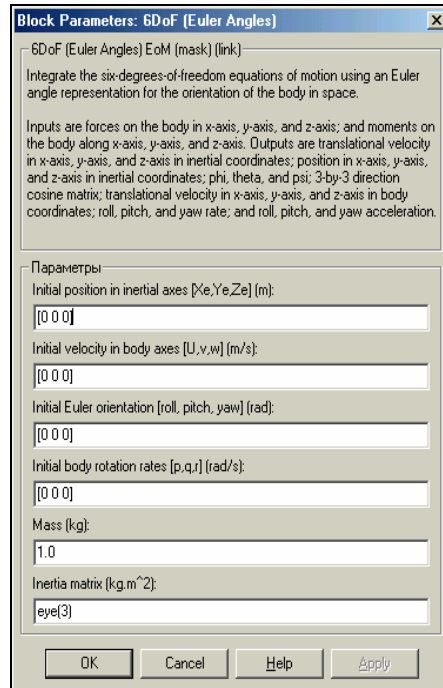


Рис. 10.4. Вікно налаштування блоку 6DoF (Euler Angles)

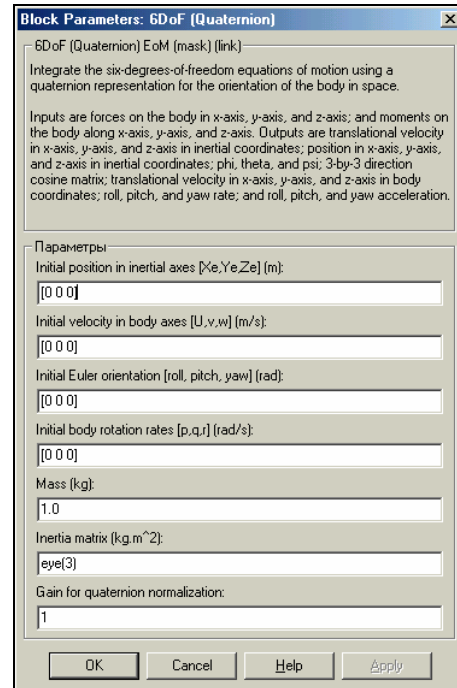


Рис. 10.5. Вікно налаштування блоку 6DoF (Quaternion)

Дві інші осі лежать у площині горизонту. Земля припускається нерухомою, плоскою і такою, що не обертається в просторі. Звідси випливає:

- 1) система земних осей $X_e Y_e Z_e$ за цих умов є також і інерціальною;
- 2) за допомогою бібліотеки *Aerospace* можна вивчати рух поблизу поверхні Землі лише на невеликих відстанях від початкової точки і протягом невеликого проміжку часу, коли кривизною Землі і її обертанням у просторі можна знехтувати.

Друга система координат $X_b Y_b Z_b$ за замовчуванням має початок у центрі мас літального апарату (ЛА). Вісь X_b спрямована по поздовжній осі ЛА до носа, вісь Y_b перпендикулярна їй, лежить у площині крил і спрямована праворуч (якщо дивитися з хвоста на ніс ЛА), вісь Z_b перпендикулярна площині крил і напрямлена вниз.

Проекції вектора \mathbf{V} швидкості ЛА на осі $X_b Y_b Z_b$ позначаються u_b , v_b и w_b відповідно, проекції вектора $\boldsymbol{\omega}$ абсолютної кутової швидкості ЛА – відповідно p , q і r , а проекції вектора \mathbf{M} моменту зовнішніх сил, що діють на ЛА – L , M і N .

Кути Ейлера, які використовуються в бібліотеці, складаються з кутів ристання ψ (yaw), тангажа θ (pitch) і крена ϕ (roll). Кут ристання являє собою кут відхилення у площині горизонту поздовжньої осі X_b ЛА від напрямку осі X_e земної системи координат. Кут тангажа – це кут підйому поздовжньої осі ЛА над площиною горизонту, а кут крена є кутом повороту корпусу ЛА навколо поздовжньої його осі.

Повертаючись до вікон настроювання, відзначимо, що в параметри настроювання входять такі величини:

Initial position in inertial axes [Xe,Ye,Ze] (m)	Початкове положення у інерціальних (земних) осях. Слід задати початкове відхилення центру мас Про ЛА від початку земної системи координат в метрах
Initial velocity in body axes [u,v,w] (m/s)	Початкові швидкості по осях тіла. Слід задати проекції швидкості центру мас ЛА у початковий момент часу на осі, пов'язані з ЛА, в метрах за секунду
Initial Euler orientation [roll,pitch,yaw] (rad)	Початкова орієнтація в кутах Ейлера. Слід задати початкові кути крену, тангажу і рискання в радіанах
Initial body rotation rates [p,q,r] (rad/s)	Початкові кутові швидкості тіла. Слід задати початкові значення проекцій кутової швидкості ЛА на осі, пов'язані з ЛА, в радіанах в секунду
Mass (kg)	Задається значення маси ЛА в кілограмах
Inertia matrix (kg.m²)	Задається матриця 3 × 3 моментів інерції ЛА відносно осей, пов'язаних з ним

Вхідні величини в обох блоків однакові. Це вектор поточних проекцій на осі ЛА всіх зовнішніх сил, що діють на нього, і вектор поточних моментів сил відносно осей ЛА.

Виходи обох блоків також однакові. Вони перелічені у наступній таблиці.

Ve (m/s)	Вектор проекцій поточного значення вектора швидкості центра мас ЛА на осі земної (інерціальної) системи координат
Xe (m)	Вектор поточних зміщень центру мас ЛА щодо початку земної (інерціальної) системи
Euler (rad) DCM	Вектор поточних значень кутів крену, тангажу і рискання відповідно Поточне значення матриці напрямних косинусів пов'язаних осей відносно земних осей
Vb (m/s)	Вектор проекцій поточного значення вектора швидкості центра мас ЛА на осі системи координат, пов'язаної з корпусом ЛА
p,q,r (rad/s)	Вектор проекцій поточної кутової швидкості ЛА на осі, пов'язані з ЛА
pdot,qdot,rdot (rad/s²)	Вектор похідних від проекцій поточної кутової швидкості ЛА на осі, пов'язані з ЛА

Друга група блоків 3DoF дозволяє моделювати рух ЛА в одній площині (зазвичай – поздовжній рух у вертикальній площині). Вона містить два блоки (рис. 10.6) – *Equations of Motion (Body Axes)* (Рівняння руху в пов'язаних осях) і *Incidence & Airspeed* (Кут атаки і повітряна швидкість).

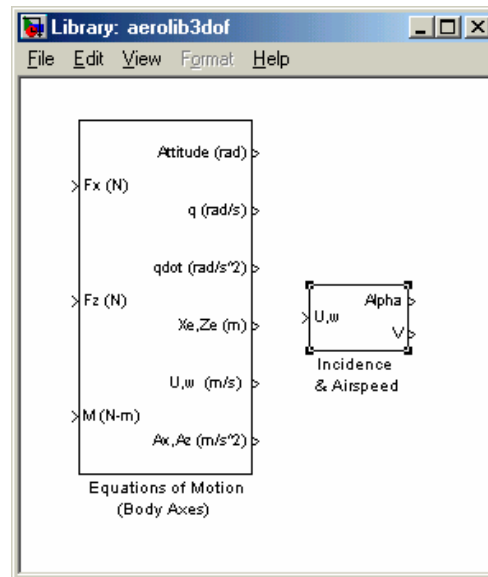
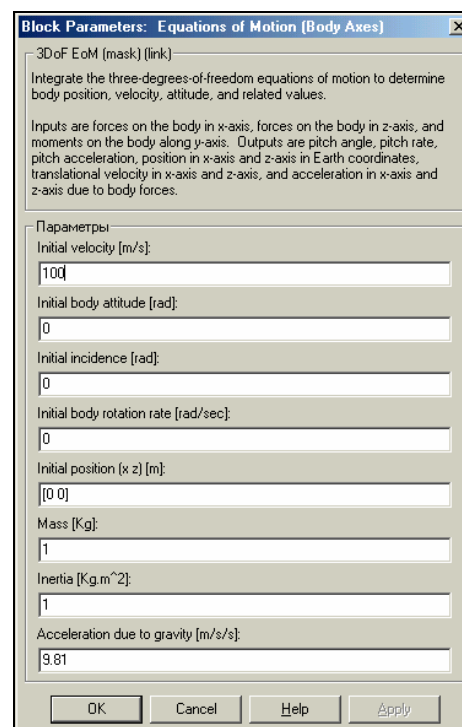


Рис. 10.6. Вміст розділу 3DoF

Перший блок дозволяє моделювати поздовжній рух шляхом чисельного інтегрування рівнянь поздовжнього руху ЛА. Другий обчислює за заданими проекціями швидкості ЛА на осі X_b і Y_b кут атаки α і величину вектора повітряної швидкості.

На рис. 10.7. показано вікно настроювання блоку *Equations of Motion*.

Рис. 10.7. Вікно настроювання блоку *Equations of Motion*

З його допомогою вводяться наступні параметри, необхідні для чисельного інтегрування диференціальних рівнянь поздовжнього руху.

Initial velocity (m/s)	Початкова швидкість
Initial body attitude (rad)	Початковий кут підйому вектора швидкості над площиною горизонту
Initial incidence (rad)	Початковий кут атаки
Initial body rotate rate (rad/sec)	Початкова кутова швидкість тангажа
Initial position [x z] (m)	Початкове положення центра мас
Mass (kg)	Маса ЛА
Inertia (kg m²)	Момент інерції ЛА відносно поперечної осі
Acceleration due to gravity (m/s/s)	Прискорення сили гравітації

Входи блоку переліковані нижче.

F_x (N)	поточне значення сили (в ньютонках), що діє на ЛА уздовж його проздовжньої осі X_b
F_z (N)	поточне значення сили (в ньютонках), що діє на ЛА вздовж його нормальній осі Z_b
M (Nm)	поточне значення моменту сил (в ньютонках на метр), що діє на ЛА навколо його поперечної осі Y_b

Виходами блоку є нижчезазначені величини.

Altitude (rad)	поточне значення кута між вектором швидкості ЛА і площиною горизонту (в радіанах)
q (rad/s)	поточне значення проекції кутової швидкості ЛА на його поперечну вісь (в радіанах в секунду)
qdot (rad/s²)	поточне значення проекції кутового прискорення ЛА на його поперечну вісь (в радіанах в секунду в квадраті)
X_e, Z_e (m)	поточні координати центра мас ЛА у поздовжній площині в земній системі координат (в метрах)
U, w (m/s)	поточні значення проекцій швидкості ЛА відповідно на поздовжню і нормальну осі ЛА (в метрах в секунду)
A_x, A_z (m/s²)	поточні значення проекцій прискорення ЛА відповідно на поздовжню і нормальну осі ЛА (в метрах в секунду в квадраті)

Розглянуті блоки є основними для моделювання руху ЛА. В них зосереджені програми, які здійснюють чисельне інтегрування диференціальних рівнянь. Але для їх роботи необхідно формувати поточні значення сил і моментів сил, що діють на ЛА протягом його польоту. Ці сили і моменти можна поділити на три групи:

- сили і моменти, що діють на ЛА з боку маршового двигуна;
- сили і моменти, що діють на корпус ЛА з боку оточуючого його середовища; сюди відносяться сили й моменти аеродинамічного опору рухові ЛА в атмосфері і сила тяжіння ЛА;

- сили й моменти, що накладаються на ЛА з боку керуючих органів, таких як керма, елерони, елевони, закрилки і т. п.

У загальному випадку ці сили і моменти настільки розрізняються для різних типів ЛА, що не можна створити універсальні блоки для їх обчислення. Тому в бібліотеці *Aerospace* є лише кілька блоків, що формують залежності в законах утворення деяких сил і моментів, які найбільш часто трапляються.

10.1.2. Розділ Environment

У цьому розділі бібліотеки містяться три групи блоків, які дозволяють враховувати вплив параметрів навколишнього середовища на сили і моменти, діючі на ЛА в його польоті (рис. 9.8), - *Atmosphere* (Атмосфера), *Gravity* (Гравітація) і *Wind* (Вітер).

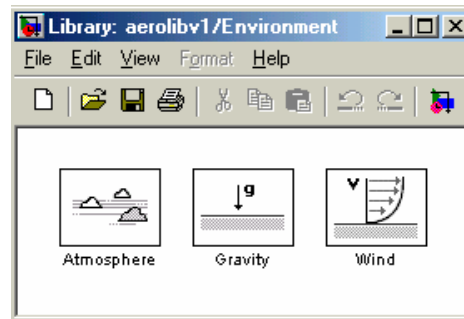


Рис. 10. 8. Вміст розділу Environment

У першій групі (Atmosphere) містяться блоки (рис. 10.9) *ISA Atmosphere Model* (ISA-модель атмосфери) і *COESA Atmosphere Model* (COESA-модель атмосфери).

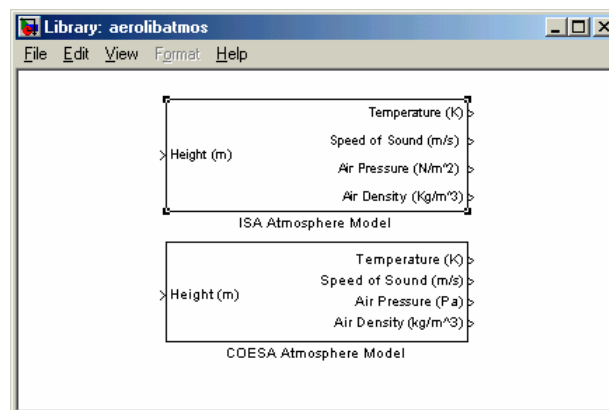


Рис. 9. 9. Вміст групи Atmosphere

Ці блоки розраховують параметри атмосфери на поточній висоті польоту. Вхідним параметром обох блоків є значення поточної висоти польоту ЛА. Вихідні параметри наведені нижче.

Temperature (K)
Speed of sound (m/s)

Температура (в градусах Кельвина)
Швидкість звука (в метрах в секунду)

Air Pressure (N/m²)
Air Density (kg/m³)

Тиск повітря (ньютон на метр в квадраті)
Густина повітря (кілограмм на метр кубічний)

Перший блок здійснює розрахунки за міжнародною моделлю стандартної атмосфери до висоти польоту 20 км. Другий – за американською розширеною моделлю стандартної атмосфери. Тут можливе врахування особливостей атмосфери до висоти 84 852 м. Параметрів настроювання у цих блоків немає.

Друга група *Gravity* – містить лише один блок *WGS84 Gravity Model* (рис. 10.10), який розраховує значення прискорення вільного падіння на поточній висоті польоту ЛА і на поточній географічній широті.

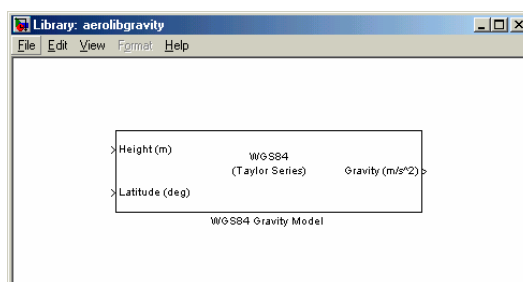


Рис. 10. 10. Вміст групи Gravity

10.1.3. Розділ Propulsion

Розділ *Propulsion* (Двигун) містить єдиний блок (рис. 10.11) – *Turbofan Engine System* (Система турбовентиляторного двигуна).

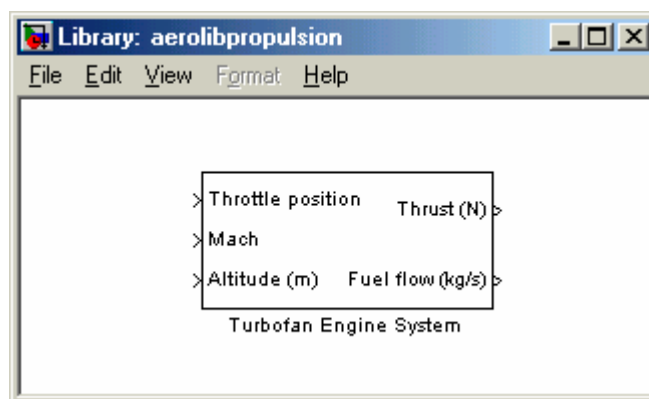


Рис. 10. 11. Вміст розділу Propulsion

Блок має такі входи:

Throttle position
Mach
Altitude (m)

Поточне положення регулюючого дроселя
Поточне значення числа Маха
Поточне значення висоти польоту (в метрах)

і такі виходи:

Thrust (N)
Fuel flow (kg/s)

Сила тяги (в ньютонях)
Витрата пального (в кілограмах в секунду)

Вікно настроювання блоку наведено на рис. 10.12.

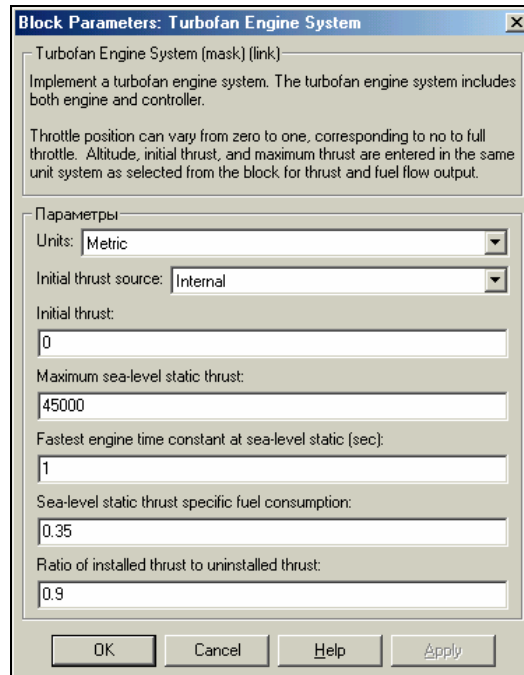


Рис. 10. 12. Вікно настроювання блоку Turbofan Engine System

У число параметрів настроювання блоку входять:

Initial thrust source	Джерело (внутрішнє або зовнішнє) початкового значення тяги двигуна
Initial thrust	Початкове значення тяги двигуна
Maximum sea level static thrust	Максимальне значення сили тяги на рівні океану
Fastest engine time constant at sea level static (sec)	Стала часу двигуна на рівні океану (в секундах)
Sea level static thrust specific fuel consumption	Питома витрата палива на рівні океану
Ratio of installed thrust to uninstalled thrust	Відношення встановленої сили тяги до невстановленої

10.1.4. Розділи Actuators і GNC

Розділи *Actuators* (Виконавчі елементи) і *GNC* (Регулятори керування рухом) містять блоки, що допомагають створити модель системи автоматичного керування рухом ЛА.

У загальному випадку система автоматичного керування рухом складається з наступних частин:

– вимірювачів параметрів руху ЛА – гіроскопічних приладів, які вимірюють кути повороту корпусу ЛА і його кутові швидкості, вимірювачів швидкості ЛА, кутів атаки і дрейфу і т. п.;

– регуляторів – ланок системи керування, які формують закон керування (необхідні залежності регульованих параметрів від виміряних величин);

– силового приводу, який забезпечує поворот керма, елеронів і елевонів на необхідні кути, величини яких визначені регулятором;

– виконавчих органів (рулів, елеронів і елевонів), які забезпечують накладення на ЛА необхідних моментів сил.

Моделі вимірювачів параметрів руху та роботи виконавчих органів користувачеві необхідно створювати самому. Універсальних блоків, що моделюють їх поведінку в бібліотеці немає.

Розділ *Actuators* містить блоки, що моделюють рух силового приводу рульових виконавчих органів (рис. 10.13).

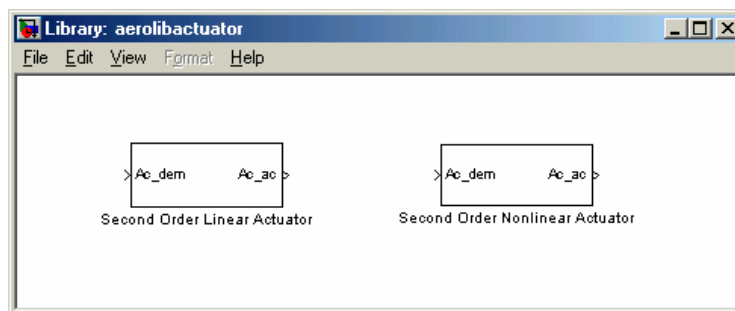


Рис. 10.13. Вміст розділу *Actuators*

У ньому є два блоки – *Second Order Linear Actuator* (Лінійний силовий привід другого порядку) і *Second Order Nonlinear Actuator* (Нелінійний силовий привід другого порядку). Обидва блоки моделюють процес переміщення рульового органу при подачі на вхід силового приводу сигналу з заданим значенням цього переміщення як проходження цього заданого сигналу через ланку другого порядку з заданими частотою власних коливань та коефіцієнтом демпфірування.

В обох блоках входом є поточне потрібне значення (*Ac_dem*) положення регулюючого органу, а виходом – дійсне (*Ac_ac*) значення його положення.

Вікна налаштування блоків зображені на рис. 10.14 і 10.15.

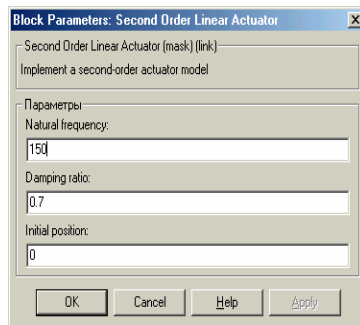


Рис. 10.14. Вікно настроювання блоку *Second Order Linear Actuator*

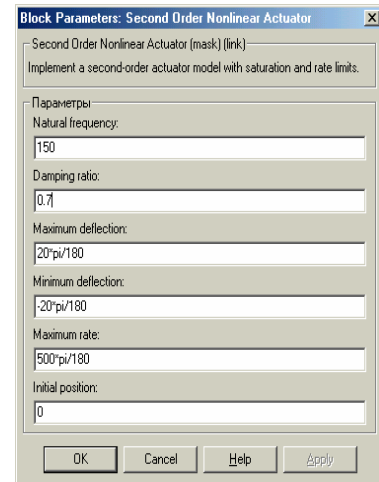


Рис. 10.15. Вікно настроювання блоку *Second Order Nonlinear Actuator*

Обидва блоки мають три однакові параметри настроювання:

Natural frequency

Частота власних коливань

Damping ratio

Відносний коефіцієнт загасання

Initial position

Початкове положення регулюючого органу

Блок нелінійного силового приводу містить ще такі параметри настроювання:

Maximum deflection

Максимальне відхилення регулюючого органу

Minimum deflection

Мінімальне відхилення регулюючого органу

Maximum rate

Максимальна швидкість відхилення регулюючого органу

Саме наявністю вказаних обмежень на відхилення регулюючого органу і його швидкість відрізняється нелінійний привід от лінейного.

Вміст розділа GNC показано на рис. 10.16. В нього входять блоки, які моделюють процес формування сигналів, які керують відхиленням рульових органів ЛА.

Крім трьох останніх допоміжних блоків, які здійснюють інтерполяцію матриць, 13 блоків цього розділу покликані виробляти сигнал, пропорційний необхідному куті повороту регулюючого органу. Тому вихід у всіх цих блоках один - u - потрібне поточне положення регулюючого органу.

У 11 блоках, що моделюють різного виду регулятори і спостерігачи, основним входом є вектор « u » величин, що характеризують поточний рух об'єкту і вимірюваних на борту ЛА наявними вимірювальними приладами. Більш детальне вивчення цих блоків становить інтерес, головним чином, для фахівців в області проектування систем автоматичного керування рухом ЛА і не входить у завдання попереднього ознайомлення з можливостями бібліотеки *Aerospace*.

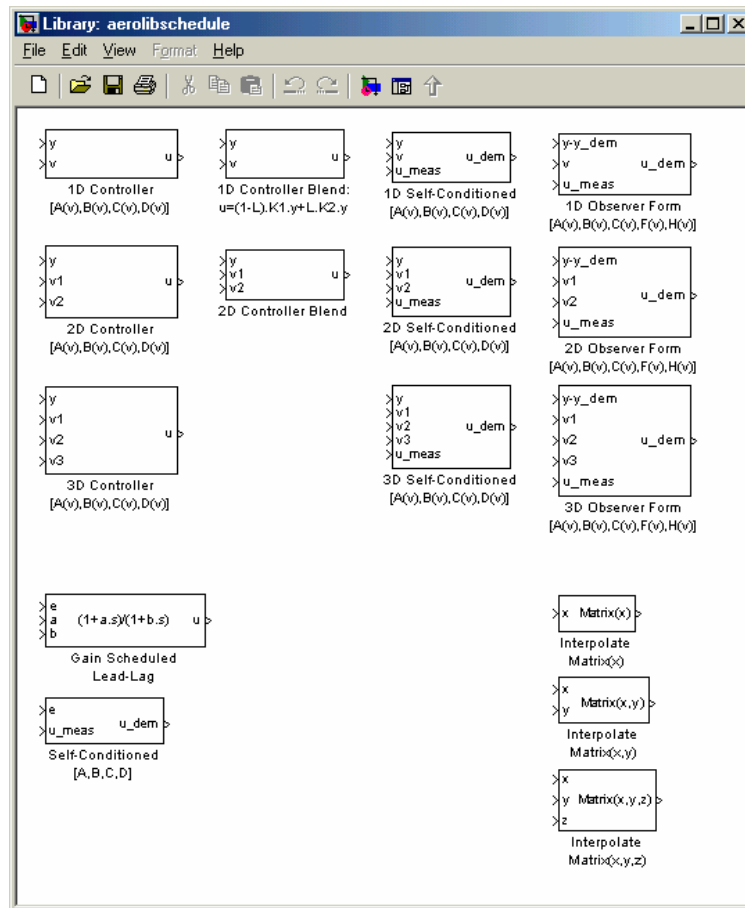


Рис. 10.16. Вміст розділу GNC

10.1.5. Розділ Transformations

Тут містяться блоки двох груп (рис. 10.17) – *Axes* (Осі) і *Units* (Одиниці).

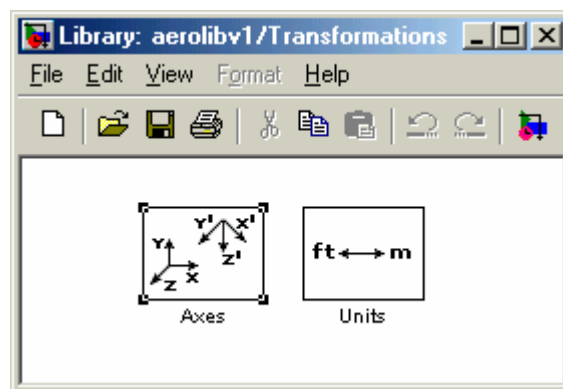


Рис. 10. 17. Вміст розділу Transformations

Група *Axes* (рис. 10.18) включає 6 блоків, які здійснюють різні перетворення форм подання кутового положення твердого тіла у просторі та 1 блок (3×3 *Cross Product*) векторного добутку двох трикомпонентних векторів.

Подамо призначення блоків перетворення координат.

Quaternions to Euler Angles

Перетворює кватерніон поворота у вектор трьох кутів Ейлера

<i>Quaternions to Direction Cosine Matrix</i>	Перетворює кватерніон поворота у матрицю напрямних косинусів
<i>Euler Angles to Quaternions</i>	Перетворює вектор трьох кутів Ейлера у вектор кватерніону поворота
<i>Direction Cosine Matrix to Quaternions</i>	Перетворює матрицю напрямних косинусів у кватерніон поворота
<i>Euler Angles to Direction Cosine Matrix</i>	Перетворює вектор трьох кутів Ейлера у матрицю напрямних косинусів
<i>Direction Cosine Matrix to Euler Angles</i>	Перетворює матрицю напрямних косинусів у вектор трьох кутів Ейлера

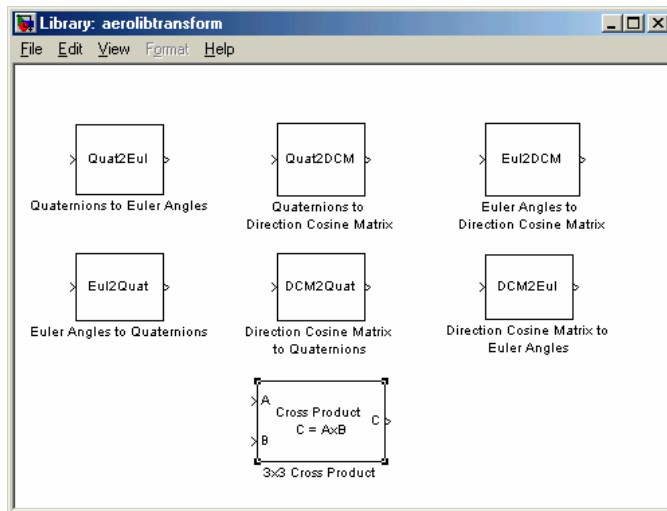


Рис. 10. 18. Вміст групи Axes

Друга група блоків – *Units* – містить (рис. 10.19) 11 блоків перетворення величин з однієї системи одиниць в іншу .

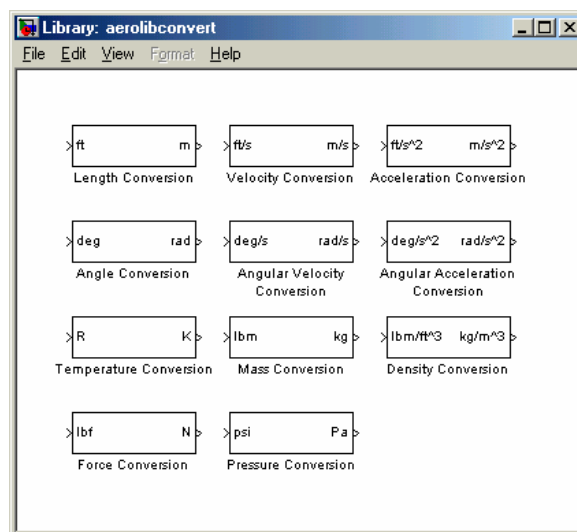


Рис. 10.19. Вміст групи Units

Призначення блоків впливає з їхніх назв. Параметри настроювання у всіх блоках відсутні.

10.2. Модель вільного кутового руху космічного апарату

Роботу з бібліотекою *Aerospace* почнемо зі створення простої моделі кутового руху КА в інерціальному просторі.

Варіант *SvDvigKA.mdl* блок-схеми цієї моделі поданий на рис. 10.20.

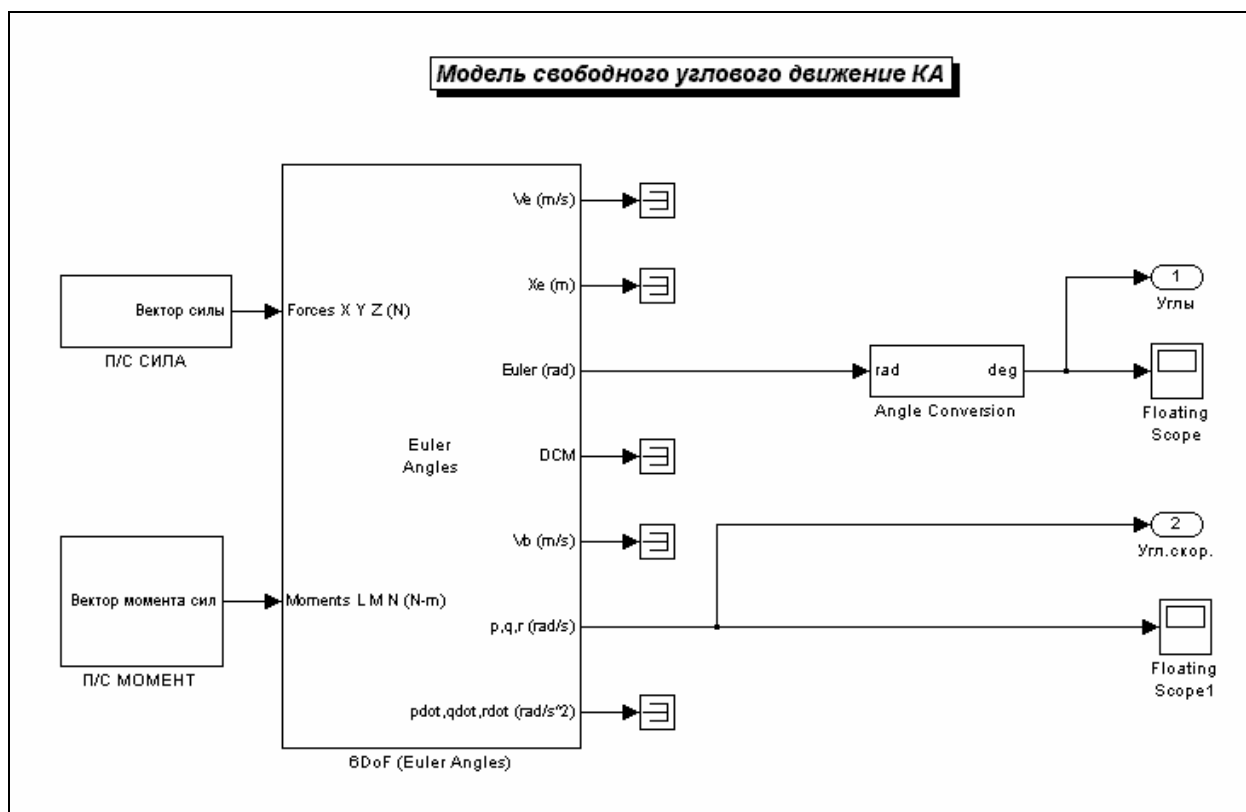


Рис. 10. 20. Модель вільного кутового руху КА

В основу моделі можна покласти блок 6DoF (Euler Angles), який забезпечує моделювання руху тіла з шістьма ступенями свободи.

Якщо завданням моделювання є дослідження вільного руху, то на вхід цього блоку слід подати векторні сигнали, що відповідають проекціям зовнішньої сили і зовнішнього моменту сил, рівні нулю. Це можна забезпечити за допомогою двох підсистем – «П/С СИЛА» (рис. 10.21) і «П/С МОМЕНТ» (рис. 10.22).

Надалі використовуватимемо такі позначення

m	Маса КА
J	Матриця моментів інерції КА
XYZ0	Вектор проєкцій координат центру мас КА
v0	Вектор проєкцій швидкості центру мас КА
UG0	Вектор кутів повороту КА

UgSk0

Вектор проєкцій кутової швидкості КА на осі КА

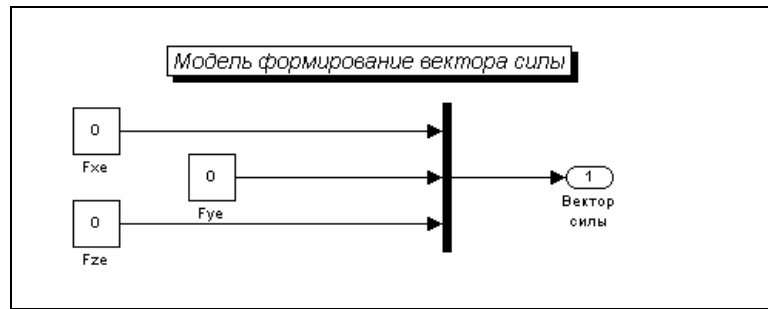


Рис. 10. 21. Підсистема «П/С СИЛА»

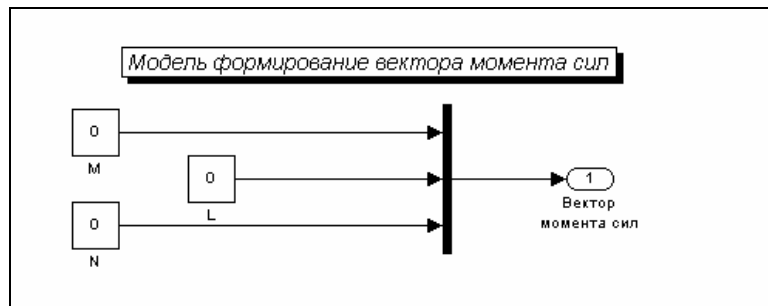


Рис. 10. 22. Підсистема «П/С МОМЕНТ»

На рис. 10.23 показані настройки блоку 6DoF (Euler Angles).

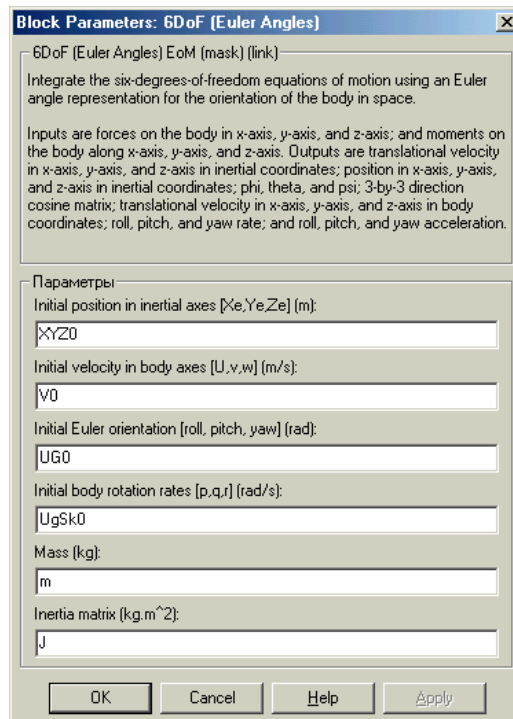


Рис. 10. 23. Настройки блоку 6DoF (Euler Angles)

Керування роботою моделі і виведення результатів здійснимо за допомогою керувальної програми **SvobDvigKA_upr**, текст якої наводиться нижче.

```
% SvobDvigKA_upr
```

```

% Керувальна програма для моделі SvDvigKA

% Лазарєв Ю.Ф. 14-07-2010

clear all, clc
% Встановлення параметрів КА
J=[3400 300 -200;300 2200 100;-200 100 1400]; % Матриця моментів інерції КА
m=2000; % Маса КА
% Встановлення початкових умов
XYZ0=[0 0 0]; % Початкове положення КА
V0=[0 0 0]; % Початкові швидкості КА
UG0=[0 0 0]; % Початкові кути КА
UgSk0=[1 0.1 0]; % Початкові кутові швидкості КА
% Встановлення параметрів інтегрування
TK=300; % Кінцевий термін інтегрування
hi=0.1; % Крок інтегрування
% Запуск моделі
sim('SvDvigKA');
% Запис результатів інтегрування
Fl=yout(:,1); TE=yout(:,2); PSI=yout(:,3);
omx=yout(:,4); omy=yout(:,5); omz=yout(:,6); t=tout;
% Графіческое представлення результатів
subplot(2,2,1)
plot(TE,PSI), grid
axis('equal'); set(gca,'FontSize',12,'FontName','MS Sans Serif')
title('Рух осі Xb у просторі');
xlabel('Кут \theta (градуси)'); ylabel('Кут \psi (градуси)');
subplot(2,2,3)
plot(t,TE,t,PSI, '.'), grid
set(gca,'FontSize',12,'FontName','MS Sans Serif'); title('Кути у часі');
xlabel('Час (с)'); ylabel('Кути (градуси)'); legend(' \theta ', ' \psi ',0);
subplot(2,2,4)
plot(t,omy,t,omz, '.'), grid, set(gca,'FontSize',12,'FontName','MS Sans Serif'); title('Кутові швидкості у часі');
xlabel('Час (с)'); ylabel('Кутові швидкості (рад/с)'); legend(' оmy ', ' omz ',0);
subplot(2,2,2), axis('off');
h=text(-0.3,1.1,'Вільний кутовий рух космічного апарату','FontSize',14,'FontName','MS Sans Serif');
h=text(0.1,0.9,'|','FontSize',12); h=text(0.2,0.9,num2str(J(1,1)),'FontSize',12);
h=text(0.4,0.9,num2str(J(1,2)),'FontSize',12); h=text(0.6,0.9,num2str(J(1,3)),'FontSize',12);
h=text(0.8,0.9,'|','FontSize',12); h=text(-0.1,0.8,'J = ','FontSize',12);
h=text(0.1,0.8,'|','FontSize',12); h=text(0.2,0.8,num2str(J(2,1)),'FontSize',12);
h=text(0.4,0.8,num2str(J(2,2)),'FontSize',12); h=text(0.6,0.8,num2str(J(2,3)),'FontSize',12);
h=text(0.8,0.8,'|','FontSize',12); h=text(0.1,0.7,'|','FontSize',12);
h=text(0.2,0.7,num2str(J(3,1)),'FontSize',12); h=text(0.4,0.7,num2str(J(3,2)),'FontSize',12);
h=text(0.6,0.7,num2str(J(3,3)),'FontSize',12); h=text(0.8,0.7,'|','FontSize',12);
h=text(-0.1,0.5,'Початкові кути (градуси)','FontSize',12,'FontName','MS Sans Serif');
h=text(0.1,0.4,['\psi0 = ',num2str(UG0(3)*180/pi)],'FontSize',12);
h=text(0.4,0.4,['\theta0 = ',num2str(UG0(2)*180/pi)],'FontSize',12);
h=text(0.7,0.4,['\phi0 = ',num2str(UG0(1)*180/pi)],'FontSize',12);
h=text(-0.1,0.2,'Початкові кутові швидкості (рад/с)','FontSize',12);
h=text(0.1,0.1,['omx0 = ',num2str(UgSk0(1))], 'FontSize',12);
h=text(0.4,0.1,['omy0 = ',num2str(UgSk0(2))], 'FontSize',12);
h=text(0.7,0.1,['omz0 = ',num2str(UgSk0(3))], 'FontSize',12);
h=text(-0.1,-0.05,'-----');
h=text(-0.1,-0.1,'Програма SvobDvigKA-upr Лазарєв Ю. Ф. 14-08-2004');
h=text(-0.1,-0.15,'-----');

```

На рис. 10.24 ... 10.26 наведені результати моделювання для трьох випадків різного розподілу мас КА: 1) матриця моментів інерції є діагональною (динамічно симетричний КА); 2) матриця моментів інерції є діагональною з різними моментами інерції; 3) матриця моментів інерції КА довільна.

В усіх випадках припускається, що КА приведений у обертання навколо осі X_b з порівняно великою кутовою швидкістю $\omega_{X0} = 1$ рад/с, що надає КА властивості триступеневого гіроскопа з власним кінетичним моментом $H = 3400$ Н м с.

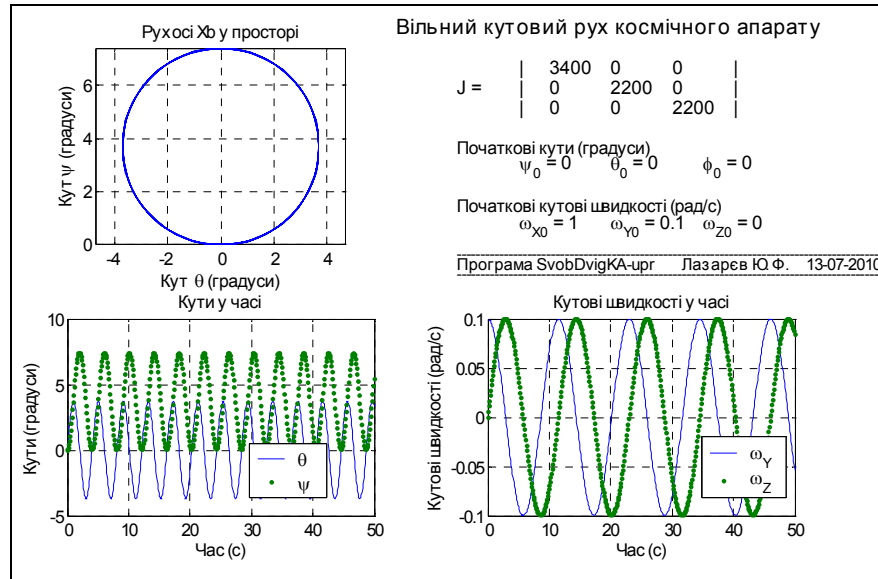


Рис. 10.24. Вільний рух динамічно симетричного КА

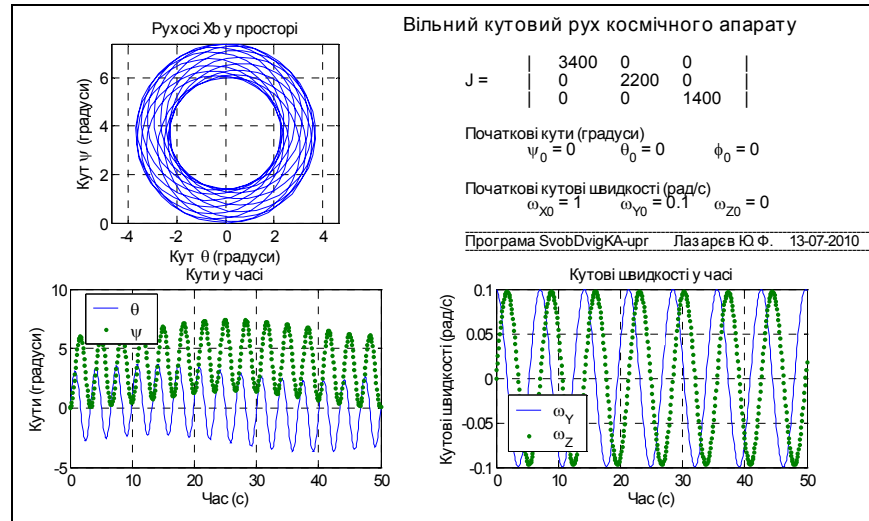


Рис. 9. 25. Вільний рух динамічно несиметричного КА

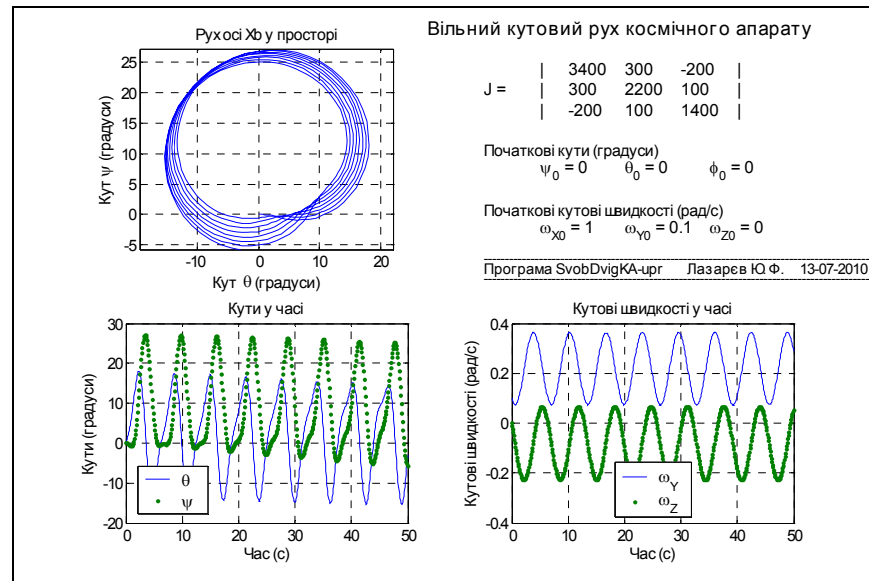


Рис. 10.26. Вільний рух динамічно незбалансованого КА

Крім того припускається, що цьому гіроскопу надається (наприклад, внаслідок удара) у початковий момент часу екваторіальна складова кутової швидкості $\omega_{y0} = 0,1$ рад/с навколо осі Y_b .

Отримані результати підтверджують висновки теоретичного аналізу, наведені у [14].

10.3. Модель керованого кутового руху космічного апарата

Перейдемо тепер до створення моделі керованого руху КА з кутів орієнтації. Задачею керування вважатимемо приведення КА у деяке нерухоме в інерціальному просторі положення.

Для забезпечення керування слід додати до попередньої моделі контур керування орієнтацією КА, припускаючи, що кути і кутові швидкості повороту КА в інерціальній системі координат вимірюються приладами, встановленими на його борті, і формуючі моменти сил керування на основі цієї інформації. Контур керування реалізований у моделі UprDvigKA.mdl, представлений на рис. 10.27, у вигляді підсистеми «СУО», блок-схему якої показано на рис. 10.28.

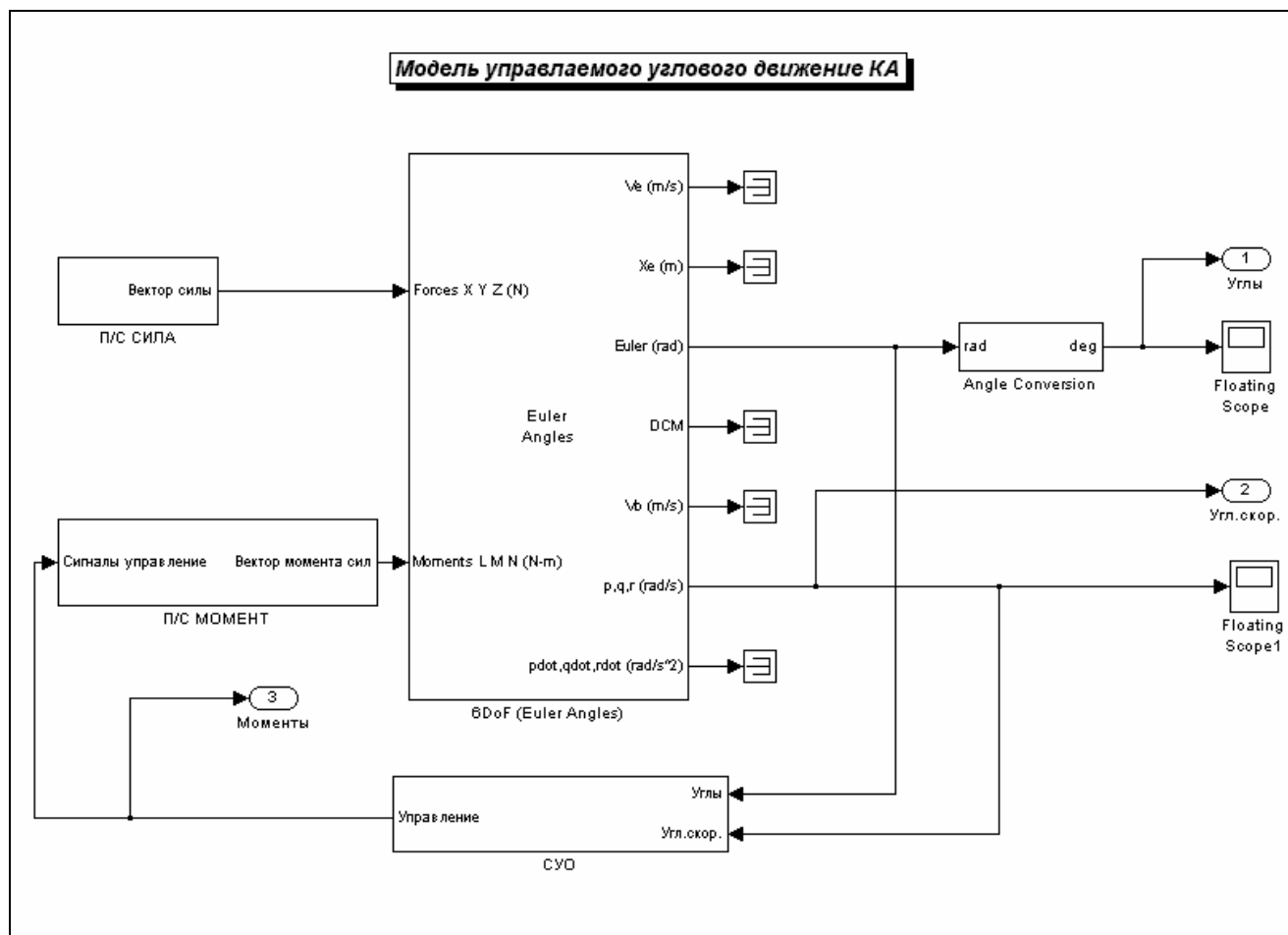


Рис. 10. 27. Блок-схема управляемого процесса ориентацией КА

Окрім прийнятих раніше тут використовуються такі позначення:

- kug** коефіцієнт обратной связи по углу отклонения
- Kugsk** коэффициент обратной связи по угловой скорости
- k** коэффициент компенсации гироскопического момента

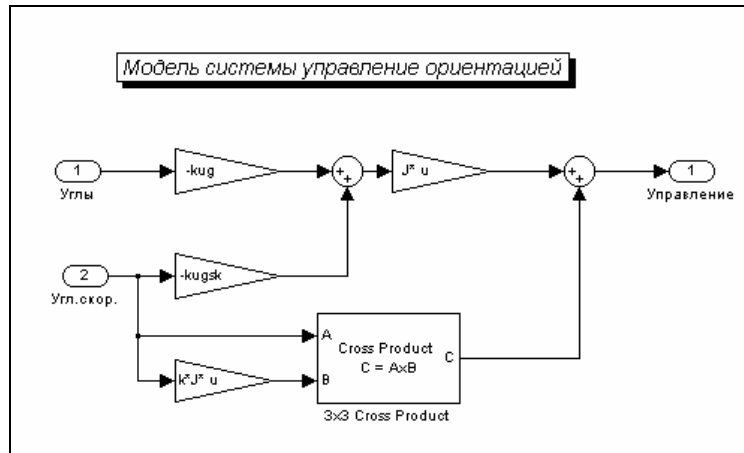


Рис. 10. 28. Блок-схема подсистемы «СУО»

При цьому припускається, що момент керування орієнтацією формується за законом:

$$M = -kug * J * Ug - kugsk * J * om - k * (om \times) * J * om,$$

де M – вектор проєкцій моменту керування, Ug – вектор кутів Ейлера, om – вектор проєкцій кутової швидкості, $(om \times)$ – косиметрична матриця з проєкцій кутової швидкості. Саме цей закон формується у підсистемі «СУО».

Проілюструємо роботу моделі на кількох прикладах.

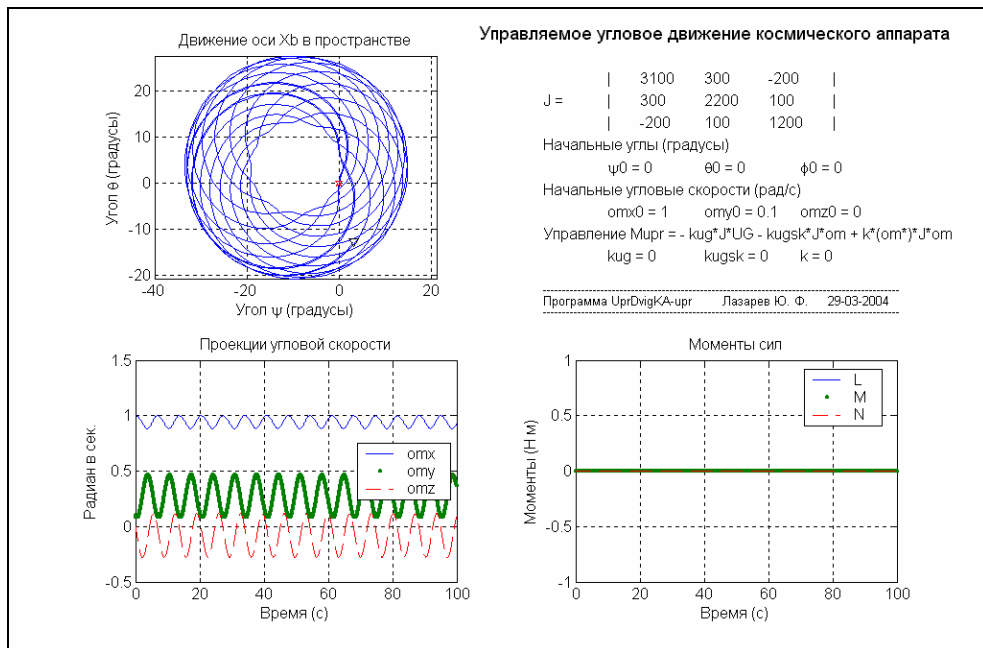


Рис. 10. 29. Некерований рух незбалансованого КА

Перевіримо роботу, моделюючи рух КА за відсутності керування ($k_{ug}=k_{ugsk}=k=0$).

На рис. 10.29 показані результати такого моделювання для КА з матрицею моментів інерції, прийнятою у [5]. Неважко впевнитися, що результат моделювання збігається з очікуванням (див. [14]).

Спочатку розглянемо, як вплине на рух КА компенсація гіроскопічного моменту ($k=1$). Результат наведений на рис. 10.30.

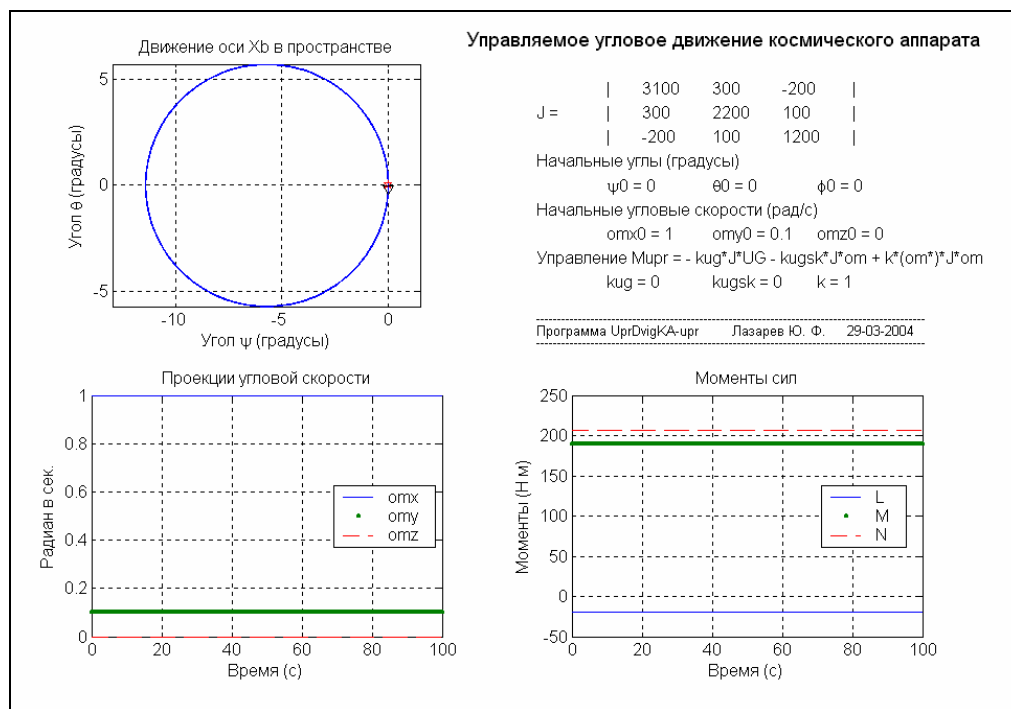


Рис. 10. 30. Вплив компенсації гіроскопічного моменту

Як неважко впевнитися, результатом керування з компенсацією гіроскопічного моменту є значне зменшення (приблизно у 5 разів) амплітуди коливань осі X_b .

Тепер розглянемо задачу демпфірування (точнее – приведення КА у нерухоме відносно інерціального простору положення). Для цього введемо керування (зворотний зв'язок) лише за швидкостями. Якщо, наприклад, встановити $k_{ugsk}=0.1$, а решту коефіцієнтів керування рівними нулю, то для динамічно симетричного КА отримаємо рух, відображений на рис. 10.31. Якщо до демпфіруючого моменту додати компенсацію гіроскопічного моменту, то рух КА буде здійснюватися (рис. 10.32) зі значно меншими відхиленнями осей від початкового положення.

Як бачимо, поставлене завдання успішно вирішується. Побічним наслідком такого керування є відхилення осі X_b від початкового положення на значні кути.

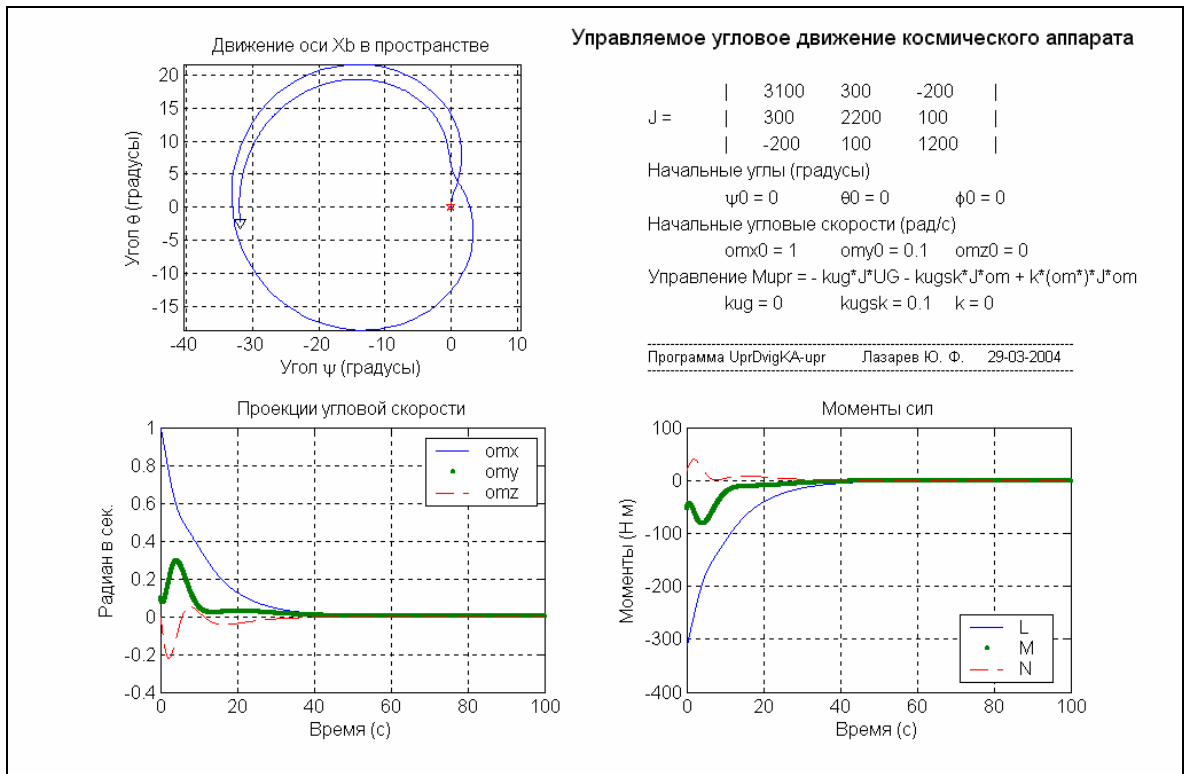


Рис. 10. 31. Режим "знерухомлення" КА

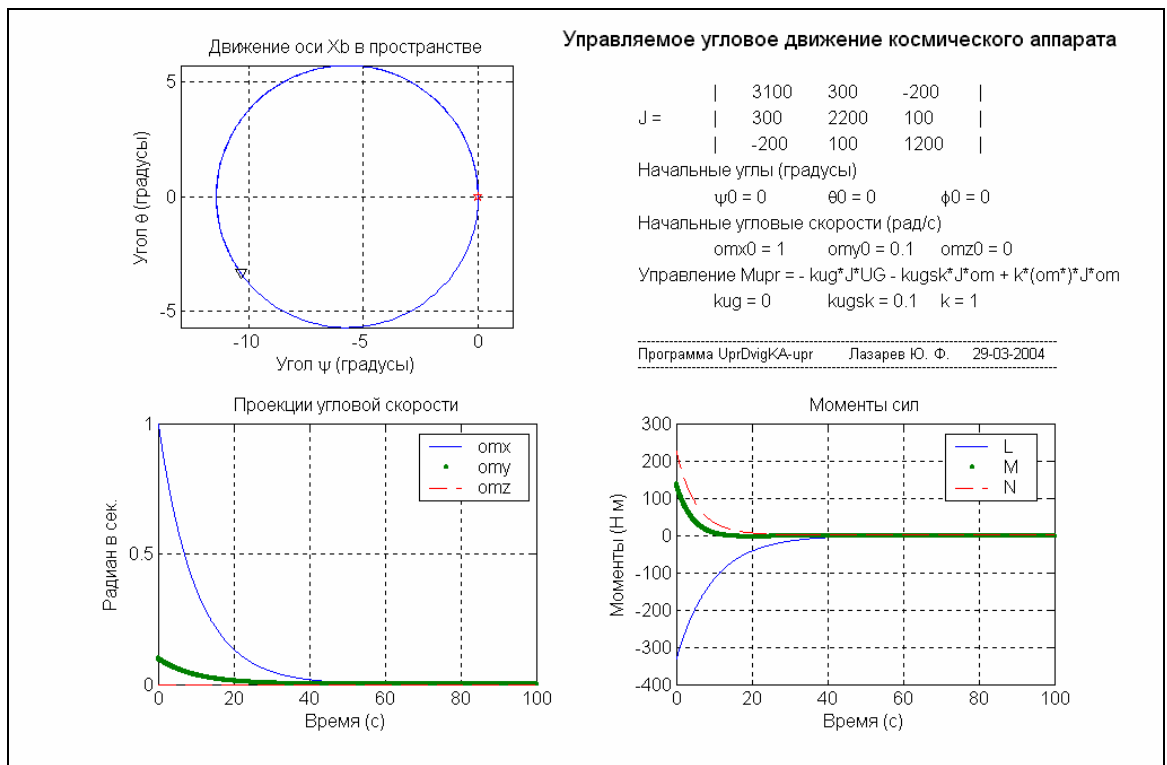


Рис. 10. 32. Режим "знерухомлення" КА з компенсацією гіроскопічного моменту

Перейдемо тепер до режиму приведення КА до заданого положення, під яким розумітимемо положення, коли усі три кути дорівнюють нулю. Для цього, окрім зворотного зв'язку зі швидкості введемо зворотний зв'язок з кутів. Поча-

ткові відхилення з кутів осі X_b , встановимо рівними 0.1 радіан. Тоді при $kug=0.1$ и $kugsk=0.3$ прийдемо до результатів, показаних на рис. 10.33.

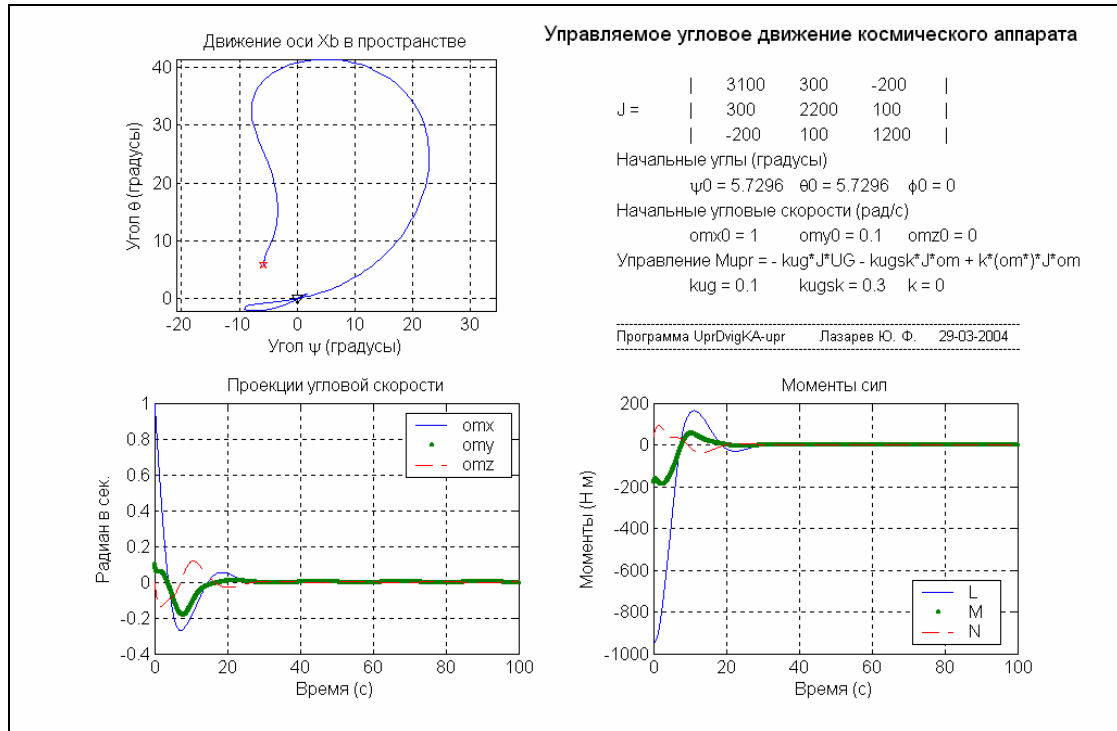


Рис. 10.33. Режим стабилизации положения КА

Додаткова компенсація гіроскопічного моменту (рис. 10.34) і в цьому разі поліпшує процес стабілізації.

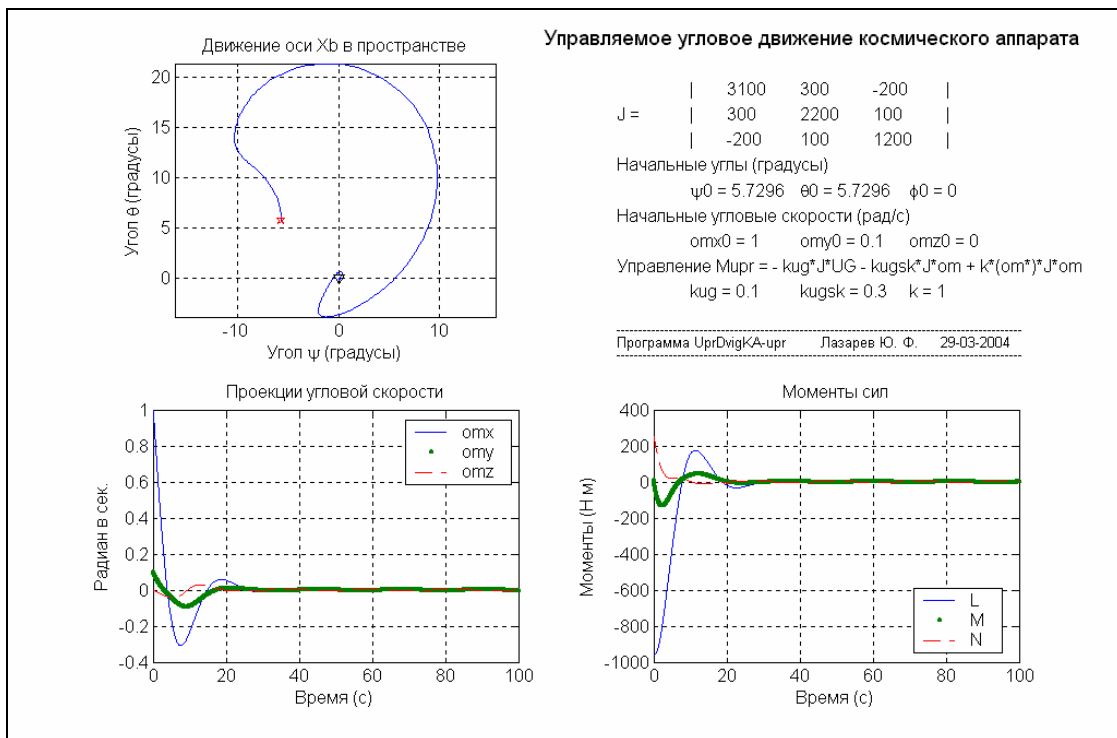


Рис. 10.34. Режим стабилизации с компенсацией гироскопического момента

Керування з кутів відхилення від заданого положення має низку недоліків. До них відноситься, перш за все, відхилення осей поворотів (за великих кутів) від осей прикладення відповідних керувальних моментів сил. Це може значно загальмувати процес стабілізації і навіть зробити його нестійким (за кутів відхилення більших 90°).

Цього недоліку позбавлено керування за кватерніонами, коли керувальні моменти сил по осях КА пропорційні складовим векторної частини кватерніону відхилення поточного положення КА від заданого. Ці складові, по-перше, пропорційні не кутам відхилень, а синусам половинних кутів, по-друге, ці складові кватерніону збіжні з осями, вздовж яких напрямлені керувальні моменти, а це забезпечує найшвидше (і без втрати стійкості) приведення КА у задане положення. Побічним, але теж важливим чинником є скінченність величин керувальних моментів за будь-яких значень кутів початкового відхилення.

Модель UprDvigKAqw.mdl керованого по кватерніонах процесу орієнтації відрізняється від моделі, наведеної на рис. 27, тільки вмістом блоку «СУО», блок-схема якого для цього випадку наведена на рис. 10.35.

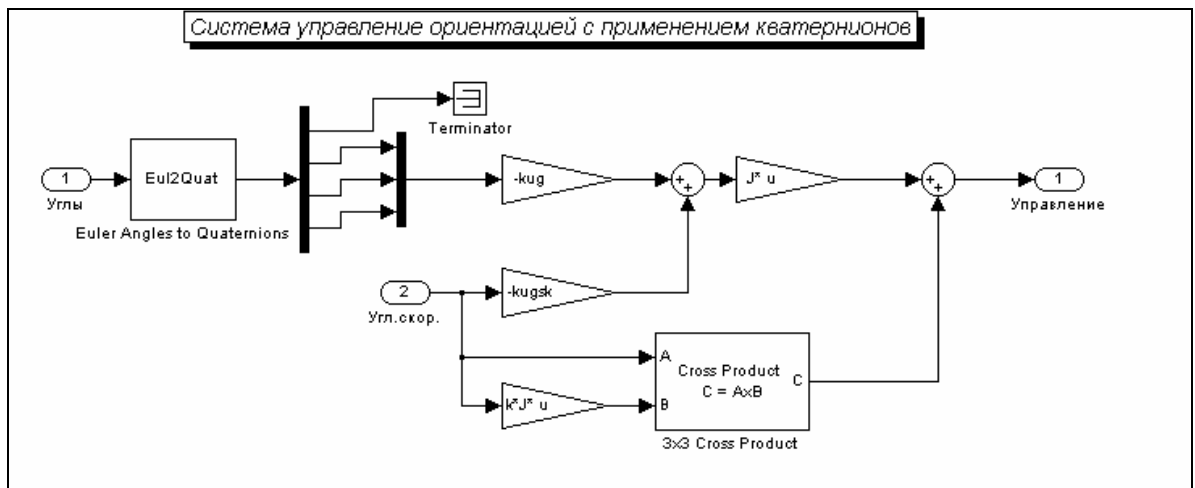


Рис. 10.35. Блок-схема підсистеми «СУО» для керування по кватерніону

Керування роботою моделі і виведенням результатів здійснюється програмою UprDvigKAqw1_upr, текст якої наведено нижче.

```
% UprDvigKAqw1_upr
% Управляющая программа для модели UprDvigKA

% Лазарев Ю.Ф. 14-05-2004

clear all
clc
% Установка параметров КА
%J=[3100 0 0;0 2200 0;0 0 2200]; % Матрица моментов инерции КА
%J=[3100 0 0;0 2200 0;0 0 1200]; % Матрица моментов инерции КА
J=[3100 300 -200;300 2200 100;-200 100 1200]; % Матрица моментов инерции КА
m=2000; % Масса КА
% Задание коэффициентов управления
kug=0.1; % К-нт обратной связи по углам
kugsk=0.3; % К-нт обратной связи по угловым скоростям
k=0; % К-нт компенсации гироскопического момента
% Установка начальных условий
XYZ0=[0 0 0]; % Начальное положение КА
V0=[0 0 0]; % Начальные скорости КА
UG0=[0 1 1]; % Начальные углы КА
```

```

UgSk0=[0 0 0]; % Начальные угловые скорости КА
% Установка параметров интегрирования
TK=100; % Конечное время интегрирования
hi=0.1; % Шаг интегрирования
% Запуск модели
sim('UprDvigKAqw');
% Запись результатов интегрирования
Fl=yout(:,1); TE=yout(:,2); PSI=yout(:,3);
omx=yout(:,4); omy=yout(:,5); omz=yout(:,6);
t=tout;
L=yout(:,7); M=yout(:,8); N=yout(:,9);
% Графическое представление результатов
subplot(2,2,1)
plot(-PSI,TE,-PSI(1),TE(1),'pr',-PSI(end),TE(end),'kv'), grid
axis('equal');
set(gca,'FontSize',12)
title('Движение оси Xb в пространстве');
ylabel('Угол \theta (градусы)');
xlabel('Угол \psi (градусы)');
subplot(2,2,3)
plot(t,omx,t,omy,'t,omz','--'), grid%plot(t,L,t,M,'t,N','--'), grid
set(gca,'FontSize',12)
title('Проекция угловой скорости');%title('Моменты сил');
xlabel('Время (с)');
ylabel('Радьян в сек.');//ylabel('Моменты (Н м)');
legend(' omx ',' omy ',' omz ',0);%legend(' L ',' M ',' N ',0);
subplot(2,2,4)
plot(t,L,t,M,'t,N','--'), grid
set(gca,'FontSize',12)
title('Моменты сил');
xlabel('Время (с)');
ylabel('Моменты (Н м)');
legend(' L ',' M ',' N ',0);
subplot(2,2,2)
axis('off');
h=text(-0.3,1.1,'Управляемое по кватернионам угловое движение КА','FontSize',14);
h=text(0.1,0.9,'| ','FontSize',12);
h=text(0.2,0.9,num2str(J(1,1)),'FontSize',12);
h=text(0.4,0.9,num2str(J(1,2)),'FontSize',12);
h=text(0.6,0.9,num2str(J(1,3)),'FontSize',12);
h=text(0.8,0.9,'| ','FontSize',12);
h=text(-0.1,0.8,'J = ','FontSize',12);
h=text(0.1,0.8,'| ','FontSize',12);
h=text(0.2,0.8,num2str(J(2,1)),'FontSize',12);
h=text(0.4,0.8,num2str(J(2,2)),'FontSize',12);
h=text(0.6,0.8,num2str(J(2,3)),'FontSize',12);
h=text(0.8,0.8,'| ','FontSize',12);
h=text(0.1,0.7,'| ','FontSize',12);
h=text(0.2,0.7,num2str(J(3,1)),'FontSize',12);
h=text(0.4,0.7,num2str(J(3,2)),'FontSize',12);
h=text(0.6,0.7,num2str(J(3,3)),'FontSize',12);
h=text(0.8,0.7,'| ','FontSize',12);
h=text(-0.1,0.6,'Начальные углы (градусы)','FontSize',12);
h=text(0.1,0.5,['\psi0 = ',num2str(UG0(3)*180/pi)],'FontSize',12);
h=text(0.4,0.5,['\theta0 = ',num2str(UG0(2)*180/pi)],'FontSize',12);
h=text(0.7,0.5,['\phi0 = ',num2str(UG0(1)*180/pi)],'FontSize',12);
h=text(-0.1,0.4,'Начальные угловые скорости (рад/с)','FontSize',12);
h=text(0.1,0.3,['omx0 = ',num2str(UgSk0(1))],'FontSize',12);
h=text(0.4,0.3,['omy0 = ',num2str(UgSk0(2))],'FontSize',12);
h=text(0.7,0.3,['omz0 = ',num2str(UgSk0(3))],'FontSize',12);
h=text(-0.1,0.2,'Управление Mupr = - kug*J*Qwat - kugsk*J*om + k*(om*)*J*om','FontSize',12);
h=text(0.1,0.1,['kug = ',num2str(kug)],'FontSize',12);
h=text(0.4,0.1,['kugsk = ',num2str(kugsk)],'FontSize',12);
h=text(0.7,0.1,['k = ',num2str(k)],'FontSize',12);
h=text(-0.1,-0.05,'-----');
h=text(-0.1,-0.1,'Программа UprDvigKAqw1-upr Лазарев Ю. Ф. 14-05-2004');
h=text(-0.1,-0.15,'-----');

```

Результати роботи цієї програми подані на рис. 10.36. та 10.37. Коефіцієнти керування прийняті попередніми, але кути відхилення – більшими у 10 разів (по одному радіану). Як і раніше, розглянуті рухи без і з компенсацією гіроскопічного моменту.

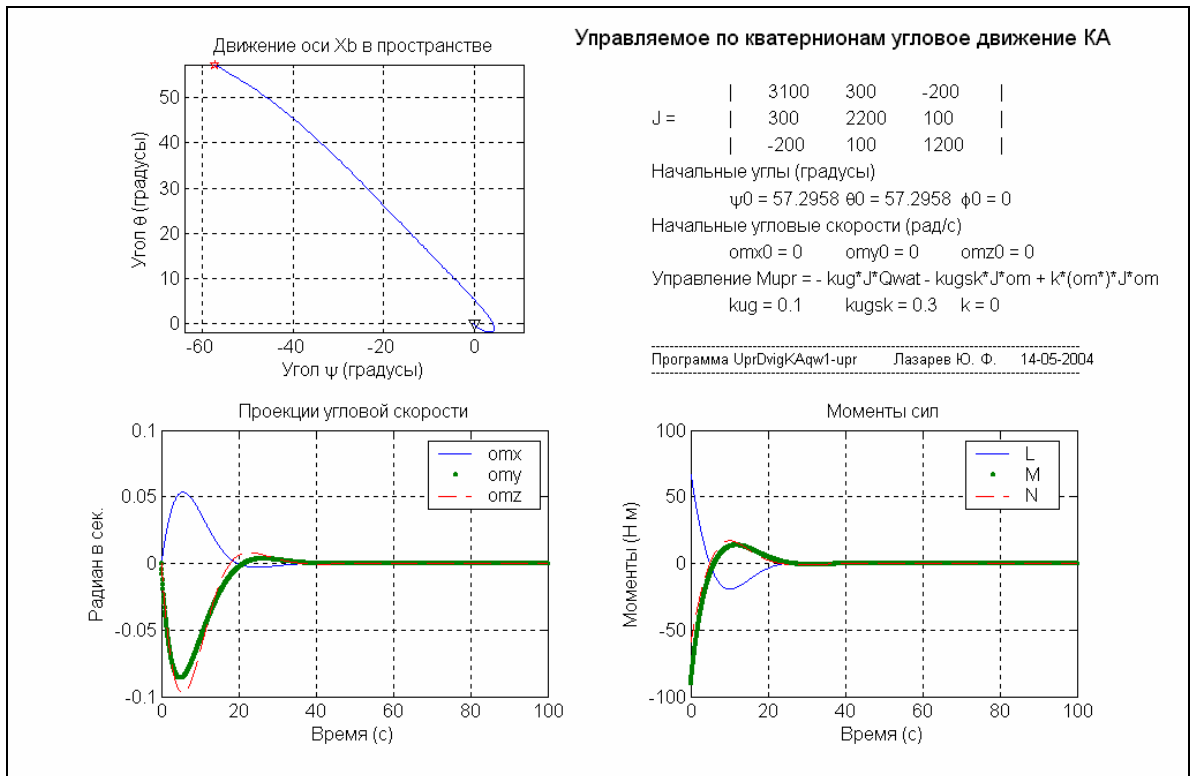


Рис. 10. 36. Керування орієнтацією по кватерніону відхилення

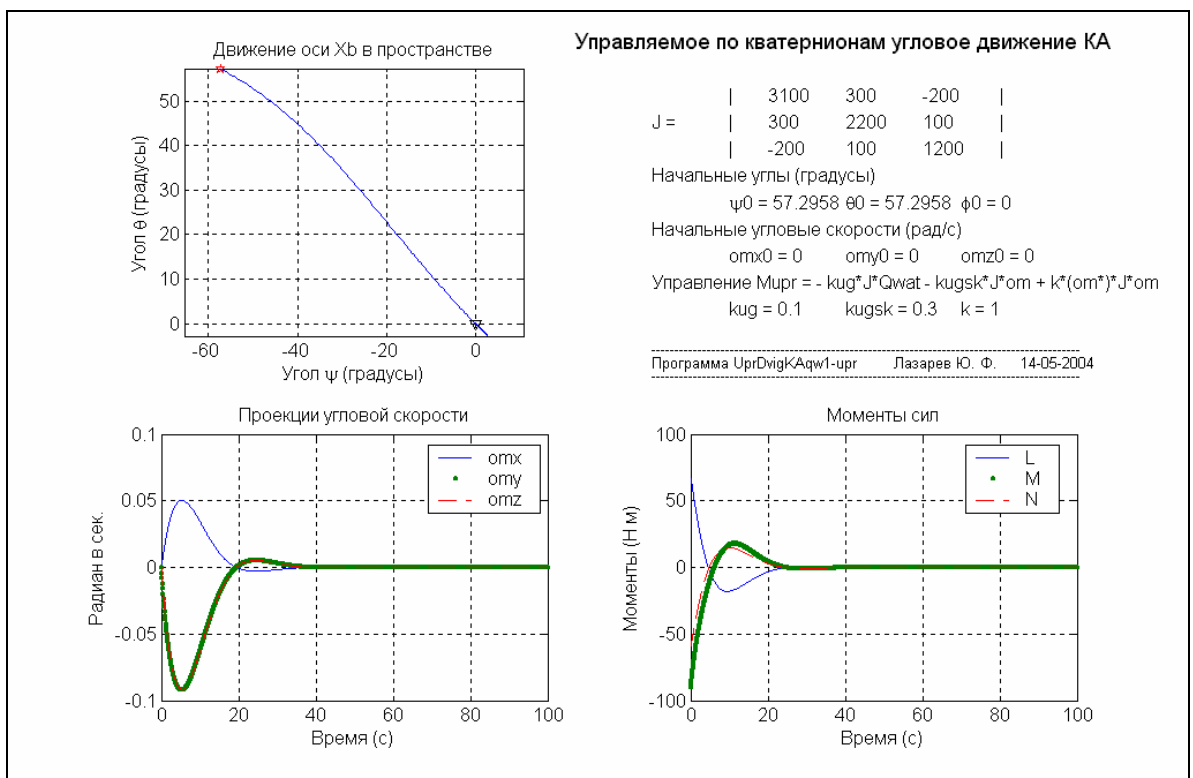


Рис. 10. 37. Керування по кватерніону з компенсацією гіроскопічного моменту

З наведених результатів випливає:

- 1) за керування по кватерніону відхилення рух КА до заданого положення здійснюється за найкоротшим шляхом;

2) процес усталення у задане положення є стійким навіть за дуже великих початкових відхилень;

3) тривалість процесу приведення зберігається такою самою, як й за керування по кутах, хоча величина початкового відхилення збільшення у 10 разів;

4) величини моментів керування значно менше потрібних для керування по кутах;

5) як і раніше, компенсація гіроскопічного моменту приводить до поліпшення динаміки процесу орієнтування.

10.4. Модель руху штучного супутника Землі

Перейдемо до складання моделі орбітального руху штучного супутника Землі (ШСЗ) із врахуванням його кутової стабілізації.

Для цього введемо такі системи координат:

1) інерціальну (нерухому) систему $X_e Y_e Z_e$, початок якої помістимо у центр Землі, вісь Z_e направимо перпендикулярно до площини орбіти, вісь X_e – вздовж лінії, що з'єднує центр Землі з початковим положенням ШСЗ на орбіті, а вісь Y_e – у площині орбіти вздовж вектора початкової швидкості ШСЗ;

2) орбітальну систему $X_o Y_o Z_o$, початок якої помістимо у центр мас ШСЗ, вісь Z_o направимо паралельно осі Z_e , вісь X_o – вздовж лінії, що з'єднує центр Землі із поточним положенням ШСЗ на орбіті, а вісь Y_o – у площині орбіти вперед по руху ШСЗ по орбіті;

3) зв'язану з тілом ШСЗ систему $X_b Y_b Z_b$, початок якої помістимо також у центр мас ШСЗ.

ШСЗ як об'єкт керування має шість ступенів вільності – три координати положення центра мас в інерціальній системі координат та три кутові ступеня вільності, які визначають кутове положення тіла супутника по відношенню до інерціальної системи. Керування рухом ШСЗ складається з двох тісно пов'язаних процесів – керування рухом центра мас ШСЗ та керування його кутовим положенням. Другий з них був розглянутий і промодельований раніше. Для керування орбітальним рухом ШСЗ важливо попередньо орієнтувати і стабілізувати його кутове положення в орбітальній системі координат (див. вище).

Можливий варіант `UprDigISZqw1.mdl` моделі повного руху супутника наведений на рис. 10.38.

Основною відмінністю цієї моделі від раніше розглянутих є наявність зворотного зв'язку (керування) по вектору сили, що діє на центр мас ШСЗ. Він реалізований у вигляді підсистеми «П/С СИЛА», блок-схему якої показано на рис. 10.39.

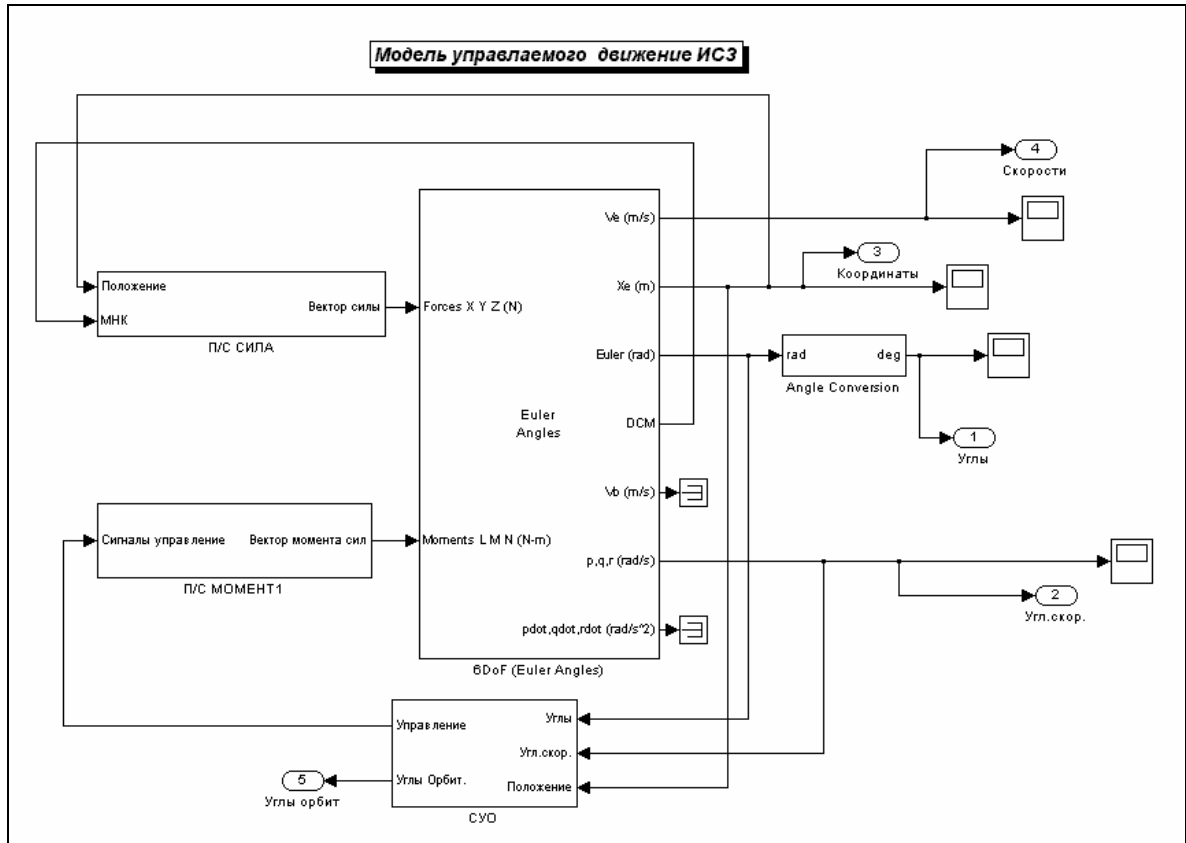


Рис. 10. 38. Блок-схема моделі керованого руху ШСЗ

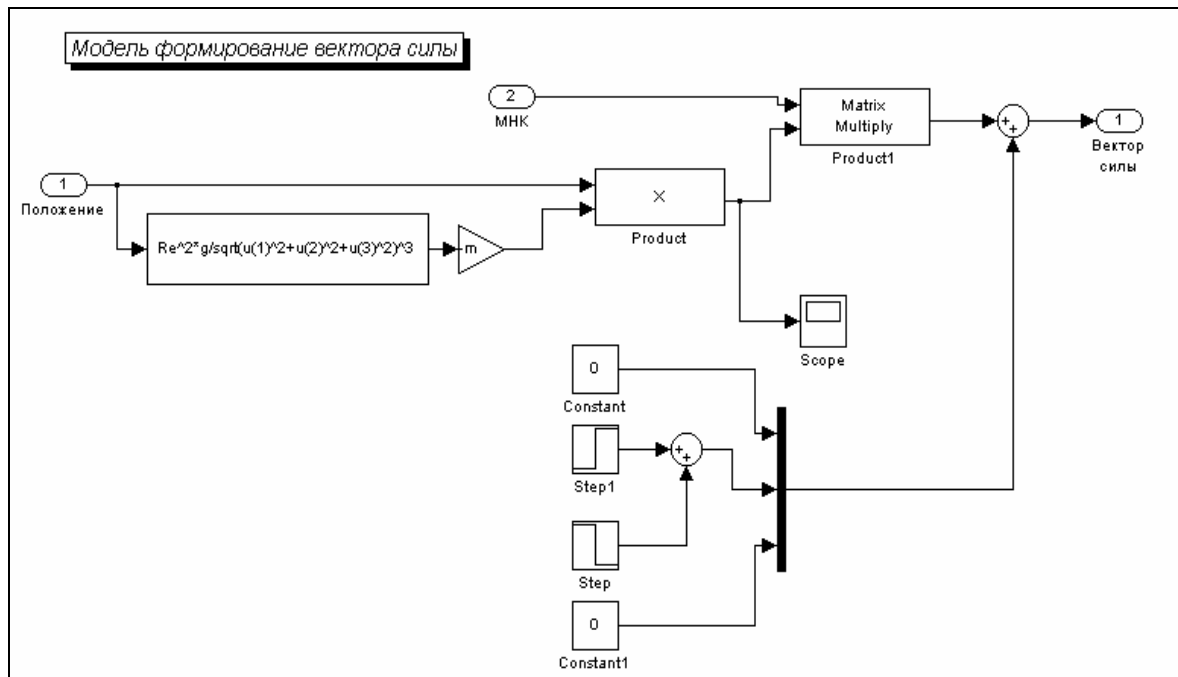


Рис. 10. 39. Блок-схема блока «П/С СИЛА»

Блок-схему підсистеми «СУО» (рис. 10.40) також змінено, з огляду на те, що ШСЗ необхідно орієнтувати не в інерціальній, а в орбітальній системі координат.

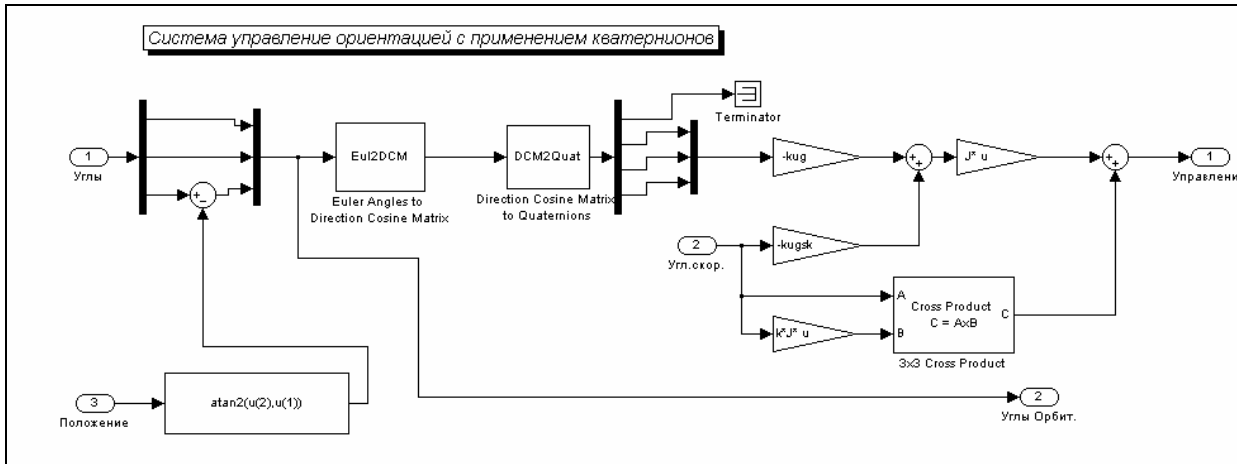


Рис. 10.40. Блок-схема блоку «СУО»

Основним призначенням блоку «П/С СИЛА» є формування вектора сили гравітації, що діє на ШСЗ з боку Землі. Для цього спочатку потрібно визначити величину прискорення гравітації у тій точці простору, де міститься супутник. Це прискорення визначається виразом:

$$g_c = g \left(\frac{R}{r} \right)^2,$$

де g – прискорення гравітації на поверхні Землі, g_c – прискорення у точці, що міститься на відстані r від центра Землі, а R – радіус Землі. Тепер вектор \mathbf{F} сили гравітації, що діє на ШСЗ можна визначити з співвідношення:

$$\mathbf{F} = -mg_c \frac{1}{r} \mathbf{r}.$$

Тут m – маса ШСЗ, а \mathbf{r} – радіус-вектор цього центра мас.

Якщо відомі поточні координати центра мас X_e, Y_e, Z_e супутника в інерціальній системі, то

$$r = \sqrt{X_e^2 + Y_e^2 + Z_e^2},$$

і обчислення проєкцій вектора \mathbf{F} в інерціальній системі можна звести до формули

$$\mathbf{F} = -mg_c = -mg \frac{R^2}{(\sqrt{X_e^2 + Y_e^2 + Z_e^2})^3} \mathbf{r}.$$

Виконання саме цих обчислень і забезпечено у блоці «П/С СИЛА» (див. рис. 10.39). Через те, що до входу блоку 6DoF потрібно подати вектор сили через його проєкції на осі, зв'язані з ШСЗ, то далі отриманий вектор сили в інерціальній системі перетворюється у вектор проєкцій цієї сили на зв'язану систему координат через множення на матрицю напрямних косинусів ШСЗ в інерціальній системі.

Друге завдання блоку – сформулювати силу активної дії вздовж осі Y_b зв'язаної системи для здійснення переходу на іншу орбіту. Ця мета досягається нижньою гілкою у блок-схемі рис. 10.39. Тут формується незмінна за величиною

сила, яка починає свою дію у заданий момент часу і діє заданий проміжок часу. При цьому використані такі позначення:

Tn	Початковий момент часу дії сили корегування орбіти
tau	Проміжок часу, протягом якого діє сила корекції
DF	Величина сили корекції

Блок «СУО» в представленій моделі (рис. 10.40) відрізняється від аналогічного блоку у попередній моделі (рис. 10.35), перш за все, наявністю частини, яка обчислює кутове відхилення зв'язаної з ШСЗ системи координат від орбітальної. Це є необхідним для забезпечення керуванням орієнтацією супутника відносно не інерціальної, а орбітальної системи. Крім того, замість блоку безпосереднього перетворення кутів Ейлера у кватерніон повороту використовується послідовне перетворення кутів Ейлера у матрицю напрямних косинусів, а потім останньої у кватерніон повороту. Це дозволяє позбутися розривів у кутових координатах при переході значень кутів через 180° .

Текст керувальної програми *UprDvigISZqw1_upr* наведений нижче.

```
% UprDvigISZqw1_upr
% Управляющая программа для модели SvDvigKA

% Лазарев Ю.Ф. 15-05-2004

clear all, clc
OM=7.292115e-5; Re=6.37814e6; g=9.78;
% Установка параметров КА
%J=[1800 0 0;0 3400 0;0 0 1800]; % Матрица моментов инерции КА
%J=[3400 0 0;0 2200 0;0 0 1400]; % Матрица моментов инерции КА
J=[2200 300 -200;300 3400 100;-200 100 1400]; % Матрица моментов инерции КА
m=2000; % Масса КА
% Установка начальных условий
XYZ0=[6.4e6 0 0]; % Начальное положение КА
V0=[0 8e3 0]; % Начальные скорости КА
UG0=[0 1 0]; % Начальные углы КА
UgSk0=[0 0.2 0]; % Начальные угловые скорости КА
% Задание коэффициентов управления
kug=0.1; % К-нт обратной связи по углам
kugsk=0.3; % К-нт обратной связи по угловым скоростям
k=1; % К-нт компенсации гироскопического момента
Tn=3600*2; tau=1000; DF=2000;
% Установка параметров интегрирования
TK=6*3600; % Конечное время интегрирования
hi=0.5; % Шаг интегрирования
% Запуск модели
sim('UprDvigISZqw1');
% Запись результатов интегрирования
Fi=yout(:,1); TE=yout(:,2); PSI=yout(:,3);
omx=yout(:,4); omu=yout(:,5); omz=yout(:,6);
Xe=yout(:,7); Ye=yout(:,8); Ze=yout(:,9);
Vex=yout(:,10); Vey=yout(:,11); Vez=yout(:,12);
Flo=yout(:,13); TEo=yout(:,14); PSIo=unwrap(yout(:,15));
t=tout; n=length(t);
k1=0;
for ks=1:n
    if t(ks)<=Tn
        k1=k1+1; t1(k1)=t(ks);
    end
end
K1=k1; Xe1=Xe(1:K1); Ye1=Ye(1:K1); k1=0;
for ks=1:n
    if (t(ks)>Tn)&(t(ks)<=Tn+tau)
        k1=k1+1; t2(k1)=t(ks);
    end
end
end
```

```

K2=k1; Xe2=Xe(K1+1:K1+K2); Ye2=Ye(K1+1:K1+K2); k1=0;
for ks=1:n
    if t(ks)>Tn+tau
        k1=k1+1; t3(k1)=t(ks);
    end
end
K3=k1; Xe3=Xe(K1+K2+1:K1+K2+K3); Ye3=Ye(K1+K2+1:K1+K2+K3);
% Графическое представление результатов
subplot(2,2,1)
plot(Xe1*1e-6,Ye1*1e-6,'--',Xe3*1e-6,Ye3*1e-6,'.',...
      Xe2*1e-6,Ye2*1e-6,'*',0,0,'o',XYZ0(1)*1e-6,XYZ0(2)*1e-6,'pk'), grid
axis('equal');
title('Движение ИСЗ в плоскости X - Y');
xlabel('Координата X (тыс. км)'); ylabel('Координата Y (тыс. км)');
legend(' орбита 1 ',' орбита 2 ',' активный ',0);
subplot(2,2,3)
plot(t1/3600,Xe1*1e-6,'k--',t2/3600,Xe2*1e-6,'k*',t3/3600,Xe3*1e-6,'k.',...
      t1/3600,Ye1*1e-6,'b--',t2/3600,Ye2*1e-6,'b*',t3/3600,Ye3*1e-6,'b.'), grid
title('Инерциальные координаты');
xlabel('Время (часы)'); ylabel('Координаты (тыс. км)');
legend(' орбита 1 ',' активный ',' орбита 2 ',0);
subplot(2,2,4)
plot(t(1:80),PSI(1:80),'*',t(1:80),TE(1:80),t(1:80),PSIo(1:80)*180/pi,'.',...
      t(1:80),TEo(1:80)*180/pi), grid
title('Процесс угловой стабилизации'); xlabel('Время (с)'); ylabel('Углы (градусы)');
legend(' \psi ',' \theta ',' \psi_0 ',' \theta_0 ',0);
subplot(2,2,2)
axis('off');
h=text(0.0,1.1,'Управляемое движение ИСЗ','FontSize',14);
h=text(0.3,0.95,' '); h=text(0.4,0.95,num2str(J(1,1)));
h=text(0.6,0.95,num2str(J(1,2))); h=text(0.8,0.95,num2str(J(1,3)));
h=text(1.0,0.95,' '); h=text(0.1,0.9,'J = ');
h=text(-0.3,0.9,['m = ',num2str(m)]);
h=text(0.3,0.9,' '); h=text(0.4,0.9,num2str(J(2,1)));
h=text(0.6,0.9,num2str(J(2,2))); h=text(0.8,0.9,num2str(J(2,3)));
h=text(1.0,0.9,' '); h=text(0.3,0.85,' ');
h=text(0.4,0.85,num2str(J(3,1))); h=text(0.6,0.85,num2str(J(3,2)));
h=text(0.8,0.85,num2str(J(3,3))); h=text(1.0,0.85,' ');
h=text(-0.1,0.75,'Начальное положение:'); h=text(-0.3,0.7,['Xe0 = ',num2str(XYZ0(1)*1e-6)];
h=text(0.0,0.7,['Ye0 = ',num2str(XYZ0(2)*1e-6)];
h=text(0.3,0.7,['Ze0 = ',num2str(XYZ0(3)*1e-6)]; h=text(0.7,0.7,'(тыс. км) ');
h=text(-0.3,0.65,['\psi_0 = ',num2str(UG0(3)*180/pi)];
h=text(0.0,0.65,['\theta_0 = ',num2str(UG0(2)*180/pi)];
h=text(0.3,0.65,['\phi_0 = ',num2str(UG0(1)*180/pi)]; h=text(0.7,0.65,'(градусы) ');
h=text(-0.1,0.55,'Начальные скорости:');
h=text(-0.3,0.5,['Vex0 = ',num2str(V0(1)*1e-3)]; h=text(0.0,0.5,['Vey0 = ',num2str(V0(2)*1e-3)];
h=text(0.3,0.5,['Vez0 = ',num2str(V0(3)*1e-3)]; h=text(0.7,0.5,'(км/с) ');
h=text(-0.3,0.45,['omx0 = ',num2str(UgSk0(1))]; h=text(0.0,0.45,['omy0 = ',num2str(UgSk0(2))];
h=text(0.3,0.45,['omz0 = ',num2str(UgSk0(3))]; h=text(0.7,0.45,'(рад/с) ');
h=text(-0.1,0.35,'К-нты управления ориентацией:'); h=text(-0.3,0.3,['kug = ',num2str(kug)];
h=text(0.0,0.3,['kugsk = ',num2str(kugsk)]; h=text(0.3,0.3,['k = ',num2str(k)];
h=text(-0.1,0.2,['Начало активного участка Tn = ',num2str(Tn),' сек'];
h=text(-0.1,0.1,['Длительность активного участка tau = ',num2str(tau),' сек'];
h=text(-0.1,0.0,['Тяга F = ',num2str(DF),' Н'];
h=text(-0.1,-0.05,'-----');
h=text(-0.1,-0.1,'Программа UprDvigISZqw1-upr Лазарев Ю. Ф. 15-05-2004');
h=text(-0.1,-0.15,'-----');

```

На рис. 10.41 відображені результати роботи програми і моделі.

Отримані результати підтверджують достатньо добрі властивості моделі, які дозволяють досліджувати доволі складні процеси керування рухом ШСЗ.

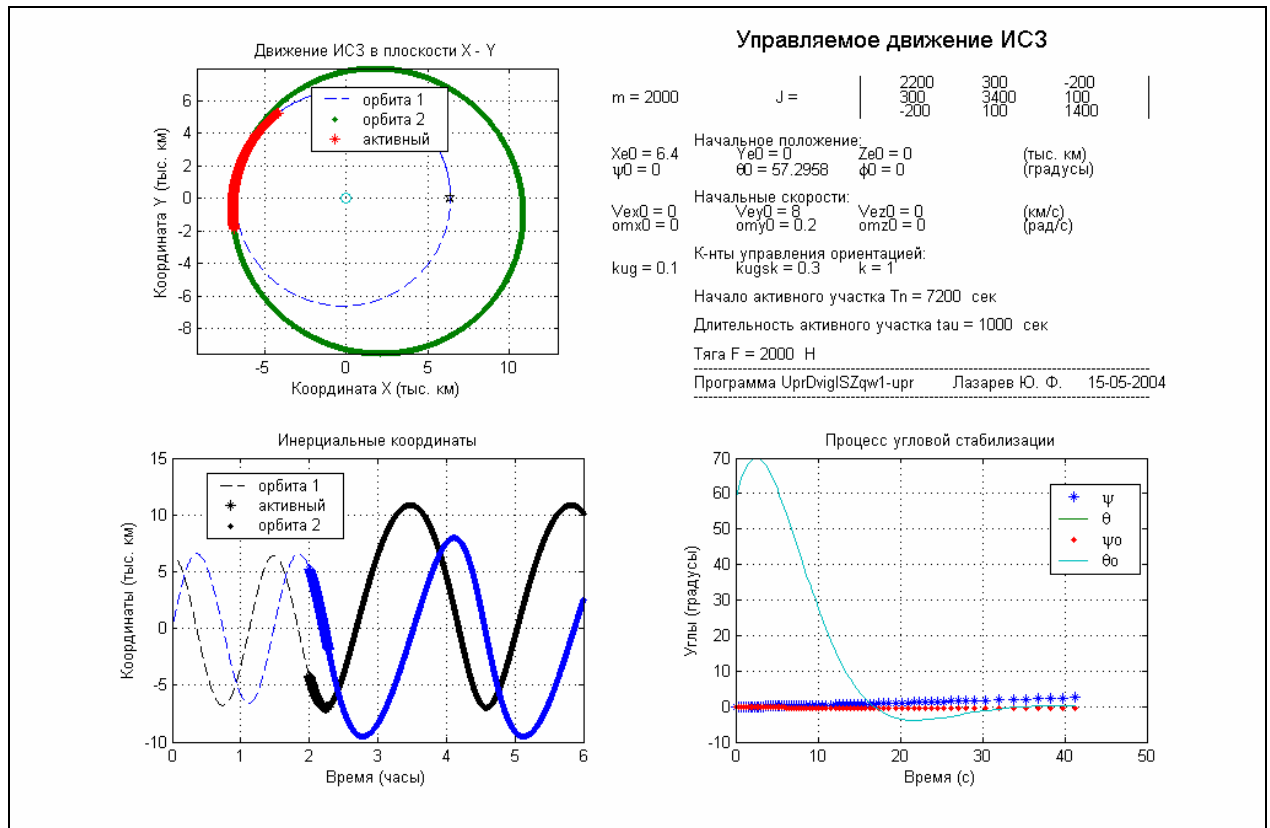


Рис. 10.41. Маневрирование ИСЗ

10.5. Запитання для самоперевірки

1. Яке головне призначення бібліотеки *Aerospace*? З яких розділів вона складається?
2. Яке головне призначення блоків розділу *Equations of Motion*? у чому полягає різниця між блоками цього розділу?
3. Для чого призначені блоки розділу *Environment*?
4. Какиє групи силових впливів визначають рух літального апарату?
5. Для чого прислужуються блоки розділу *Propulsion*?
6. Яке головне призначення блоків розділів *Actuators* і *GNC*?
7. Які завдання вирішують блоки розділу *Transformations*?

11. ЗАГАЛЬНА ХАРАКТЕРИСТИКА БІБЛІОТЕКИ SIMPOWERSYSTEMS

Бібліотеку блоків *SimPowerSystems* призначено для моделювання поведінки електричних силових систем, які являють собою комбінації електричних ланцюгів і електромеханічних пристроїв. Бібліотека функціонує у складі пакету *Simulink* у середовищі Matlab і містить моделі типових пристроїв силової електроенергетики, такі як трансформатори, перетворювачі, лінії електропередач, електромашини, електровимірювачі, елементи силової електроніки.

Ідеологія складання блок-схеми моделі у бібліотеці *SimPowerSystems* суттєво відрізняється від ідеології складання блок-схем в S-моделях. В S-моделях блоки, які з'єднуються між собою, являють собою програми математичного перетворення вхідних величин блоку у вихідні величини незалежно від їх фізичного змісту. У блок-схемах *SimPowerSystems* з'єднання блоків цієї бібліотеки зазвичай слід розглядати як імітацію електричних з'єднань, лінії з'єднань – як дротовий зв'язок, що здійснює передавання електричного сигналу (струму) від вихода одного блоку до входу наступного блоку, а самі блоки бібліотеки – як моделі електричних процесів, що течуть у пристрої, поведінка якого моделюється. З цього випливає, що блоки S-моделі, у загальному випадку, не можуть з'єднуватися з більшістю блоків бібліотеки *SimPowerSystems*. Зокрема, неможливо безпосередньо використовувати блоки бібліотеки *Simulink* для формування електричних сигналів заданої форми, неможливо безпосередньо вивести значення струмів і напруг в оглядові вікна Score і на вихідні порти і т. п. Тим не менш, моделі бібліотеки *SimPowerSystems* функціонують у середовищі *Simulink*, і тому їм мають бути досяжні усі можливості цього середовища, що надаються бібліотекою *Simulink*.

Надалі задля спрощення використовуватимемо таку термінологію. Будем називати S-блоками блоки бібліотеки *Simulink*, P-блоками – блоки бібліотеки *SimPowerSystems*, лінії, що з'єднують S-блоки, – m-лініями, входи і виходи S-блоків – m-входами і m-виходами, лінії, що з'єднують P-блоки – r-лініями, входи і виходи P-блоків – r-входами і r-виходами. Нагадаємо, що m-лінії переносять сигнали-функції, незалежно від їх фізичної природи, а r-лінії – електричні сигнали і є аналогом ідеального дротового зв'язку. Аналогічно, m-входи і m-виходи сприймають або генерують функціональні сигнали, а r-входи і r-виходи реалізують електричне з'єднання провідника r-лінії з відповідним P-блоком.

Для зв'язування P-блоків зі звичайними S-блоками призначені лише деякі блоки бібліотеки *SimPowerSystems*, такі як блоки-вимірювачі електричних сигналів (струму, напруги і т. п.), які мають r-входи і m-виход, деякі блоки джерел електрики (вони мають m-вхід і r-виходи), а також деякі інші блоки. Саме ці блоки дозволяють використовувати усі багатющі можливості бібліотеки

Simulink для моделювання електроенергетичних систем і, зокрема, можливості програмування у Matlab процесів введення, перетворення і виведення (у том числі – у графічній формі) інформації.

В браузері **Simulink** за допомогою контекстного меню бібліотеки **SimPowerSystems** викликається вікно **powerlib2** цієї бібліотеки (рис. 11.1).

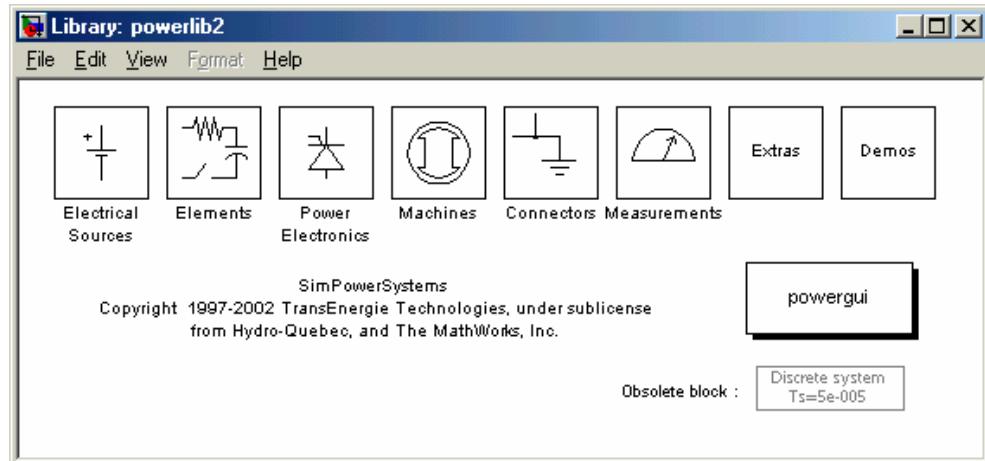


Рис. 11. 1. Склад бібліотеки *powerlib2*

У бібліотеку **SimPowerSystems** входять дев'ять розділів:

Electrical Sources

(Джерела електрики)

Elements

(Елементи електричних ланцюгів)

Power Electronics

(Елементи силової електроніки)

Machines

(Машини)

Connectors

(З'єднувачи)

Measurements

(Вимірювальні елементи)

Extras

Demos

Powergui

містить блоки, що моделюють джерела струму і напруги

містить блоки, що моделюють електричні процеси в елементарних електричних елементах (RLC-ланцюги, реальний дрововий зв'язок, електричне навантаження, трансформатори і т. п.)

містить блоки, що моделюють поведінку комутувальних елементів силової електроніки (діоди, тиристори, мости, керовані ключі і т. п.)

містить блоки – моделі електричних машин різних типів

Складається з блоків, що моделюють електричні з'єднання різних типів

Містить блоки, що імітують вимірювальні електричні прилади (амперметри, вольтметри, омметри і т. п.)

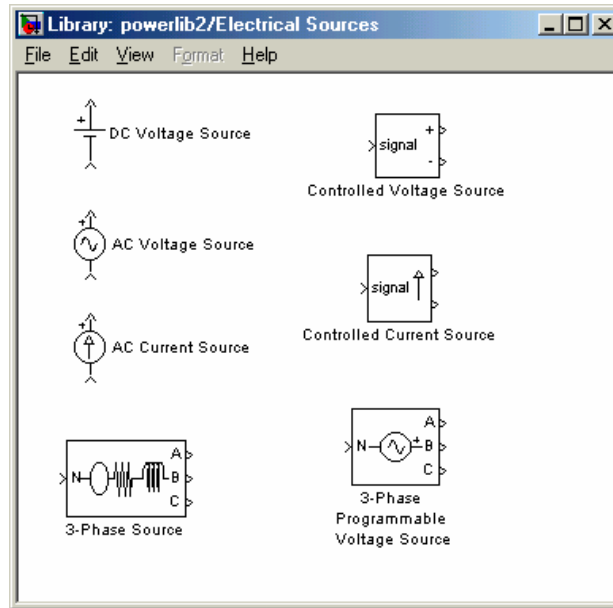
Бібліотека додаткових блоків

Демонстраційні програми-моделі

Інтерактивний блок вікна графічного інтерфейсу користувача

11.1. Розділ *Electrical Sources*

Вміст розділу *Electrical Sources* (Джерела електрики) поданий на рис. 11.2.

Рис. 11. 2. Вміст розділу *Electrical Sources*

В нього входять 7 блоків.

DC Voltage Source

AC Voltage Source

AC Current Source

Controlled Voltage Source

Controlled Current Source

3-Phase Source

3-Phase Programmable Voltage Source

Джерело постійної напруги

Джерело змінної (сіносоїдальної) напруги

Джерело змінного (сіносоїдального) струму

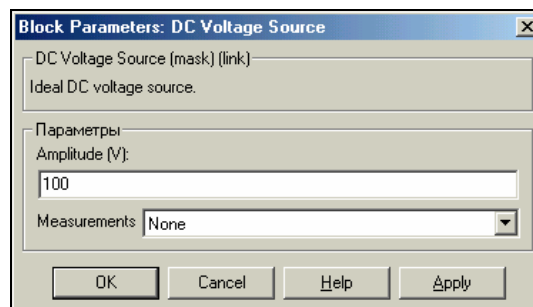
Джерело керованої напруги

Джерело керованого струму

Трифазне джерело

Трифазне програмоване джерело напруги

Блок ***DC Voltage Source*** імітує роботу ідеального джерела постійної напруги. Єдиний параметр настроювання (рис. 11.3) – величина напруги на клеммах джерела.

Рис. 11. 3. Вікно настроювання блоку *DC Voltage Source*

Блоки ***AC Voltage Source*** и ***AC Current Source*** імітують роботу джерел сіносоїдальної напруги і струму відповідно. Вікна настроювання блоків (рис. 11.4 та 11.5) майже не відрізняються.

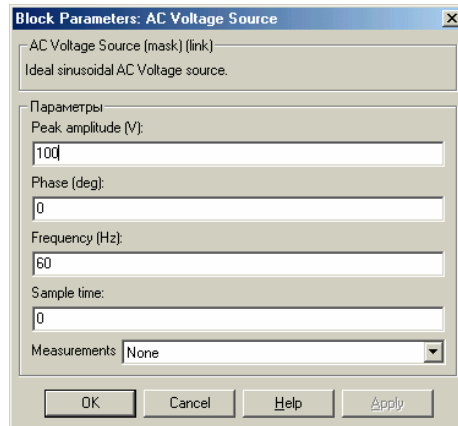


Рис. 11. 4. Вікно настроювання блоку AC Voltage Source

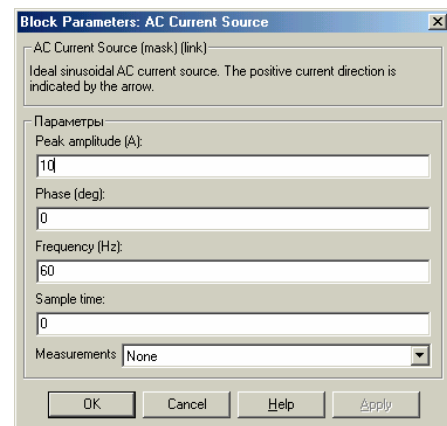


Рис. 11. 5. Вікно настроювання блоку AC Current Source

До параметрів настроювання відносяться такі величини:

Pick amplitude (V)

Амплітуда синусоїдальної напруги у вольтах

Pick amplitude (A)

Амплітуда синусоїдального струму в амперах

Phase (deg)

Початкова фаза в градусах

Frequency (Hz)

Частота змінювання струму (напруги) у герцах

Вхід і вихід обох блоків – це клеми під'єднання джерел до елементів електричної схеми.

Два блоки *Controlled Voltage Source* (Кероване джерело напруги) та *Controlled Current Source* (Кероване джерело струму) дозволяють утворювати джерело з довільним законом змінювання напруги або струму у часі, використовуючи блоки розділу *Sources* бібліотеки *SIMULINK*. На відміну від розглянутих раніше блоків обидва блоки мають m-вхід, до якого можуть бути під'єднані виходи будь-якого S-блоку. На рис. 11.6 та 11.7 наведені вікна настроювання цих блоків. Вони цілком ідентичні.

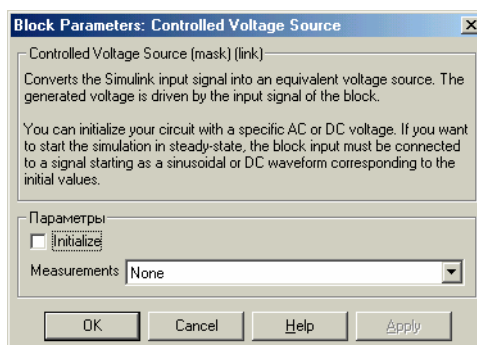


Рис. 11. 6. Вікно настроювання блоку Controlled Voltage Source

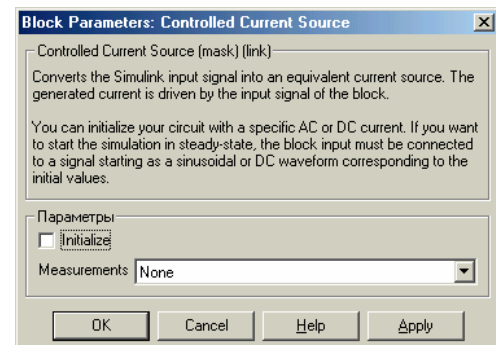


Рис. 11. 7. Вікно настроювання блоку Controlled Current Source

Блоки виробляють на вихідних клеммах напругу або струм, рівний тому S-сигналу, що надається до його входу.

На рис. 11.8 наведено схему моделювання трьох видів джерел напруги.

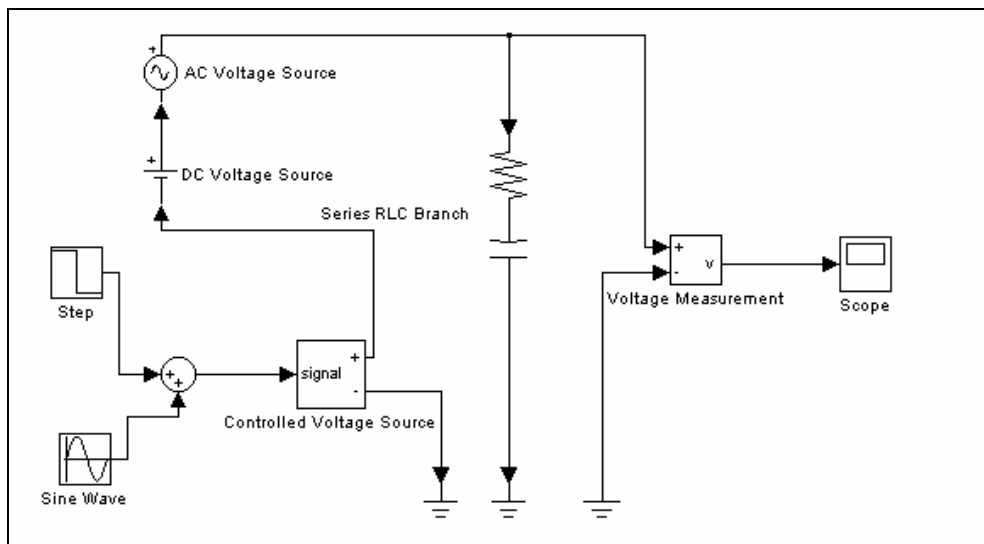


Рис. 10. 8. Схема моделювання роботи трьох видів джерел напруги

На схемі послідовно з'єднані три блоки джерел напруги:

- блок **AC Voltage Source**, на якому встановлено амплітуду 100 В і частоту 50 Гц;
- блок **DC Voltage Source**, де встановлено постійну напругу 50 В;
- блок керованого джерела напруги **Controlled Voltage Source**, до входу якого (m-вхід) надається сума двох сигналів – гармонічного з частотою 20 радіан в секунду і амплітудою 50 В і ступінчастого сигнала, рівного 0 до моменту часу 0,1 секунди і -200 В – після цього моменту.

Ці три джерела напруги замкнені на RC-ланцюг ($R=1$ Ом, $C=10^{-6}$ Ф). Загальна напруга вимірюється за допомогою вимірювального блоку **Voltage Measurement** (Вольтметр), вихід якого є m-виходом, до якого під'єднаний блок **Scope**.

Результат моделювання, що відбивається у блоці **Scope**, поданий на рис. 11.9.

Неважко углядити, що сумарна напруга, як і варто було очікувати, дорівнює сумі напруг вказаних трьох джерел.

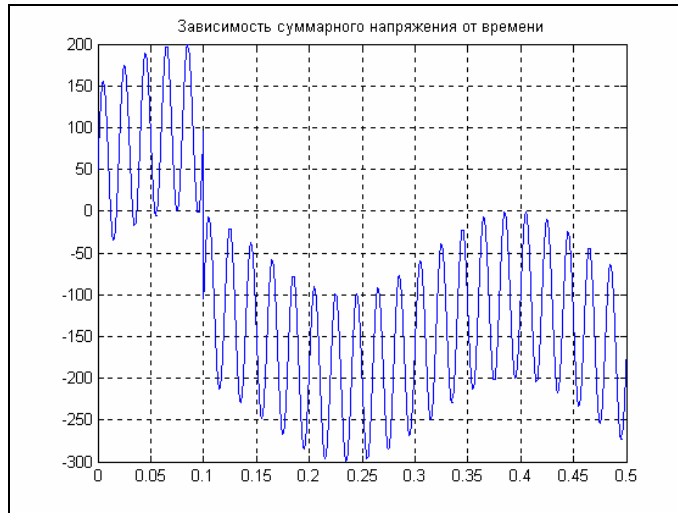


Рис. 11. 9. Результат моделювання за схемою рис. 10. 8

Два останніх блоки (**3-Phase Source** та **3-Phase Programmable Voltage Source**) імітують роботу трифазних джерел напруги. В обох блоках входом є нейтраль (N), а виходів три – фази А, В і С трифазної напруги. Вікна настроювання їх показані на рис. 11.10 та 11.11.

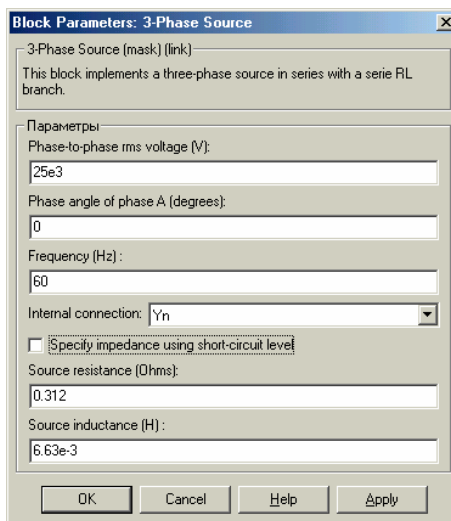


Рис. 11. 10. Вікно настроювання блоку 3-Phase Source

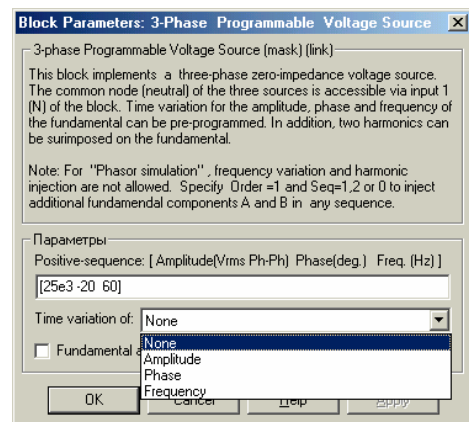


Рис. 11. 11. Вікно настроювання блоку 3-Phase Programmable Voltage Source

Загальними параметрами настроювання є:

Phase to phase voltage
Phase angle of phase A
Frequency

Напруга між фазними клеммами у вольтах
 Початковий фазовий кут фази А у градусах
 Частота змінювання напруги в герцах

У блоці **3-Phase Source** моделюється трифазне джерело зі внутрішнім послідовним RL-ланцюгом, параметри якого задають решта два параметри настроювання

Source resistance
Source inductance

Активний опір джерела в омах
Індуктивність джерела у генрі

У блоці *3-Phase Programmable Voltage Source* імітується джерело з внутрішнім імпедансом, рівним нулю. Блок додатково дозволяє модулювати (амплітудно, частотно або фазно) напругу джерела (див. рис. 11.11). Для цього до параметрів настроювання додається параметр *Time variation of*, який має такий список: *None, Amplitude, Phase, Frequency*, які позначають вид модуляції напруги. Після обрання одного з видів модуляції виникає додатковий список, який дозволяє встановити величини амплітуди (у відповідних одиницях – вольтгах, градусах або герцах), частоти і фазового кута обвідної.

11.2. Розділ Elements

У розділ Elements входять чотири групи блоків (рис. 11.12).

Elements (Елементи)	Містить блоки, які імітують елементарні послідовні і паралельні RLC-ланцюги
Lines (Лінії)	Містить блоки, що імітують лінії електропередач
Circuit Breakers (Переривники)	Містить блоки, що імітують різного роду переривники струму
Transformers (Трансформатори)	Містить блоки, що імітують роботу трансформаторів

Перша група (*Elements*) складається з таких блоків:

Series RLC Load	Послідовний RLC-ланцюг навантаження
Parallel RLC Branch	Паралельний RLC-ланцюг
Parallel RLC Load	Паралельний RLC-ланцюг навантаження
3-Phase Series RLC Branch	Трифазний послідовний RLC-ланцюг
3-Phase Series RLC Load	Трифазний послідовний RLC-ланцюг навантаження
3-Phase Parallel RLC Branch	Трифазний паралельний RLC-ланцюг
3-Phase Parallel RLC Load	Трифазний паралельний RLC-ланцюг навантаження
Mutual Inductance	Блок взаємної індуктивності
3-Phase Mutual Inductance	Блок трифазної взаємної індуктивності
3-Phase Dynamic Load	Трифазне динамічне навантаження
Surge Arrester	Обмежувач пікових напруг

Звичайні RLC-ланцюги (блоки *Series RLC Branch, Parallel RLC Branch, 3-Phase Series RLC Branch* та *3-Phase Parallel RLC Branch*) задаються трьома параметрами – опором R, індуктивністю L та ємністю C. Для введення окремих елементів (резистора, конденсатора або індуктивності) можна використовувати

будь-який з цих блоків, задавши у ньому параметри, відповідні відсутнім елементам. Наприклад, для задання резистора за допомогою послідовного ланцюга (блок **Series RLC Branch**) слід задати $L=0$, а $C=\infty$ (нескінченне значення ємності перетворює конденсатор в ідеальний провідник).

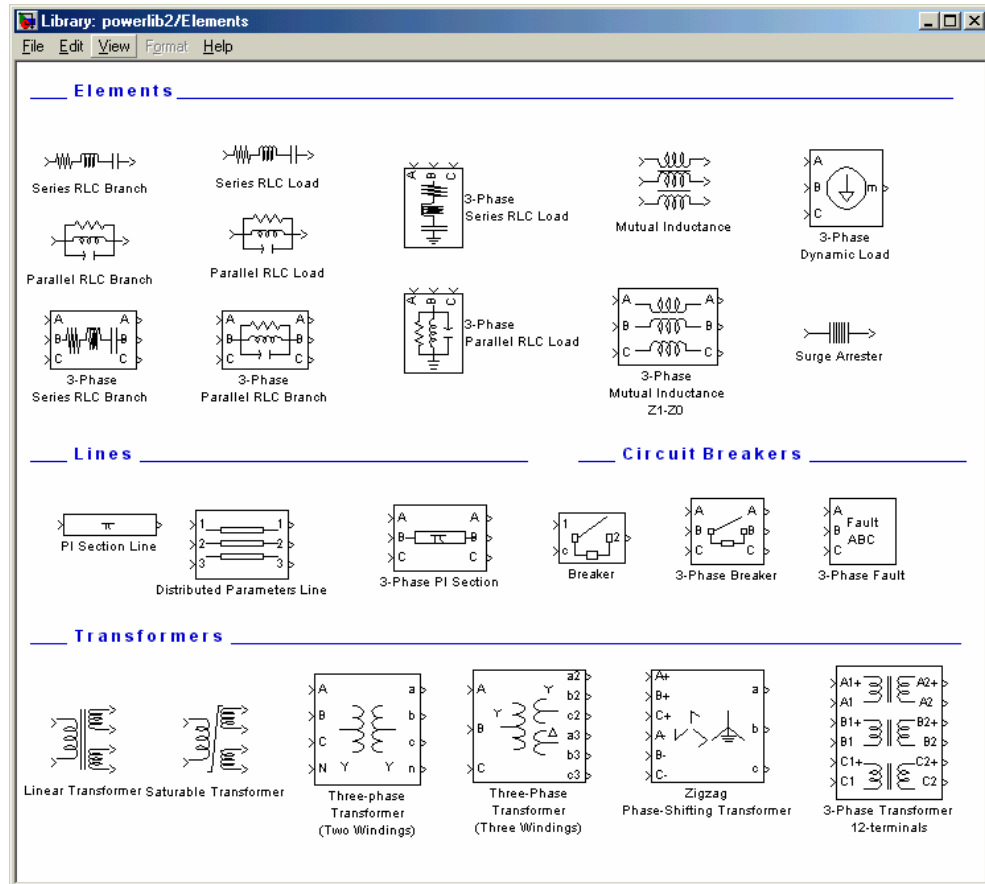


Рис. 11. 12. Вміст розділу Elements

У навантажувальних RLC-ланцюгах (блоки **Series RLC Load**, **Parallel RLC Load**, **3-Phase Series RLC Load** та **3-Phase Parallel RLC Load**) задаються допустимі потужності розсіювання – активна для резистора і реактивна для індуктивності і конденсатора.

У вікні налаштування блоку **Mutual Inductance** (рис. 11.13) передбачені такі параметри налаштування:

- Winding 1 Self Impedance** Задається вектор з двох елементів: перший – опір першої котушки в омах, другий – її індуктивність у генрі
(Власний імпеданс першої котушки)
- Winding 2 Self Impedance** Задається опір другої котушки в омах та її індуктивність у генрі
(Власний імпеданс другої котушки)
- Winding 3 Self** Задається опір третьої котушки в омах та її індуктивність у генрі

Impedance

(Власний імпеданс
третьої котушки)

Mutual Impedance

(Взаємний імпеданс
катушек)

Тут задається вектор з двох елементів: перший – опір в омах, другий – взаємна індуктивність у генрі

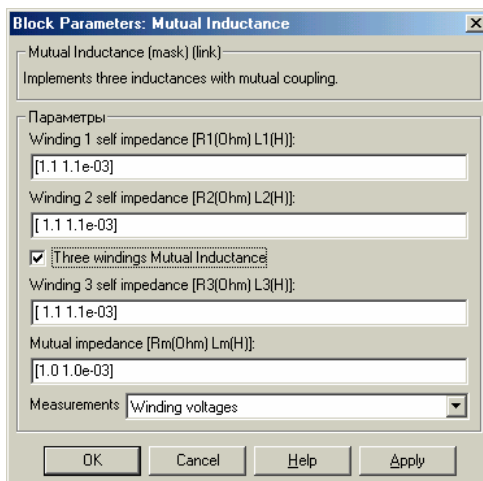


Рис. 11. 13. Вікно налаштування блоку *Mutual Inductance*

Окрім цього, передбачена можливість утворити взаємну індуктивність з двох котушек, для чого в віконці *Tree windings Mutual Inductance* слід скинути прапорець.

У групі блоків *Lines* містяться блоки імітування двох типів ліній електропередачі. Перший тип – лінії ***PI Section Line*** – представляє лінію електропередачі як сукупність послідовних з'єднаних секцій зі зосередженими параметрами. При цьому довжина лінії задається у кілометрах (рис. 11.14), а параметрами лінії є опір, індуктивність та ємність одного кілометра лінії.

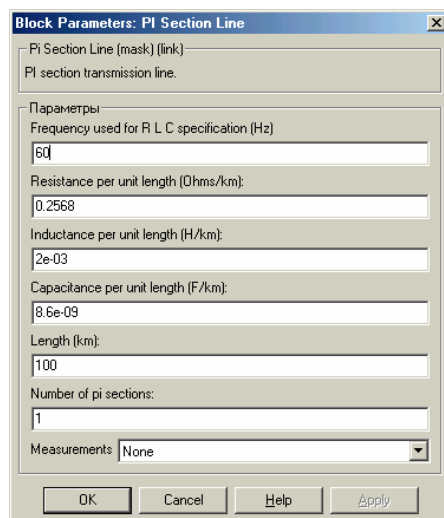


Рис. 11. 14. Вікно налаштування блоку *PI Section Line*

Другий тип лінії, представлений блоком *Distributed Parameters Line* (Лінія з розподіленими параметрами), імітує електричну лінію більш наближеною до реальності системою неперервно розподіленими вздовж довжини лінії електричними параметрами. Параметри настроювання цього блоку (рис. 11.15) в основному збігаються з описаними раніше.

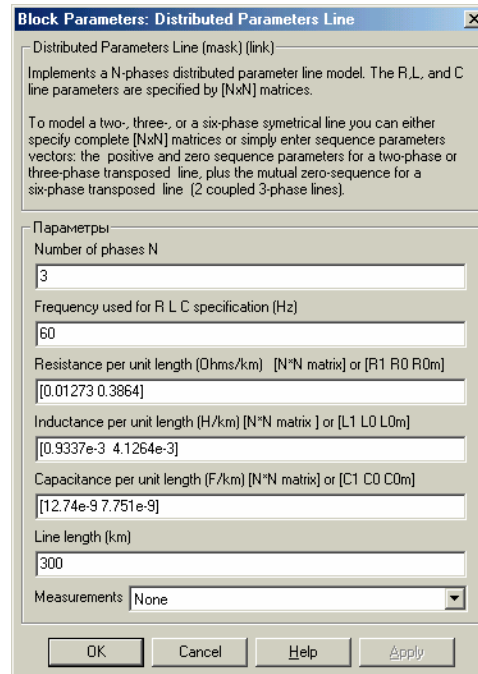


Рис. 11. 15. Вікно настроювання блоку *Distributed Parameters Line*

Хоча за замовчуванням цей блок розрахований на трифазну провідну передачу, кількість ліній можна скоротити до двох або одної, встановлюючи цю кількість у віконці *Number of Phases N*.

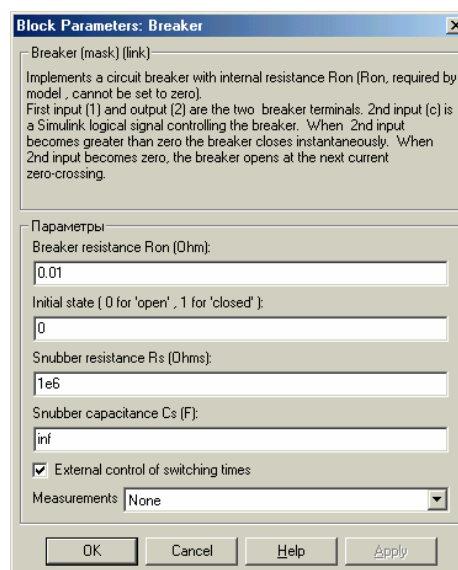


Рис. 11. 16. Вікно настроювання блоку *Breaker*

Для імітування вимикачів (рубильників) передбачені блоки третьої групи (Breakers). Вони забезпечують моделювання процесів, що виникають при вми-

канні або вимиканні змінного струму. На рис. 11.16 показано вікно налаштування простішого з них – блоку *Breaker*.

Група *Transformers* містить блоки, що імітують роботу трансформаторів. Блок *Linear Transformer* (Лінійний трансформатор) представляє собою лінійну модель трансформатора, яка не враховує насичення в обмотках трансформатора. У число параметрів налаштування блоку входять (рис. 11.17):

Nominal power and frequency (Параметри першої обмотки)	Задається вектор з двох елементів: перший – номінальна потужність трансформатора у вольт-амперах, другий – частота живлення у герцах
Winding 1 parameters (Параметри першої обмотки)	Задається вектор з трьох елементів: перший – напруга на обмотці у вольтах, другий – опір обмотки, третій – індуктивність обмотки
Winding 2 parameters (Параметри другої обмотки)	Задається вектор з трьох елементів: перший – напруга на обмотці у вольтах, другий – опір обмотки, третій – індуктивність обмотки
Winding 3 parameters (Параметри третьої обмотки)	Задається вектор з трьох елементів: перший – напруга на обмотці у вольтах, другий – опір обмотки, третій – індуктивність обмотки
Magnetization resistance and reactance (Активний і реактивний опір намагнічування)	Задається вектор з двох елементів: перший – опір намагнічування, другий – індуктивність намагнічування

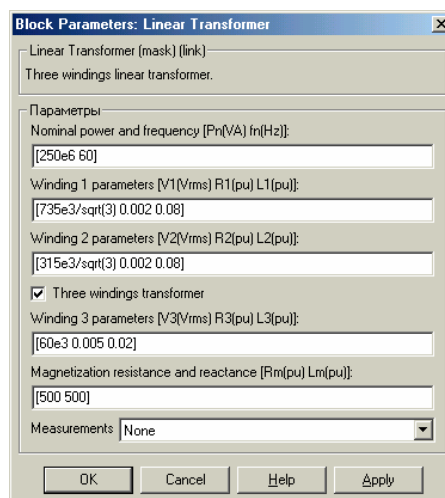


Рис. 11. 17. Вікно налаштування блоку *Linear Transformer*

Величини опорів і індуктивностей у цьому випадку задаються у відносних одиницях (pu – per unit), тобто як величини відношення до базового опору і базової індуктивності. Останні визначаються співвідношеннями:

$$R_{base_j} = V_j^2 / P_n; \quad L_{base_j} = R_{base_j} / (2\pi f_n),$$

де індекс j позначає номер обмотки, V_j – напруга на обмотці. Решта позначень такі, як зображені у вікні налаштування. Базові опір і індуктивність для параметрів намагнічування збігаються з відповідними базовими параметрами для першої обмотки.

11.3. Розділ Connectors

Вміст розділу *Connectors* (З'єднувальні елементи) поданий на рис. 11.18.

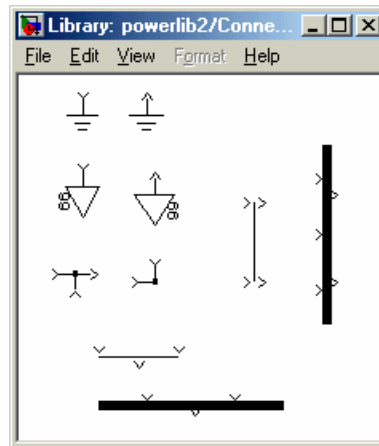


Рис. 11. 18. Вміст розділу *Connectors*

Призначення блоків цього розділу зазвичай цілком зрозуміло. Перші два блоки прислужуються для підключення елементів електричного ланцюга до загального проводу – землі. Другі два блоки – нейтралі – відображують також загальний провід для кількох елементів електричної схеми, але, на відміну від заземлення, потенціал якого приймається рівним нулю, потенціал нейтралі може бути довільним. Самих нейтралей на схемі може бути декілька. Задля відміни їх від землі та інших нейтралей (з іншими потенціалами) нейтралі забезпечуються порядковим номером (номером вузла), який проставляється у вікні налаштування блоку.

Решта чотири елементи являють собою різноманітні поєднання розгалужень і з'єднань електричних ліній.

11.4. Розділ Power Electronics

Вікно розділу *Power Electronics* (Елементи силової електроніки) показано на рис. 11.19.

У ньому містяться такі блоки:

<i>Ideal Switch</i> (Ідеальний ключ)	Керований перемикач
<i>Diode</i> (Діод)	Напівпровідниковий діод
<i>Thyristor</i> (Тиристор)	Спрощена модель тиристора
<i>Detailed Thyristor</i> (Деталізований тиристор)	Детальна модель тиристора
<i>Gto</i> (Замиканий тиристор)	Силовий біполярно-польовий блок

IGBT	Комбінація біполярних транзисторів з польовими
Mosfet	Спрощена модель польового транзистора з ізолюваним затвором
Universal Bridge	Універсальний міст
Three Level Bridge	Трирівневий міст

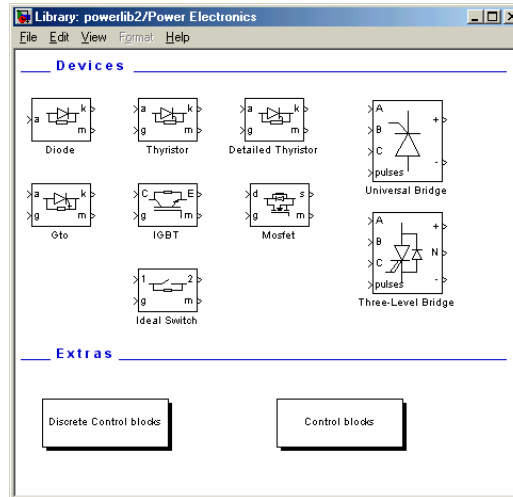


Рис. 11. 19. Содержимое раздела Power Electronics

Усі моделі містять імітацію послідовного ланцюга R_sC_s (snubber), який гасить виброси напруги. Блоки мають m -вихід, на якому формується вектор поточних значень струму, що тече крізь відповідний елемент, і напруги на клеммах елемента. Як звичайний S -сигнал, сигнал з цього виходу може бути використаний будь-якими S -блоками для перетворення і виведення результатів.

У керованих блоках (тиристри) передбачений також і m -вхід, позначений g , до входу якого надається керувальний S -сигнал, попередньо сформований S -блоками.

У блоках мостів для встановлення величин, які мають бути виміряні і вектор яких буде видавати m -вихід, у вікні настроювання у нижній його частині міститься список Measurement.

11.5. Розділ Machines

Вміст розділу Machines (Машини) поданий на рис. 11.20.

Далі наводиться перелік блоків цього розділу, які реалізують різні моделі електромашин.

Simplified Synchronous Machine pu Units	Спрощена модель синхронної машини у відносних одиницях
Simplified Synchronous Machine SI Units	Спрощена модель синхронної машини в одиницях системи СІ
Permanent Magnet Synchronous Machine	Синхронна машина з постійним магнітом

Synchronous Machine pu Fundamental	Основна (повна) модель синхронної машини у відносних одиницях
Synchronous Machine SI Fundamental	Основна (повна) модель синхронної машини в одиницях системи SI
Synchronous Machine pu Standart	Стандартна модель синхронної машини у відносних одиницях
Asynchronous Machine pu Units	Модель асинхронної машини у відносних одиницях
Asynchronous Machine SI Units	Модель асинхронної машини в одиницях SI
DC Machine	Машина постійного струму
Discrete DC Machine	Машина постійного струму з дискретним керуванням

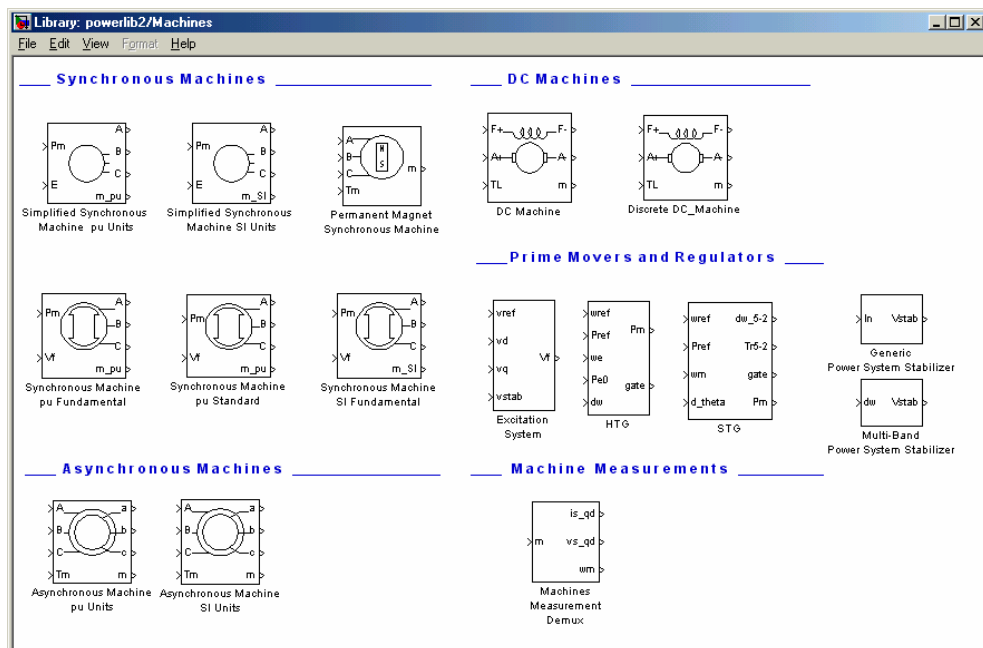


Рис. 11. 20. Вміст розділу Machines

На відміну від розглянутих раніше блоків, перераховані блоки моделюють поведінку електромеханічних пристроїв, а тому, окрім електричних зв'язків, необхідно містять зв'язки механічних величин і моделі механічних процесів. Тому вони містять *m*-входи і *m*-виходи, які дозволяють підключити до моделі *S*-блоки, в яких реалізуються динамічні процеси перетворення механічних величин.

Так, у кожному з блоків електромашин присутні один *m*-вихід, відзначений на зображенні блоку знаком «*m*». Цей *m*-вихід являє собою *S*-сигнал у вигляді вектора з електричних і механічних величин, які описують поточний стан обраного вида електромашини. Склад вектора різний для різних моделей машин. Розібратися у складі цього вектора і обрати ті з величин, які необхідно використати для побудови моделі можна, якщо під'єднати до *m*-виходу спеціальний блок з того самого розділу – **Machine Measurement Demux** (Розподільник

машинних вимірювань). На рис. 11.21 показано вікно налаштування цього блоку.

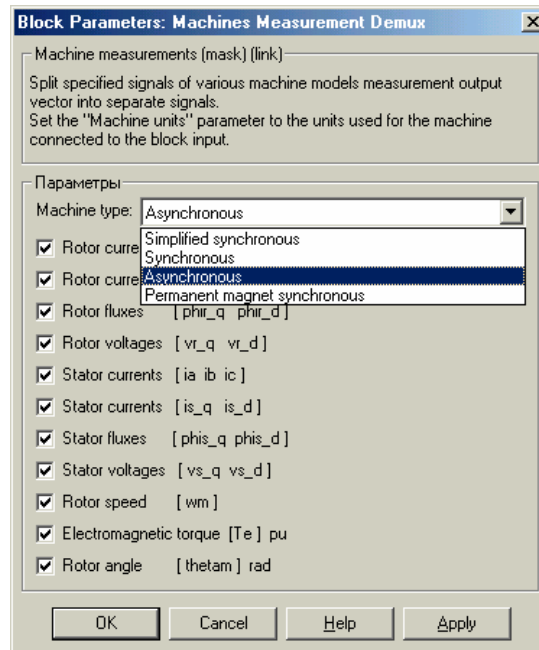


Рис. 11. 21. Вікно налаштування блоку *Machine Measurement Demux (Asynchronous)*

У верхній частині вікна розташований список, який дозволяє обрати тип моделі машини (*Simplified Synchronous, Synchronous, Asynchronous, Permanent Magnet Synchronous*). Із нього слід обрати той тип машини, до виходу якої під'єднаний цей блок *Machine Measurement Demux*. У залежності від вибору з списку у вікні налаштування виникне той чи інший набір величин, що описують вимірюваний вектор.

На рис. 11.21 поданий набір вимірюваних величин для асинхронної машини. Видно, що він містить струми і напруги у різних електричних частинах машини та, крім того, три механічні величини:

Rotor speed
Electromagnetic torque
Rotor angle

Кутова швидкість обертання ротора
Електромагнітний момент
Кут повороту ротора

Набір вимірюваних для синхронної машини величин подано на рис. 11.22. У нього входять такі вимірювані механічні величини:

Rotor angle deviation
Rotor speed
Rotor speed deviation
Electromagnetic torque
Rotor mechanical angle
Load angle

Кут відхилення ротора від положення рівноваги
Кутова швидкість обертання ротора
Кутова швидкість відхилення ротора від положення рівноваги
Електромагнітний момент
Кут повороту ротора
Кут навантаження

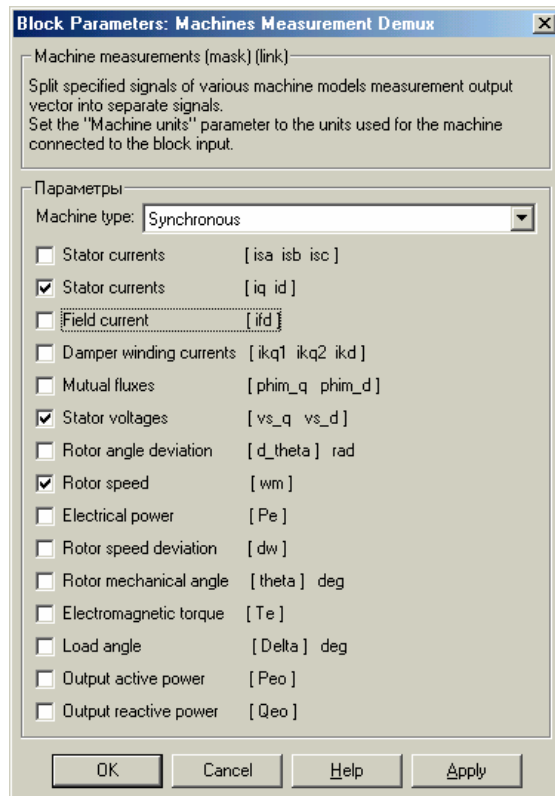


Рис. 11. 22. Вікно настроювання блоку *Machine Measurement Demux (Synchronous)*

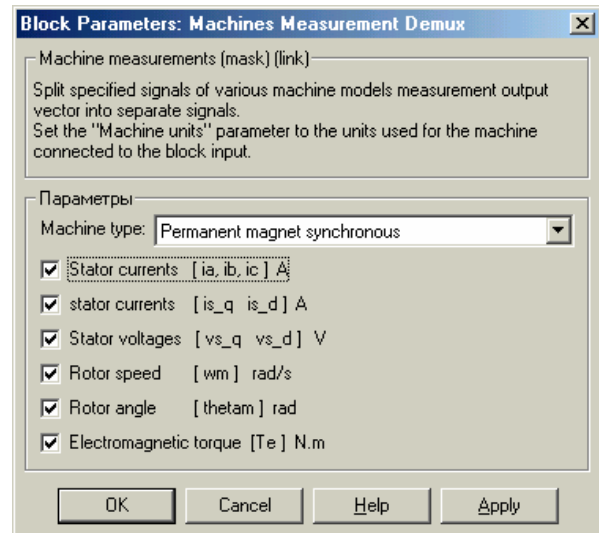


Рис. 11. 23. Вікно настроювання блоку *Machine Measurement Demux (Permanent Magnet Synchronous)*

Для машини постійного струму (рис. 11.23) передбачені ті самі вимірювані механічні величини, що й для асинхронної машини.

Встановлення прапорця поряд з обраними із поданого набору величинами забезпечує наявність цих величин у вихідному сигналі блоку *Machine Measurement Demux*. При цьому величини у вихідному векторі розташовуються у порядку, поданому у вікні настроювання блоку.

У кожному блоці електричної машини передбачені р-входи та (або) р-виходи для підключення до електричної схеми. Так, у блоках синхронних машин такими є виходи А, В і С, що імітують клеми обмоток статора. У блоках асинхронних машин – це клеми статора А, В і С та клеми ротора а, b і с. Зображення блоків машин постійного струму містять чотири клеми електричного з'єднання – F+ і F– для під'єднання обмотки збудження, а також А+ та А– для під'єднання якоря.

Решта входів блоків електричних машин являють собою m-входи, тобто до них мають надаватися S-сигнали. Так, до входу E блоку спрощеної синхронної машини надається амплітуда керувальної напруги, а до входу Pm – сигнал, рівний поточному значенню механічної потужності на валу машини. До входу Vf основної моделі синхронної машини надається сигнал збудження.

У блоках синхронної машини з постійними магнітами і асинхронних машин m-входом є механічний момент Tm на валу машини, а в блоках машин постійного струму – момент TL навантаження на валу.

Сигнали, що надаються до цих входів, мають бути сформовані у моделі системи або як явні функції часу, або в S-блоках, що імітують роботу блоків

керування або динаміку механічних пристроїв, що є навантаженням на валу машини. У розділі *Machines* є кілька блоків, що здійснюють ці функції:

Excitation system	(Система збудження) здійснює формування напруги збудження синхронної машини
Generic Power System Stabilizer	Стабілізатор коливань
Multi-band Power System Stabilizer	Багатосмуговий стабілізатор коливань
HTG	(Гідравлічна турбіна з регулятором) модель динаміки гідравлічної турбіни ПД-регулятором для синхронного генератора
STG	(Парова турбіна з регулятором) модель динаміки чотириступінчастої турбіни з ПД-регулятором для синхронного генератора

11.6. Розділ Measurements

На рис. 11.24 показано вміст розділу роздела *Measurements* (Вимірювачі).

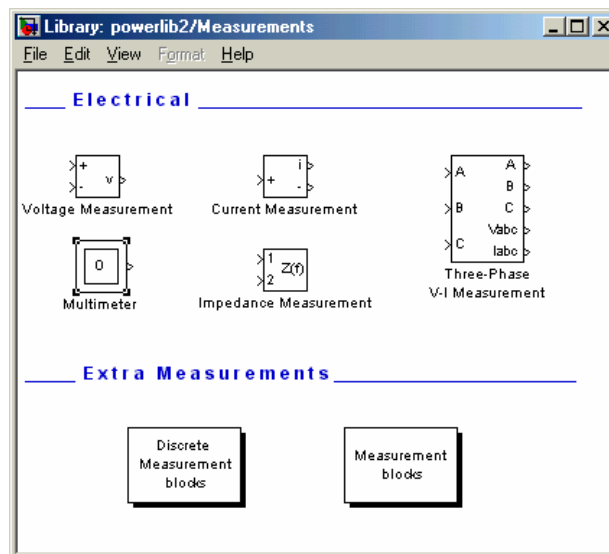


Рис. 11. 24. Вміст розділу Measurements

Найважливішими з них є блоки **Voltage Measurement** (Вольтметр), **Current Measurement** (Амперметр) и **Multimeter** (Багатофункціональний вимірювач). Важливість їх полягає у тому, що ці блоки мають, на відміну від інших блоків бібліотеки **SimPowerSystems** один m-вихід (він позначений знаком v в блоці вольтметра і знаком i – у блоці амперметра). Решта входів і виходів цих блоків являють собою, як зазвичай, клеми для під'єднання вимірювача до електричного ланцюга.

Завдяки наявності в цих блоках m-виходу, сигнал з цього виходу (напруга або струм) може бути надалі перетворений за допомогою звичайних S-блоків, а потім використаний у перетвореному вигляді знову в електричній схемі, якщо його подати до входу блоку **Controlled Voltage Source** або блоку

Controlled Current Source. Крім того, цей сигнал може бути транспортований звичайними засобами у робочий простір MatLab, а потім поданий графічними засобами у вікні *Figure*.

Особливе місце посідає блок **Multimeter** (Багатофункціональний вимірювач). Його призначено для "вимірювання" і формування вектора S-сигнала на його виході, що складається з величин, які вибрані в списках *Measurement*, що містяться в блоках тієї блок-схеми, де розташований блок **Multimeter**.

Вікно настроювання блоку **Multimeter** наведено на рис. 11.25. Воно складається з двох полів – *Available Measurements* (Доступні вимірювання) та *Selected Measurements* (Відмічені вимірювання). В першому полі автоматично (без втручання користувача) виникають усі величини, які відмічені в списках *Measurements* вікон настроювання блоків, що містяться у блок-схемі з встановленим в ній блоком **Multimeter**. Відмічаючи у цьому полі ті з вимірюваних величин, які потрібно включити в вихідний сигнал блоку **Multimeter**, за допомогою кнопки **>>** користувач переводить їх у полі *Selected Measurements*. У такий спосіб формується у цьому полі список вимірюваних величин. Порядок проходження у списку може бути змінений на бажаний за допомогою кнопок *Up*, *Down* та *Remove* (рис. 11.25) у вікні настроювання.

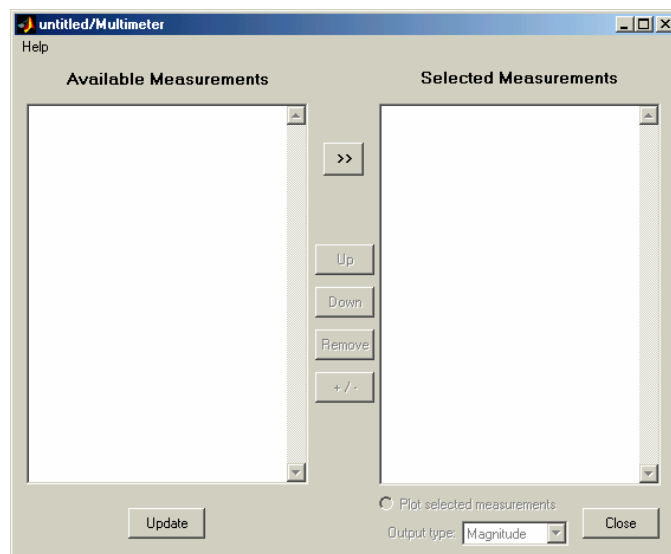


Рис. 11. 25. Вікно настроювання блоку *Multimeter*

Для виведення графіків відмічених величин безпосередньо у графічне вікно слід встановити прапорець поряд з написом *Plot selected measurement*.

Детальний опис блоків бібліотеки **SimPowerSystems** та приклади моделювання на основі використання цієї бібліотеки наведені у книзі [1, урок 10, с. 436–466] і [6].

12. ВИКОРИСТАННЯ БІБЛІОТЕКИ SIMMECHANICS

Бібліотека *SimMechanics* пакета *Simulink* містить (у вигляді візуальних блоків) програмні засоби для моделювання механічного руху механізмів і машин.

Як й в раніше розглянутій бібліотеці *SimPowerSystems*, ідеологія складання блок-схем в ній суттєво відрізняється від ідеології функціональних блок-схем бібліотеки *Simulink*, тобто S-моделей. В блок-схемі *SimMechanics* окремі блоки фактично слід розглядати як моделі, що імітують механічний рух однієї частини модельованого механізму відносно іншої його частини. "Входи" і "виходи" блоку фактично такими не є, а являють собою імітацію "посадкового" місця відповідної частини механізму. Лінії з'єднання "входів" і "виходів" блоків імітують жорсткі з'єднання однієї ("вихідної") частини одного механізму з "вхідною" частиною іншого механізму. Можна стверджувати, що це з'єднання моделює передавання силової дії між частинами різних механізмів. Але, через те що, у відповідності з третім законом Ньютона, сила дії дорівнює силі протидії, таке передавання сили не можна розглядати як однонапрявлену дію. Тому в блок-схемах *SimMechanics* на лініях з'єднання механічних блоків немає зображень стрілок, які вказують напрямок дії. З тієї самої причини графічні зображення "входів" і "виходів" механічних блоків мають вигляд не направлених стрілок, а квадратів з діагоналями.

Як і в блоках бібліотеки *SimPowerSystems*, "входи" і "виходи" механічних блоків не можна вважати джерелами чи приймачами якихось сигналів. До ліній їх з'єднань неможливо под'єднати звичайні S-блоки, а тому неможливо і сформувати за допомогою останніх задані дії або вивести (наприклад, в оглядові вікна або в систему Matlab) інформацію про рух механізмів, що отримуються в результаті моделювання. Але, через те, що будь-яке моделювання механізмів неможливо здійснити без задання потрібних досліднику дій і без виведення результатів моделювання в середовище Matlab, така ідеологія побудови блок-схем механізмів потребує включення в бібліотеку блоків, які б здійснювали прямий і зворотний зв'язок між S-блоками і механічними блоками. Такі блоки за необхідності повинні мати хоча б один m-вхід і один механічний "вихід" (для "сприйняття" заданої дії і переведення її у механічну дію), або механічний "вхід" і m-вихід (для відображення результатів модельованого механічного руху у вигляді інформаційного сигналу).

Як використовувати блоки *SimMechanics* і скласти працездатну програмну модель з їх застосуванням можна ознайомитися у [1, урок 11, с. 467–505].

12.1. Загальна характеристика бібліотеки *SimMechanics*

Якщо за допомогою контекстного меню відчинити бібліотеку *SimMechanics* із браузера *Simulink*, то на екрані виникне вікно, подане на рис. 12.1.

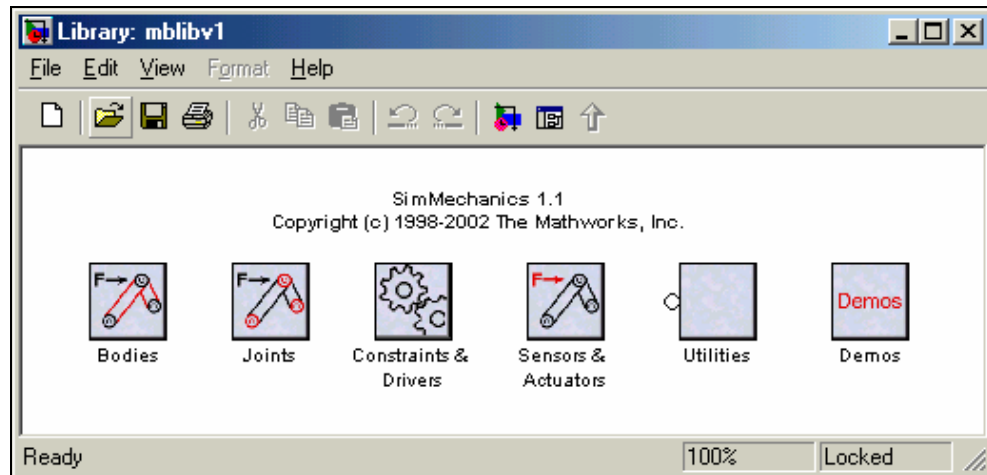


Рис. 12.1. Вікно бібліотеки *SimMechanics*

Як бачимо, бібліотека містить 6 розділів:

Bodies (Тіла)	Містить блоки, які моделюють рівняння руху твердих тіл
Joints (Зчленування)	Містить блоки імітування механічних зчленувань, що забезпечують потрібні ступені вільності однієї частини механізму відносно іншої
Constraints & Drivers (В'язі і передачі)	Складається з блоків імітування обмежень на ступені вільності механічної системи
Sensors & Actuators (Датчики і приводи)	Містить блоки, що імітують вимірювачі параметрів механічного руху і блоки задання руху частин механізму
Utilities (Утиліти)	Містить допоміжні блоки, корисні при утворенні моделі механізму
Demos (Демонстраційні програми)	Дозволяє ініціювати до виконання демонстраційні моделі

Розділ **Bodies** містить два блоки: **Ground** і **Body** (рис. 12.2).

Блок **Ground** (Основа) є обов'язковим при побудові моделі будь-якого механізму. Він презентує незмінні точки основи (Землі), нерухомі в абсолютному (інерціальному) просторі. Рухи окремих частин механізму задаються або визначаються по відношенню до системи координат, втілюваної саме цим блоком.

Блок **Body** (Тіло) подає окрему частину механізму, що розглядається як тверде тіло, рух якого моделюється. У ньому задаються маса і матриця інерції цього твердого тіла, його початкове положення і орієнтація (тобто положення і орієнтація системи координат CS, жорстко зв'язаної з ним). В число систем

координат, жорстко зв'язаних з тілом, обов'язково входить система CG (Center of Gravity), початок якої суміщено з центром тяжіння тіла. Саме відносно осей цієї системи координат задається матриця моментів інерції тіла.

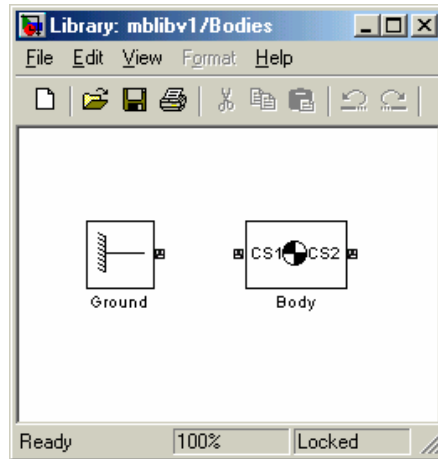


Рис. 12.2. Вміст розділу Bodies

Розділ **Joints** (Зчленування) містить блоки, які забезпечують можливість відносних рухів між тілами, поданими окремими блоками **Body**, тобто забезпечують необхідні ступені вільності (DoFs). Сюди входять блоки **Prismatic**, **Revolute**, **Spherical**, які імітують простіші призматичні, шарнирні і сферичні зчленування, а також блоки складних готових зчленувань.

У розділі **Constraints & Drivers** (В'язі і передачі) зібрані блоки, що задають попередні обмеження на відносні переміщення тіл. Ці обмеження можуть бути задані як незалежні від часу в'язі (блоки **Constraints**) або як залежні від часу руху за ступенями вільності (блоки **Drivers**).

Розділ **Sensors & Actuators** (Датчики і приводи) включає блоки **Sensors** для вимірювання відносних рухів тіл і блоки **Actuators** для задання відносних рухів. Саме ці блоки організують зв'язок між механічними блоками бібліотеки **SimMechanics** і звичайними S-блоками, що дозволяє, з одного боку, використовувати можливості бібліотеки **Simulink** для формування заданих рухів, а з іншого – використовувати S-блоки для виведення результатів моделювання у графічнч вікна Matlab.

При складанні блок-схем механізмів слід зважити на таке.

1. Основу блок-схеми будь-якого механізму складає ланцюг типу

Ground – Joint – Body – Joint - ... - Body

з відкритою чи закритою топологією, де хоча б одне з тіл подано блоком **Ground**. Блоки **Body** можуть мати більш ніж два з'єднаних з ним блоків **Joint**, які фіксують розгалуження зазначеної послідовності. Але кожне зчленування (блок **Joint**) має бути під'єднано до двох і тільки двох тіл.

6. Блоки **Body** можуть бути з'єднані й блоками **Driver** або **Constraint**, які імітують в'язі.

3. Блоки *Actuator* і *Sensor* можуть бути під'єднані до будь-якого з блоків *Body*, *Joint* або *Driver*, але тільки через додаткові порти, що встановлюються у вікнах налаштування цих блоків.

4. Задання бажаного закону змінювання у часі параметрів руху можливо тільки за допомогою блоків *Actuator*, а виведення результатів у робочий простір MatLab – через блоки *Sensor*, які зв'яують блоки *SimMechanics* з середовищем *Simulink*.

Як використовувати блоки *SimMechanics* і скласти працездатну програмну модель з їх застосуванням можна ознайомитися у [1, урок 11, с. 467–505].

12.1.1. Розділ Bodies

На рис. 12.3 показано вікно налаштування блоку *Ground*.

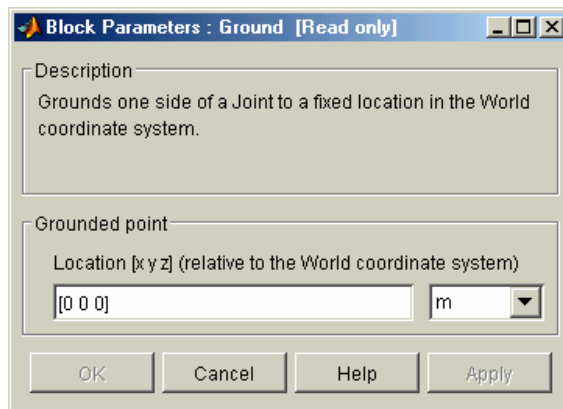


Рис. 12.3. Вікно налаштування блоку *Ground*

В ньому лише один параметр налаштування – вектор зміщення початку системи координат, зв'язаної з нерухомою частиною механізму, відносно початку інерціальної системи координат.

Вікно налаштування блоку *Body* подано на рис. 12.4 и 12.5.

Воно містить такі параметри налаштування:

Mass (Маса)	Тут вказується величина маси тієї частини механізму, яка презентується як тверде тіло
Inertia (Моменти інерції)	Тут задається квадратна матриця (3×3) моментів інерції тіла відносно ортогональних осей, які жорстко зв'язані з тілом і проходять через центр тяжіння тіла

У полі *Body coordinate systems* (Системи координат, зв'язані з тілом) розташовані дві вкладки: *Позиція* і *Орієнтація*.

У вкладці *Позиція* (рис. 12.4) розташована таблиця введення координат початку систем координат, зв'язаних з тілом. За замовчуванням, ця таблиця містить три рядки і дозволяє задати три координатні системи, зв'язані з тілом:

- систему координат *CG*, початок якої суміщено з центром тяжіння тіла;
- систему координат *CS1*, "прив'язану" до лівого порту ("входу") блоку

Body;

– систему координат CS2, "прив'язану" до правого порту ("виходу") блоку *Body*.

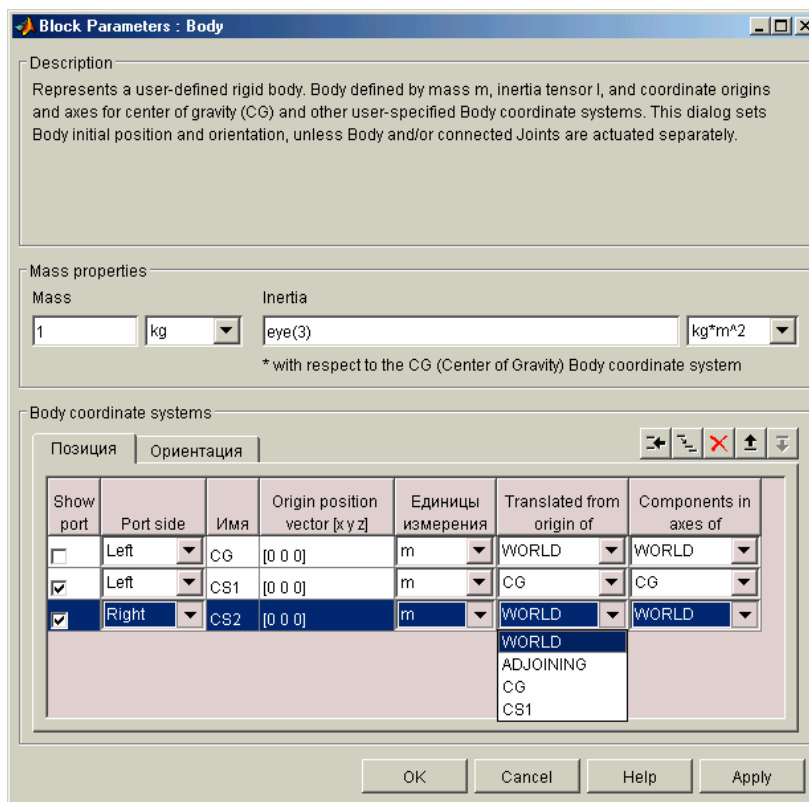


Рис. 12.4. Вікно настроювання блоку *Body*. Вкладка *Позиція*

У кожному рядку передбачено введення 7 характеристик відповідної системи координат:

Show port (Показати порт)	Встановлення (або скидання) прапорця у цій колонці дозволяє ввести (або прибрати) на зображенні блоку зображення порту, зв'язаного з відповідною системою координат тіла
Port side (Сторона порту)	Дозволяє встановити зображення порту ліворуч або праворуч на зображенні блоку
Имя	Тут встановлюється ідентифікатор запровадженої системи координат
Origin position vector [x, y, z] (Вектор положення початку)	Містить вектор координат початку відповідної системи координат
Единицы измерения	Тут встановлюються одиниці вимірювання довжини, в яких встановлені координати початку
Translated from origin of (Відраховується від початку системи координат...)	Вказується ім'я (ідентифікатор) системи координат, від початку якої відраховуються координати встановлюваної системи координат
Components in axes of (Компоненти в осях)	Вказується ім'я (ідентифікатор) системи координат, у проекціях на осі якої встановлені координати початку встановлюваної системи координат

системи координат ...)

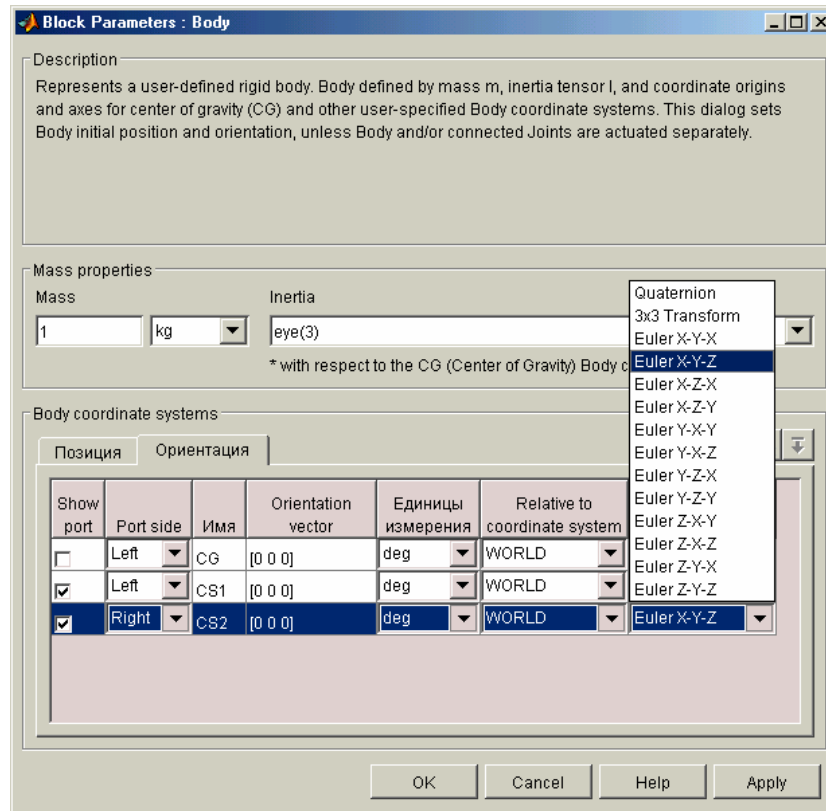


Рис. 12.5. Вікно налаштування блоку *Body*. Вкладка *Орієнтація*

Вкладка *Орієнтація* (рис. 12.5) влаштована аналогічним чином і дозволяє встановити початкову кутову орієнтацію запроваджуваної системи координат. Відмінності полягають у наступному. Замість вектору координат початку в четвертій колонці вводиться вектор кутів повороту запроваджуваної системи координат відносно вихідної, ім'я якої вказується в шостій колонці. При цьому встановлювана послідовність поворотів навколо координатних осей вказується у сьомій колонці.

Кожному місцю з'єднання тіла (блоку *Body*) з іншим тілом (блоком *Body*) відповідає своя окрема система координат *CS*. Кількість точок з'єднання тіла з іншими тілами (а, отже, й кількість зв'язаних з тілом систем координат) можна збільшити або зменшити, користуючись значками на панелі інструментів, що міститься у полі *Body coordinate systems*.

На рис. 12.5 показаний список можливого вибору систем координат відліку. В нього входять інерціальна система відліку *WORLD*, усі системи координат, що є на вкладці, а також система *ADJOINING*, під якою розуміють систему координат, зв'язану жорстко з тим зчленуванням, яке під'єднано до тіла.

12.1.2. Розділ Joints

У розділі *Joints* містяться 15 блоків різних видів зчленувань (рис. 12.6) та два підрозділи *Disassembled Joints* (Розібрані зчленування) и *Massless Connectors* (Безінерційні з'єднувачі).

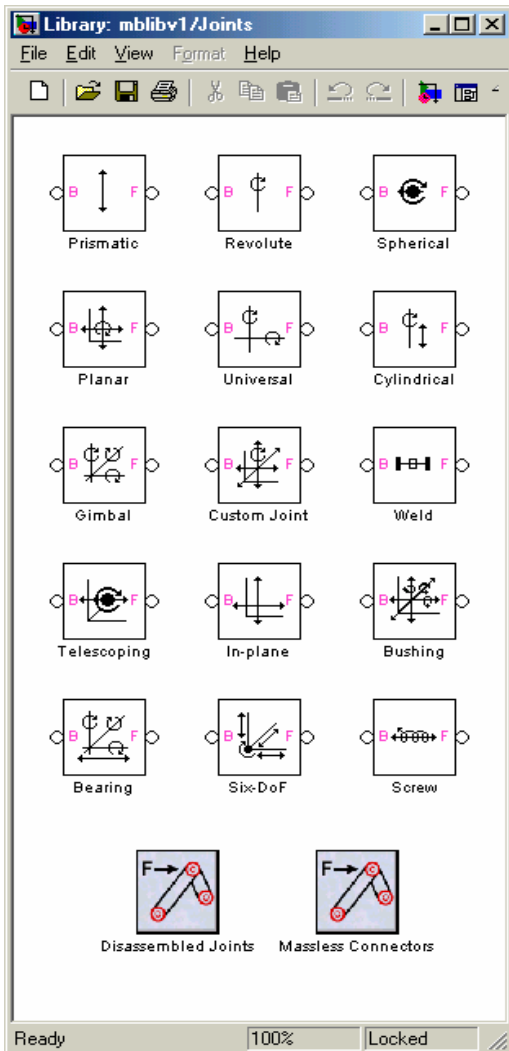


Рис. 12.6. Вміст розділу Joints

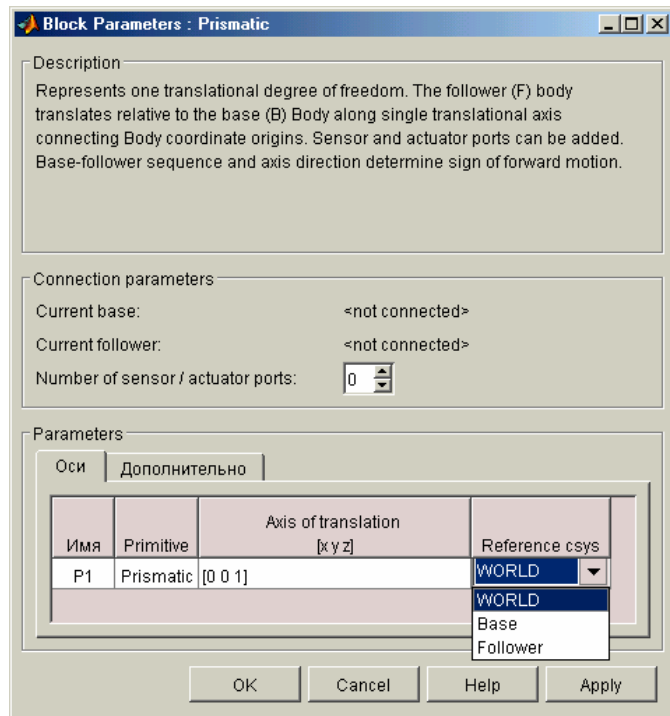


Рис. 12.7. Вікно налаштування блоку Prismatic

Блоки цього підрозділу мають два обов'язкові порти, за допомогою яких вони під'єднуються до двох блоків з розділу *Bodies*. Один з цих портів відзначений індексом B (Base – Основний), другий – індексом F (Follower – наступний). Перший призначений для з'єднання з блоком *Body*, який презентує перше (основне) тіло, другий – для під'єднання до наступного тіла у зв'язаному ланцюгу тіл, що складають механізм.

Розглянемо деякі найбільш прості і важливі блоки.

Почнемо з блоку *Prismatic*. Він забезпечує один поступальний ступінь вільності уздовж осі, вказаній у вкладці *Оси* вікна його налаштування (рис. 12.7).

На рис. 12.7 такою віссю вільного переміщення встановлена третя вісь (z) інерціальної системи координат. Як видно з того самого рисунку, є можливість зв'язати вісь відносного переміщення також з однією з осей першого тіла, з

яким зв'язаний блок **Prismatic** (обравши *Base* з пропонованого списку), або з однією з осей системи координат, зв'язаної з другим тілом (обравши *Follower*).

У полі *Connection parameters* (Параметри з'єднання) вказані три параметри – *Current base* (Поточна база), *Current follower* (Поточне ведене тіло) та *Number of sensor/actuator ports* (Кількість портів для вимірювачів та Збудників руху). Перші два параметри не встановлюються користувачем і вказують назву блоку тіла, до якого під'єднаний відповідний порт блоку **Joint**. Якщо блок **Joint** не під'єднаний до тіл, то, як видно из рис. 12.7 напроти цих параметри виникає запис *<not connected>* (не під'єднаний).

Утворимо найпростіший ланцюг з трьох блоків **Ground**, **Prismatic** та **Body** (рис. 12.8), одночасно встановивши у блоці **Prismatic** два додаткові порти для під'єднання блоків **Actuator** і **Sensor**. Тепер вікно настроювання блоку **Prismatic** буде виглядати так, як показано на рис. 12.9.

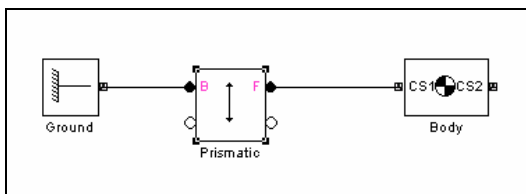


Рис. 12.8. Найпростіший механічний ланцюг

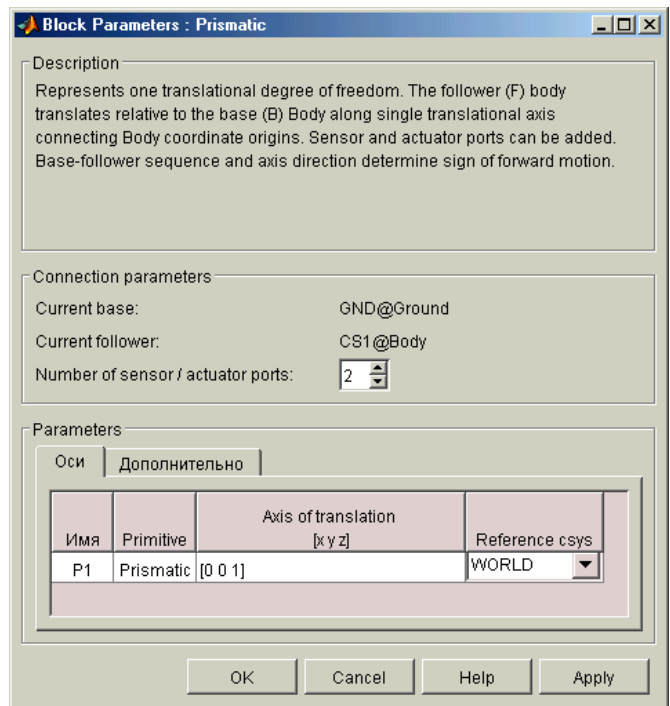


Рис. 12.9. Вікно настроювання під'єданого блоку Prismatic

Як бачимо, тепер поточна база автоматично встановлена *GND@Ground*, що означає, що базу ототожнено з системою координат *GND* блоку **Ground**, а ведене тіло *Follower* (*CS1@Body*) зв'язане з системою координат *CS1* блоку **Body**.

При цьому на зображенні блоку **Prismatic** виникли зображення двох додаткових портів. З ними тепер можна з'єднати блоки вимірювачів (**Sensor**) та (або) збудників (**Actuator**) руху.

Вікно настроювання (рис. 12.10) наступного блоку – **Revolute** (Циліндричний шарнір) – практично не відрізняється від попереднього. Різниця лише у тому, що в ньому встановлюється напрямок осі обертання тіла *Follower* відносно тіла *Base*.

Слід зазначити, що елементарне зчленування типу *Prismatic* має внутрішнє позначення P, а зчленування типу циліндричного шарніру – позначення R.

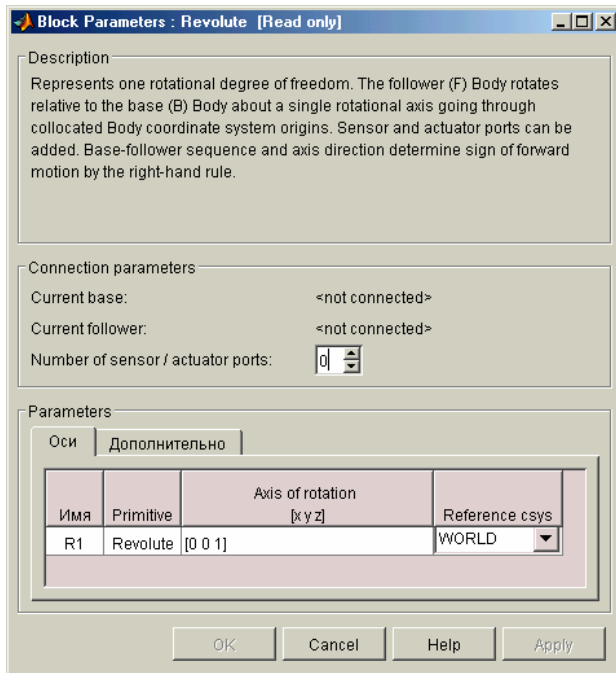


Рис. 12.10. Вікно налаштування блоку *Revolute*

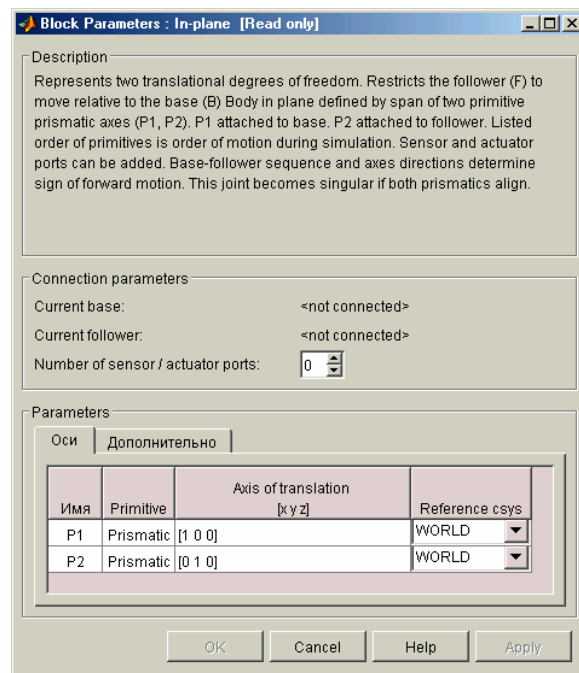


Рис. 12.11. Вікно налаштування блоку *In-plane*

Блок *In-plane* забезпечує вільність відносного поступального руху двох тіл у площині осей, напрямки яких встановлено у вікні налаштування (рис. 12.11). Неважко впевнитися, що він являє собою послідовне з'єднання двох елементарних зчленувань P1 та P2 типу *Prismatic*. Перше з них (P1) забезпечує вільність переміщення другого (P2) уздовж осі, напрямком якої вказується у першому рядку вкладки *Оси* вікна налаштування. У другому рядку тієї самої вкладки встановлюється напрямком другої осі вільного переміщення.

Блок *Universal* забезпечує вільність кутового переміщення тіла *Follower* відносно тіла *Base* відносно двох осей, що задаються у вікні його налаштування (рис. 12.12). Він являє собою послідовне з'єднання двох елементарних зчленувань R1 і R2 типу *Revolute*.

Наступний блок *Gimbal* (Кардановий підвіс) являє собою послідовне з'єднання трьох елементарних зчленувань R1, R2 і R3 типу *Revolute* і забезпечує вільність кутового переміщення одного тіла відносно другогонавколо трьох, у загальному випадку некомпланарних осей, вказаних у вікні налаштування (рис. 12.13).

Блок *Spherical* (Сферичний шарнір), як і попередній блок, забезпечує три кутові ступеня вільності відносного переміщення двох тіл. Різниця полягає у наступному. По-перше, у блоці *Spherical* немає явно виражених осей обертання, і тому вони не встановлюються у його вікні налаштування. По-друге, внаслідок цієї особливості, до блоку *Spherical* не може бути під'єднаний блок збудника руху (*Actuator*), а параметри відносних поворотів тіл не можуть бути по-

дані у кутах поворотів, а лише у вигляді вектора складових кватерніону повороту другого тіла (*Follower*) відносно першого (*Base*).

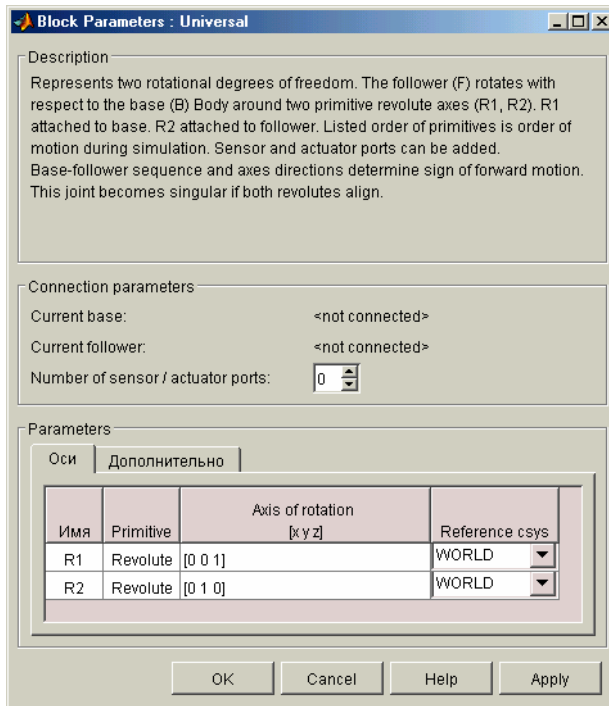


Рис. 12.12. Вікно настроювання блоку *Universal*

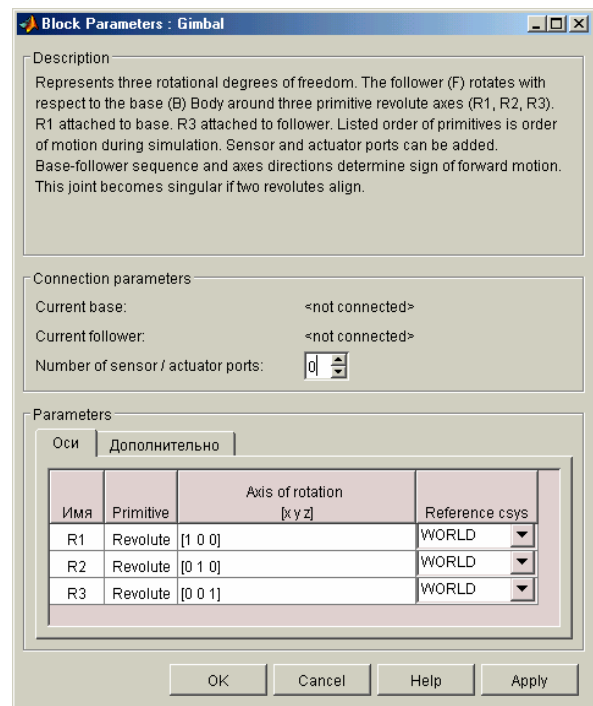


Рис. 12.13. Вікно настроювання блоку *Gimbal*

За допомогою блоку *Planar* (Плаский рух) можна забезпечити таке зчленування двох тіл, коли одна з площин провідного і веденого тіла зберігається незмінною, а точки веденого тіла можуть займати довільне положення у відповідній площині провідного тіла. Блок являє собою послідовне з'єднання трьох елементарних зчленувань – двох призматичних P1 і P2 та одного циліндричного шарніра R1. Для забезпечення плаского руху необхідно, щоб три вказані у блоці настроювання (рис. 12.15) осі були некопланарними.

Блок *Cylindrical* (Циліндричне зчленування) дає можливість імітувати таке з'єднання двох тіл, яке допускає одночасну вільність обертання навколо вказаної осі та поступального переміщення уздовж цієї ж осі (рис. 12.16). Він презентує собою послідовне з'єднання зчленування P1 типу *Prismatic* і зчленування R1 типу *Revolute*.

Особливе місце займає блок *Custom Joint* (Складальне зчленування). Він дає можливість користувачеві самому сконструювати довільний послідовний ланцюг з елементарних зчленувань. Для цього йому надається (у вигляді списку у першій колонці вкладки *Оси*, рис. 12.17) набір з трьох примитивів P1, P2, P3 типу *Prismatic*, трьох зчленувань R1, R2, R3 типу *Revolute* і одного примитиву S типу *Spherical*. Обмеження полягають у такому. Набір має забезпечувати не більш за 6 ступенів вільності – три кутові та три поступальні. Ніякі дві осі примитивів *Prismatic* або дві осі примитивів *Revolute* не мають бути паралельними (у цьому разі зчленування вироджується).

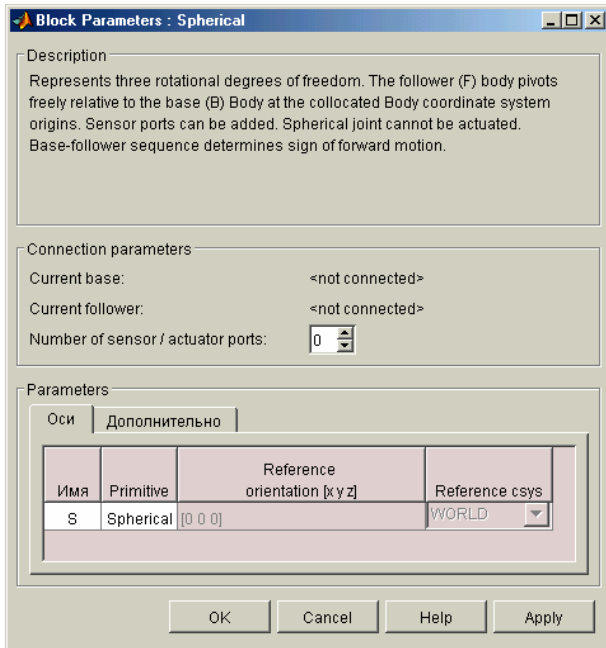


Рис. 12.14. Вікно налаштування блоку Spherical

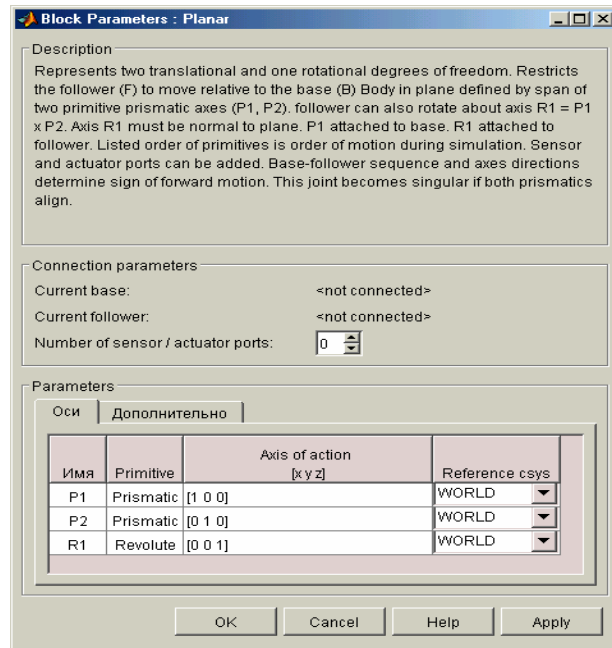


Рис. 12.15. Вікно налаштування блоку Planar

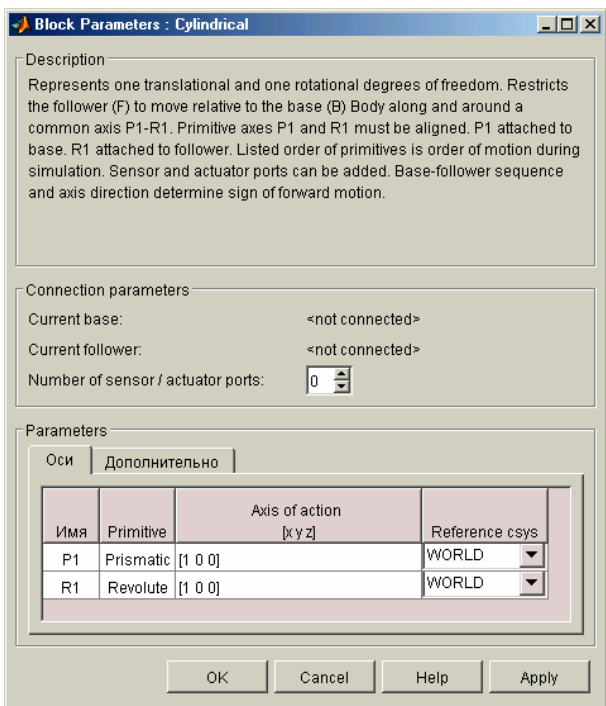


Рис. 12.16. Вікно налаштування блоку Cylindrical

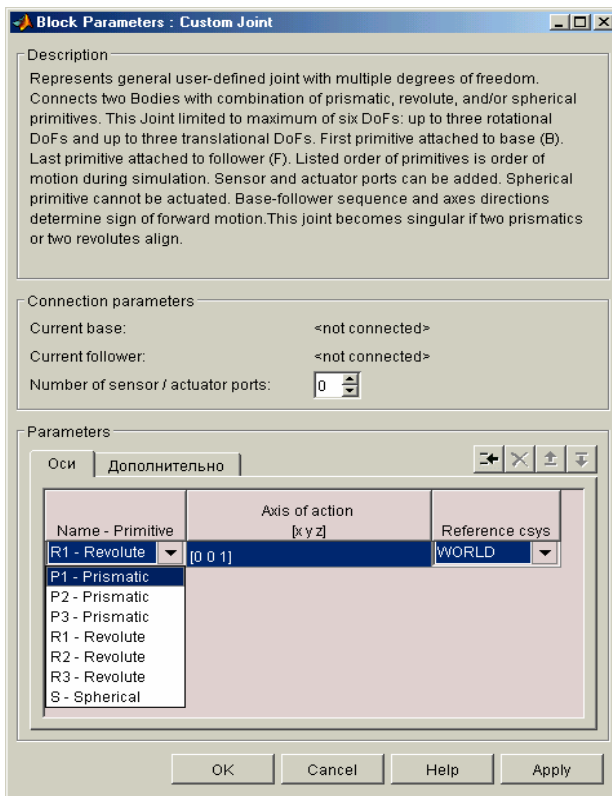


Рис. 12.17. Вікно налаштування блоку Custom Joint

Блок **Weld** (Жорстке з'єднання) прислужується для імітування жорсткого з'єднання двох тіл (рис. 12.18). Зручний для побудови моделей повідкових механізмів.

Блок *Bushing* (рис. 12.19) забезпечує шість ступенів вільності (три поступальні та три кутові) і складається з послідовно з'єднаних трьох примитивів P1, P2, P3 типу *Prismatic*, трьох зчленувань R1, R2, R3 типу *Revolute*.

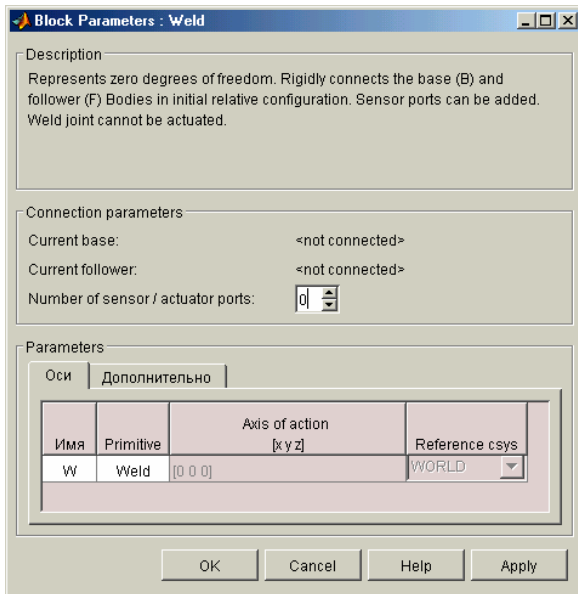


Рис. 12.18. Вікно настроювання блоку Weld

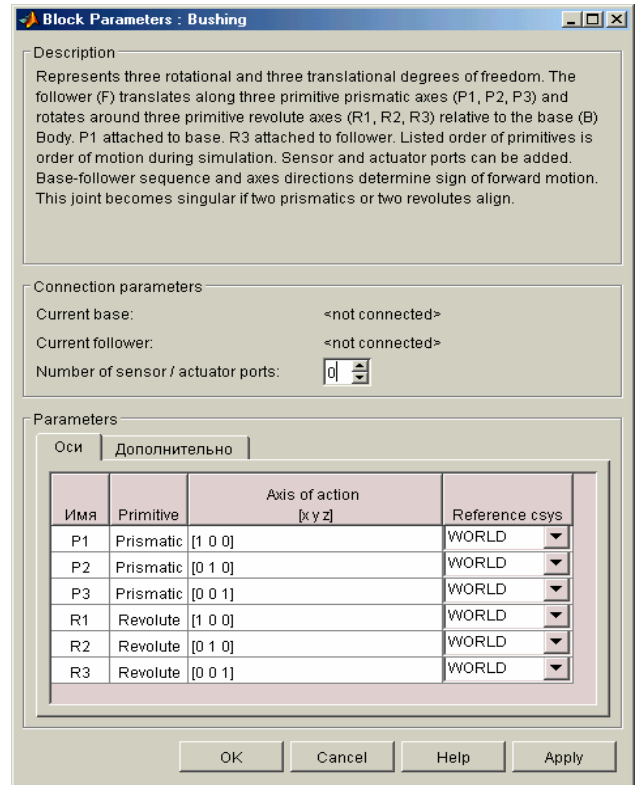


Рис. 12.19. Вікно настроювання блоку Bushing

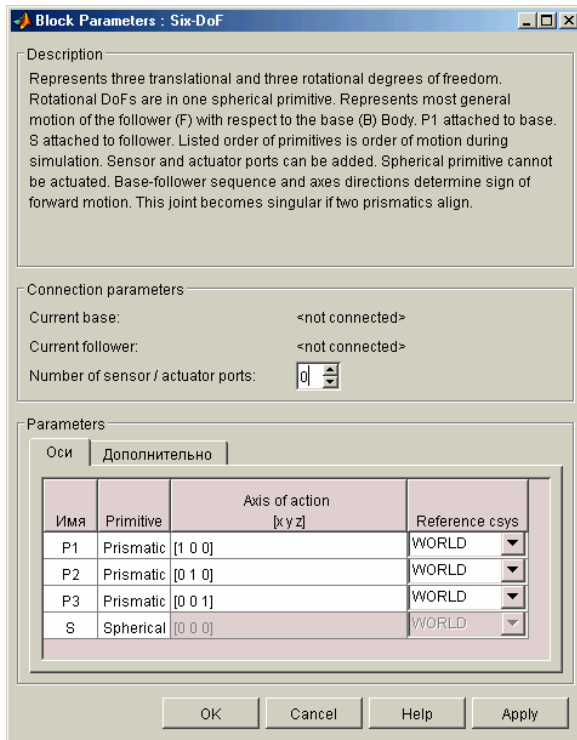


Рис. 12.20. Вікно настроювання блоку Six-DoF

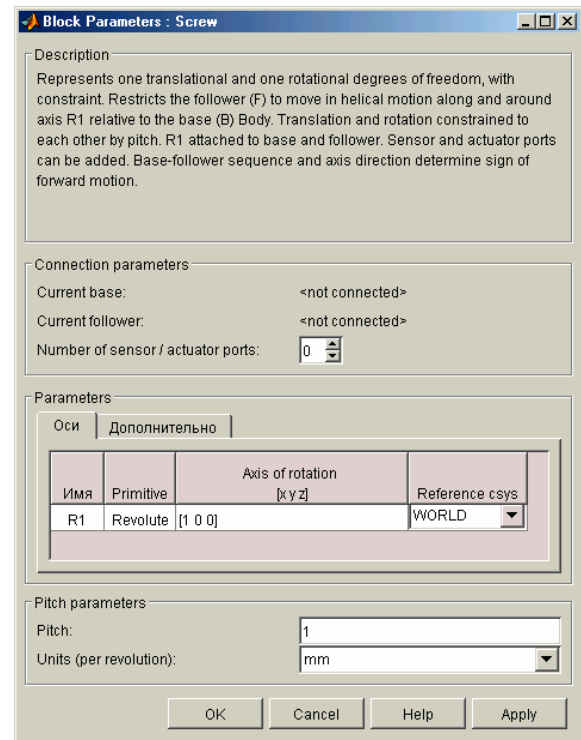


Рис. 12.21. Вікно настроювання блоку Screw

Схожу функцію виконує блок **Six-DoF** (Шість ступенів вільності). Як видно з рисунка 12.20, єдиною відмінністю від блоку **Bushing** є те, що для забезпечення трьох кутових ступенів вільності використовується один примітив S типу **Spherical** замість трьох типу **Revolute**.

Особливістю блоку **Screw** (Винт), який забезпечує гвинтову ступінь вільності відносного переміщення двох тіл, є (рис. 12.21) наявність додаткового встановлюваного параметра **Pitch** (Крок гвинта).

12.1.3. Розділ Sensors&Actuators

У цьому розділі розташовані блоки, які дозволяють задати відносні рухи тіл (блоки **Actuator**) або виміряти характеристики їх відносного руху (блоки **Sensor**). Як раніше зазначалося, блоки типу **Joint** (Зчленування) можуть бути постачені додатковими портами для під'єднання до них блоків **Actuator** та **Sensor**. Аналогічна операція можлива й по відношенню до блоків **Body**, **Driver** і **Constraint**. Вміст розділу показаний на рис. 12.22.

Особливістю розглядуваних блоків у тому, що вони є сполучними зі звичайними S-блоками бібліотеки SIMULINK, що дозволяє використовувати можливості цієї бібліотеки для формування сигналів, їх перетворення та переведення отриманих результатів у робочій простір **Simulink**.

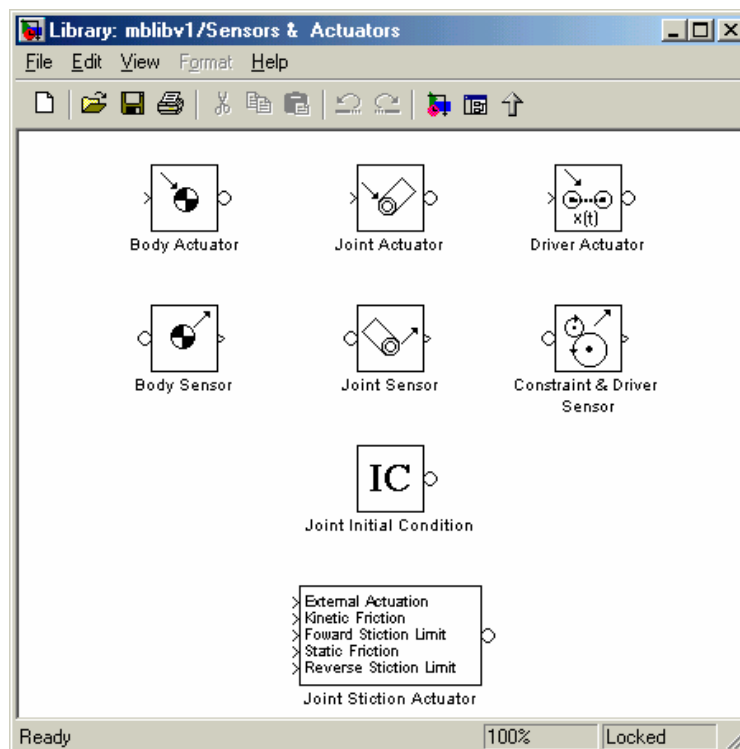


Рис. 12.22. Вміст розділу Sensors & Actuators

Як видно з рисунку 12.22, блоки збудників (**Actuator**) і вимірювачів (**Sensor**) відносного руху поділяються на три групи:

Body Actuator та
Body Sensor(Збудник руху тіла та
Вимірювач руху тіла)**Joint Actuator** та
Joint Sensor(Збудник руху зчлену-
вання та Вимірювач
руху зчленування)**Driver Actuator** та
**Driver&Constraint
Sensor**(Збудник двигуна та
Вимірювач руху двигу-
на, або в'язі)

Призначені для під'єднання до блоків Body і задають або вимірюють абсолютний рух системи координат, жорстко зв'язанної з тілом, до якого вони під'єднані

Призначені для під'єднання до блоків Joint і задають або вимірюють відносний рух того примитиву, який вказаний у вікні налаштування

Призначені для під'єднання до блоків Driver або Constraint і задають або вимірюють відносний рух двигуна або в'язі, з якими вони з'єднані

Почнемо з блоків другої групи. На рис. 12.23 и 12.24 наведені вікна налаштування блоків **Joint Actuator** та **Joint Sensor**.

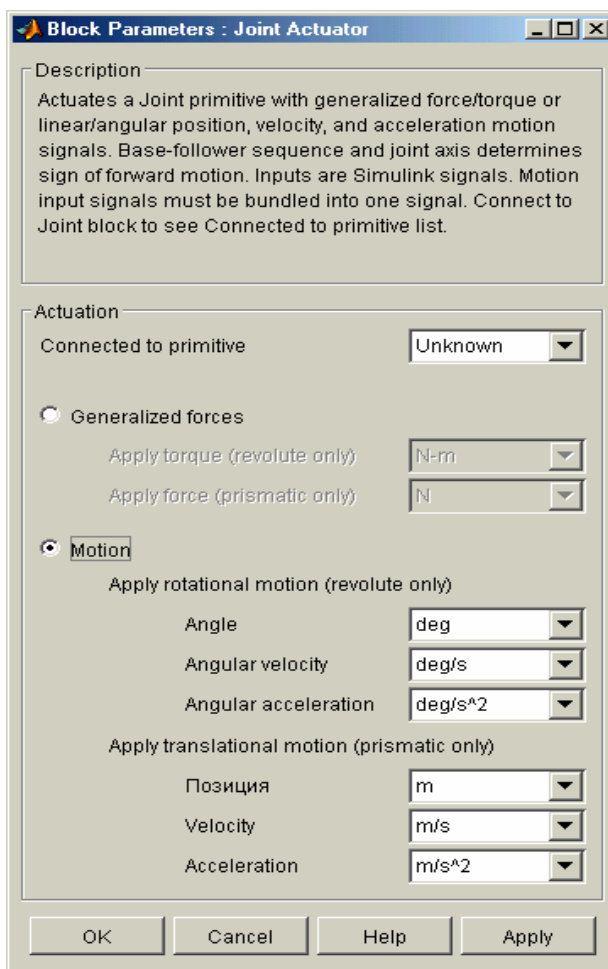


Рис. 12.23. Вікно налаштування блоку **Joint Actuator**

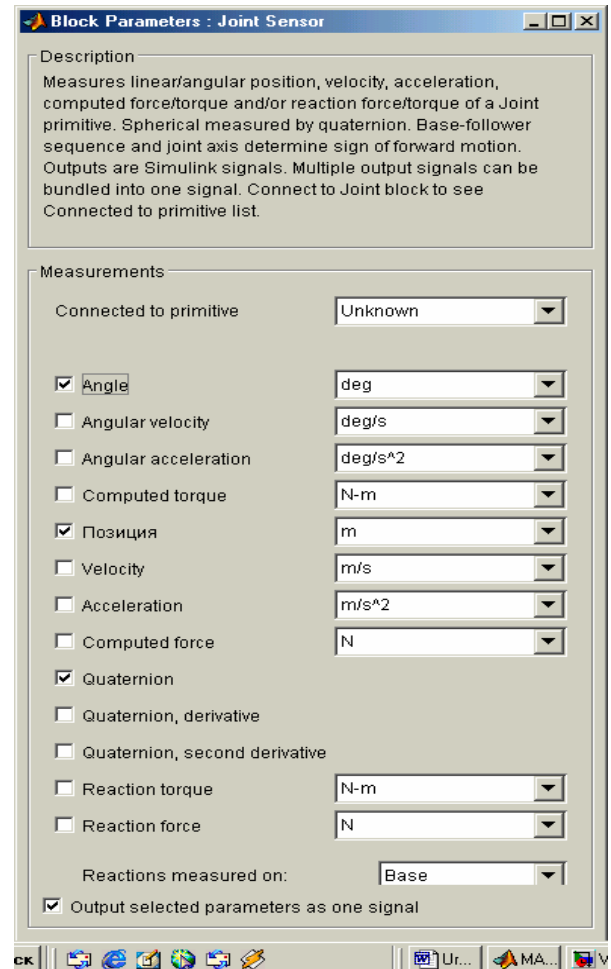


Рис. 12.24. Вікно налаштування блоку **Joint Sensor**

З їх розглядання випливає таке.

1. За допомогою блоку *Joint Actuator* можна у загальному випадку задати як функцію часу або силову дію між елементами примитиву, ім'я якого вказане у верхньому віконці введення, або відносний рух елементів цього примитиву. Встановлення виду збудження здійснюється через активізацію відповідного розділу *Generalized Forces* (Узагальнені сили) або *Motion* (Рух) у вікні налаштування блоку кнопкою ліворуч від відповідної назви розділу.

2. Завдання відносного руху частин примитиву здійснюється у вигляді завдання векторного сигналу з трьох елементів – першій з них визначає відносне переміщення, другий – відносну швидкість, а третій – відносне прискорення частин вказаного елементарного зчленування.

3. Блок *Joint Sensor* дозволяє у загальному випадку вимірювати такі характеристики відносного руху частин примитиву, ім'я якого встановлюється у верхньому віконці введення вікна налаштування блоку:

Angle (Кут)	Кут повороту вихідної частини примитиву (Follower) відносно його частини, з'єднаної зі входом (Base)
Angular velocity (Кутова швидкість)	Відносна кутова швидкість
Angular acceleration (Кутове прискорення)	Відносне кутове прискорення
Computed torque (Обчислений момент)	Повний момент сил, що спричиняє кутовий відносно прискорений рух
Позиція	Переміщення вихідної частини примитиву (Follower) відносно його частини, з'єднаної зі входом (Base)
Velocity (Швидкість)	Відносна швидкість
Acceleration (Прискорення)	Відносне прискорення
Computed force (Обчислена сила)	Повна сила, що спричиняє відносний прискорений рух
Quaternion (Кватерніон)	Вектор з чотирьох елементів, який характеризує поточне відносне положення частин примитиву
Quaternion, derivative (Похідна від кватерніону)	Вектор з чотирьох елементів, що презентують похідні за часом від відповідних елементів кватерніону відносного повороту
Quaternion, second derivative (Друга похідна від кватерніону)	Вектор з чотирьох елементів, що презентують другі похідні за часом від відповідних елементів кватерніону відносного повороту
Reaction torque (Момент реакції)	Момент сил реакції відносно осі примитиву
Reaction force (Сила реакції)	Сила реакції, напрямлена уздовж осі примитиву

Примітки.

1. Вибір необхідних для вимірювання величин проводиться встановленням прапорця поруч з назвою відповідної величини.

2. Виходом блоку є вектор, елементами якого є зазначені величини у тому порядку, як вони зазначені в списку у вікні настроювання.
3. При підключенні до конкретних елементарних зчленувань ряд вікон введення у вікні настроювання стають неактивними, тому вибір обмежується залишившимися величинами.
4. Примітив типу Spherical дозволяє виміряти тільки кватерніон повороту і його похідні

Перейдемо до розглядання блоків збудження і вимірювачів для тіл. Їх вікна настроювання наведені на рис. 12.25 і 12.26.

Як бачимо, параметри збудження можуть бути визначені як в абсолютній (інерціальній), так й по відношенню до системи координат, зв'язаної з тілом в тій точці, куди під'єднаний блок. При цьому збуджуватися можуть тільки сили і моменти, прикладені до точки під'єднання. Вимірені ж можуть бути такі величини:

[x, y, z] Позиція	Вектор абсолютного переміщення відповідної точки тіла
[x', y', z'] Velocity (Швидкість)	Вектор проєкцій абсолютної швидкості відповідної точки тіла
[θx', θy', θz'] Angular velocity (Кутова швидкість)	Вектор проєкцій абсолютної кутової швидкості тіла
[3×3] Rotation matrix	Матриця напрямних косинусов кутового положення тіла
[x'', y'', z''] Acceleration (Прискорення)	Вектор проєкцій абсолютного прискорення відповідної точки тіла
[θx'', θy'', θz''] Angular acceleration (Кутове прискорення)	Вектор проєкцій абсолютного кутового прискорення тіла

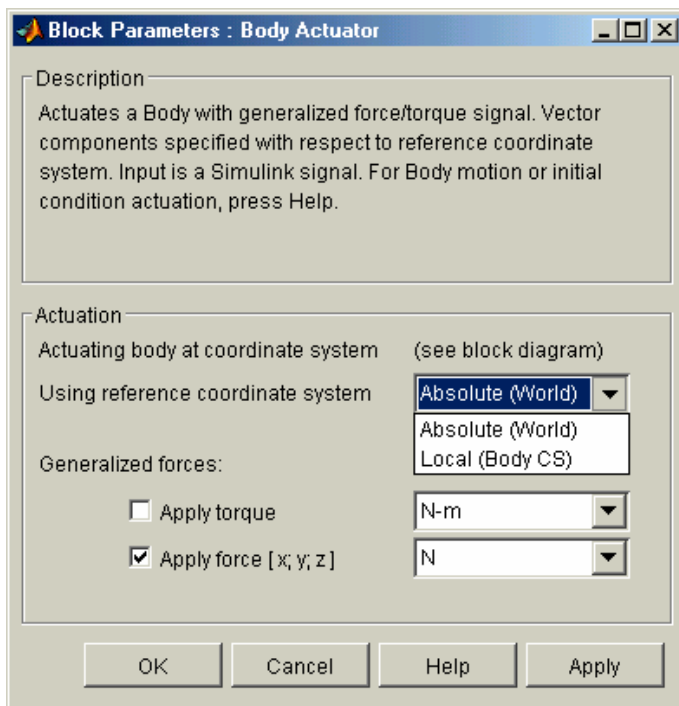


Рис. 12.25. Вікно настроювання блоку Body Actuator

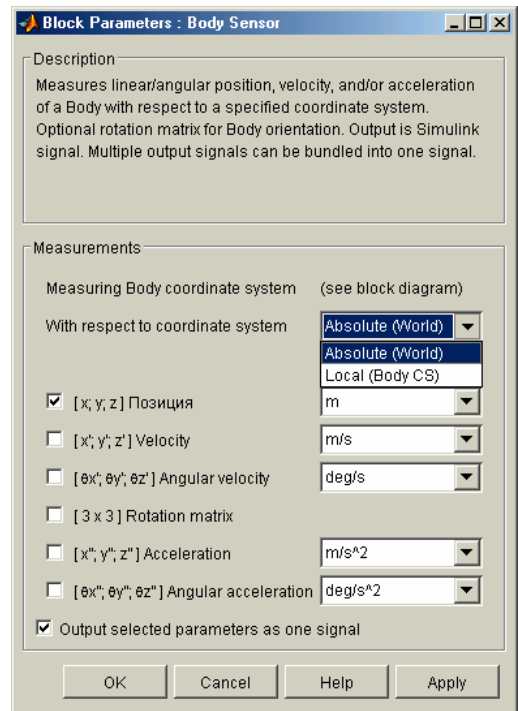


Рис. 12.26. Вікно настроювання блоку Body Sensor

Як і раніше, величини, які необхідно виміряти, мають бути відзначені прапорцями зліва, а виходом блоку є вектор усіх відзначених вимірюваних величин в порядку, зазначеному у вікні настроювання.

На рисунках 12.27 и 12.28 показані вікна настроювання блоків збудження та вимірювання у в'язях

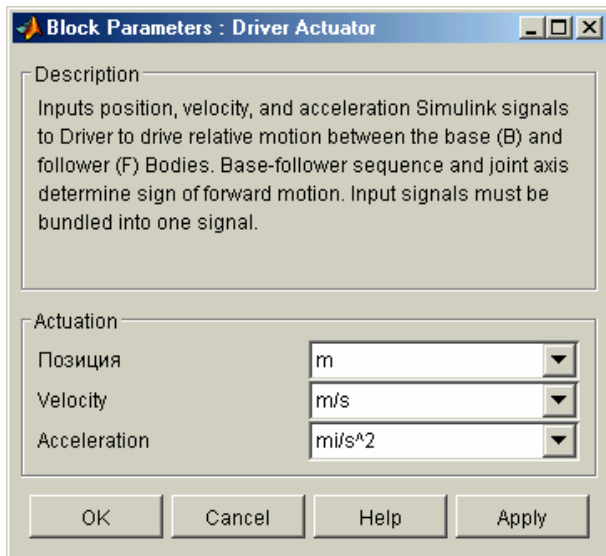


Рис. 12.27. Вікно настроювання блоку *Driver Actuator*

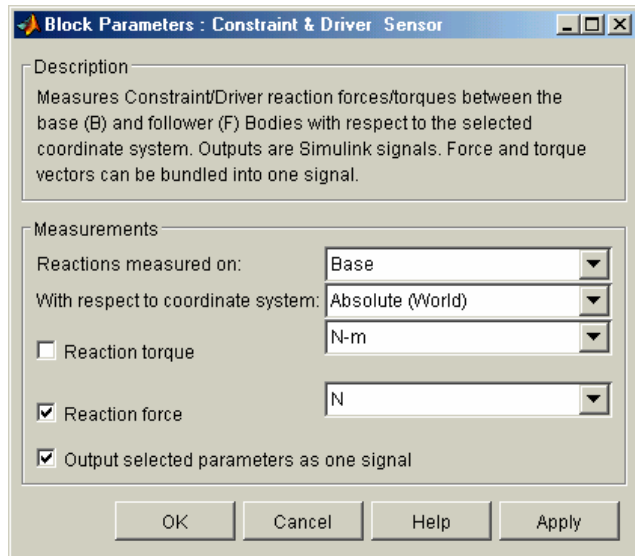


Рис. 12.28. Вікно настроювання блоку *Constraint & Driver Sensor*

З них впливає:

1) збуджуватися можуть тільки блоки нестационарних в'язів (**Driver**), а вимірювачі (**Sensor**) можуть бути під'єднані до будь-яких блоків в'язів;

2) величини, які задаються як збудження, являють собою вектори з трьох елементів (переміщення, швидкість і прискорення); при цьому переміщення можуть бути лінійними (як показано на рис. 12.27) або кутовими (якщо імітується нестационарна в'язь з куту);

3) вимірювачі (**Sensor**) можуть вимірювати тільки силу і момент сили реакції у в'язі.

Окрім зазначених блоків в розділ входять два особливі блоки. Один з них – блок **Joint Initial Condition** (Початкові Умови Зчленування) – дозволяє задати початкове відносне положення і початкову відносну швидкість (рис. 12.29) двох частин того елементарного зчленування (**Revolute** або **Prismatic**), до якого він під'єднаний.

Другий блок – **Joint Stiction Actuator** (Імітатор "зчеплення" зчленування) – дозволяє моделювати сили і моменти сил в'язького та сухого тертя в осі елементарного зчленування, включаючи явище "жорсткого" зчеплення його частин силами сухого тертя. Вікно настроювання цього блоку наведено на рис. 12.30

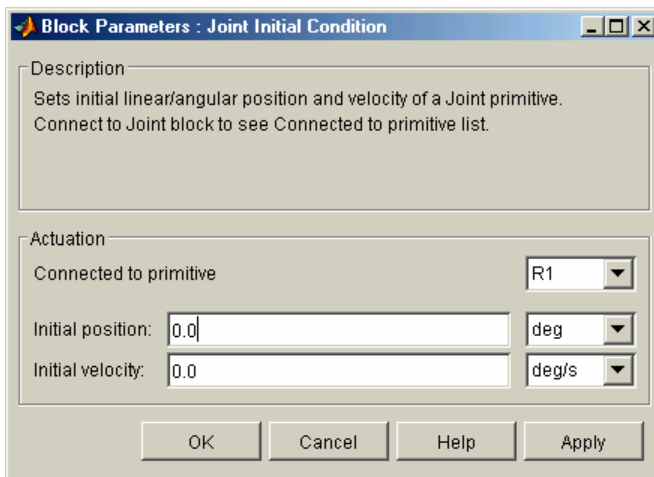


Рис. 12.29. Вікно настроювання блоку *Joint Initial Condition*

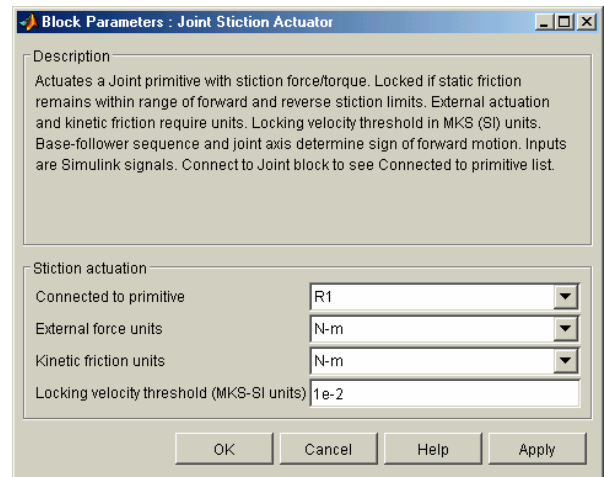


Рис. 12.30. Вікно настроювання блоку *Joint Stiction Actuator*

12.1.4. Розділ Constraints & Drivers

Цей розділ містить блоки, що імітують накладені між тілами в'язі – стаціонарні (блоки *Constraint*) або нестационарні (блоки *Driver*). Вміст розділу наведено на рис. 12.31

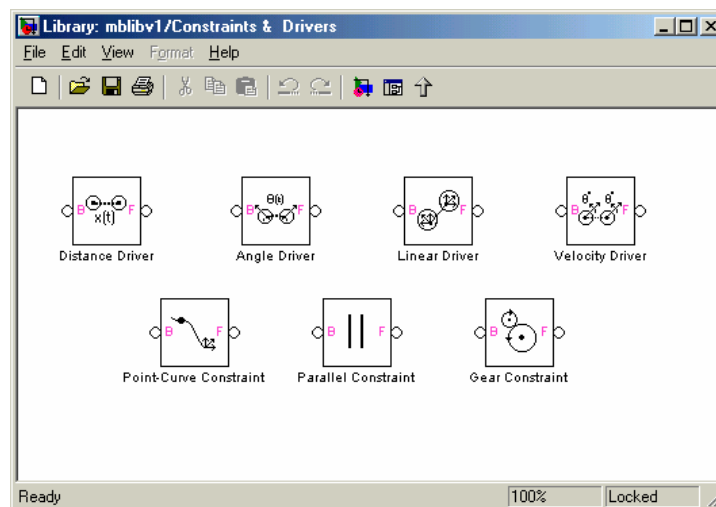


Рис. 12.31. Вміст розділу *Constraints & Drivers*

Дамо стислу характеристику блоків.

Distance Driver

Визначає нестационарну в'язь між під'єднаними до блоку тілами у вигляді залежності від часу відстані між початками систем координат, пов'язаних з цими тілами

Angle Driver

Визначає нестационарну в'язь між під'єднаними до блоку тілами у вигляді залежності від часу кута між зазначеними двома осями систем координат, пов'язаних з цими тілами

Linear Driver

Визначає нестационарну в'язь між під'єднаними до блоку тілами у вигляді залежності від часу проекції відстані між початками систем координат, пов'язаних з цими тілами, на зазначену вісь інерціальної системи відліку

Velocity

Визначає нестационарну в'язь між під'єднаними до блоку тілами у ви-

Driver	гляді залежності від часу лінійної комбінації проєкцій векторів лінійних та кутових швидкостей систем координат, пов'язаних з цими тілами, на вказані осі
Point Curve Constraint	Визначає стаціонарну в'язь між під'єднаними до блоку тілами у вигляді заданих своїми координатами в системі координат, пов'язаної з веденим (Follower) тілом, точок кривої відстані початку системи координат, пов'язаної з ведучим (Body) тілом; крива визначається сплайновою інтерполяцією зазначених точок
Parallel Constraint	Визначає таку стаціонарну в'язь між під'єднаними до блоку тілами, що зазначена вісь системи координат, пов'язаної з провідним тілом, залишається увесь час руху паралельною однойменній осі системи координат, пов'язаної з веденим тілом
Gear Constraint	Визначає таку стаціонарну в'язь між під'єднаними до блоку тілами, що визначає стаціонарну в'язь між під'єднаними до блоку тілами як зубчасту передачу з заданими радіусами ділительних кіл

- Примітки. 1. Завдання часової залежності нестационарної зв'язку в блоці Driver здійснюється через під'єднання до останнього блоку Driver Actuator.
2. Якщо до блоку Driver не підключений блок Driver Actuator, він реалізує відповідну стаціонарну в'язь. Наприклад, Distance Driver реалізує рух початку координат тіла Follower по сфері з центром у початку координат тіла Body. При цьому радіус сфери визначається заданим початковим положенням цих систем координат.

12.1.5. Розділ Utilities

Вміст розділу показано на рис. 12.32.

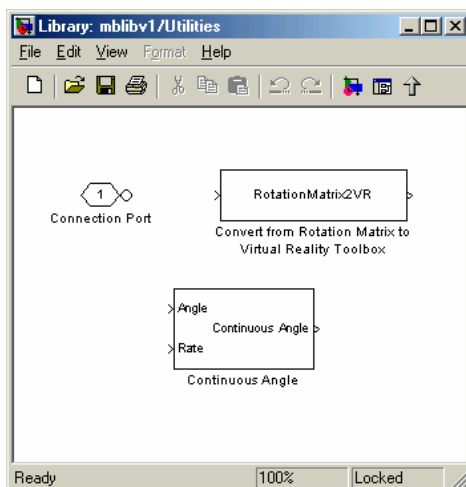


Рис. 12.32. Вміст розділу Utilities

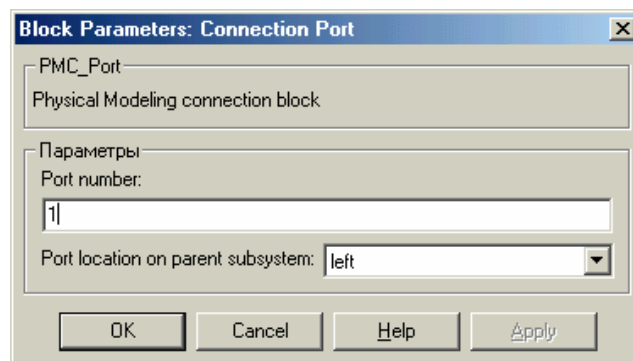


Рис. 12.33. Вікно настроювання блоку Connection Port

Блок **Connection Port** (З'єднувальний порт) грає в моделі SimMechanics ту ж роль, яку відіграють блоки **In** і **Out** у звичайній S-моделі. Завдяки йому, в моделі SimMechanics можна створювати підсистеми, формуючи входи і виходи цими блоками. Вікно настроювання наведено на рис. 12.33. Параметр настроювання один - місце розташування зовнішнього зображення порту праворуч або ліворуч на зображенні підсистеми.

При використанні блоків **Joint Sensor** для вимірювання кута відносного повороту слід мати на увазі, що вимірювач кута видає сигнал, пропорційний

вимірюваному куту лише в діапазоні $\pm \pi$ радіан. При перевищенні цього діапазону значення кута, яке видається, зазнає розрив, рівний 2π радіан. Щоб отримати реальне значення кута повороту в цьому випадку слід використовувати блок **Continuous Angle**. При цьому до числа вимірюваних блоком **Joint Sensor** величин, крім кута, слід додати і швидкість змінювання цього кута і подати відповідний сигнал на вхід **Rate**. Тоді на виході блоку вийде неперервний сигнал кута. Параметрами настроювання блоку **Continuous Angle** (рис. 12.34) є одиниці виміру кута і кутової швидкості.

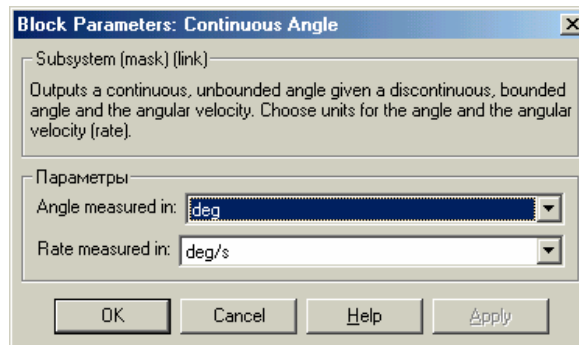


Рис. 12.34. Вікно настроювання блоку *Continuous Angle*

12.2. Модель зрівноваженого вільного гіроскопа

Розглянемо процес побудови S-моделі за допомогою бібліотеки SimMechanics на найпростіших прикладах.

Почнемо з моделі врівноваженого гіроскопа. Під врівноваженим гіроскопом розуміють пристрій, що складається з одного твердого тіла, центр тяжерсті якого нерухомий в інерціальному просторі, а саме тіло може довільно повертатися в просторі щодо цієї нерухомої його точки (точки підвісу), при цьому тілу надано швидкого обертання навколо однієї з осей, жорстко пов'язаних з тілом (осі власного обертання гіроскопа).

Забезпечити три кутові ступені свободи тіла можна за допомогою двох видів зчленувань, передбачених бібліотекою SimMechanics (розділ **Joints**), - **Spherical** (Сферичний шарнір) і **Gimbal** (карданів підвіс). Зручніше використовувати блок **Gimbal**. Цей вид зчленування являє собою поєднання трьох елементарних зчленувань виду **Revolute** (Циліндричний шарнір), кожний з яких забезпечує обертання навколо однієї з трьох взаємно-перпендикулярних осей.

Модель такого гіроскопа представлена на рис. 12.35.

Вона складається з блоку **Ground** інерціальної системи відліку, блоку **Gimbal** забезпечення триступеневої підвісу гіроскопа, блоку **Giroscop** (типу **Body**), трьох блоків **Joint Initial Condition (IC)**, кожен з яких встановлює початкові умови для одного з примітивів R1, R2 і R3 типу **Revolute**, що складають підвіс **Gimbal**, і трьох вимірювачів **Joint Sensor**, кожен з яких приєднаний до одного з зазначених примітивів і вимірює кут його відносного повороту.

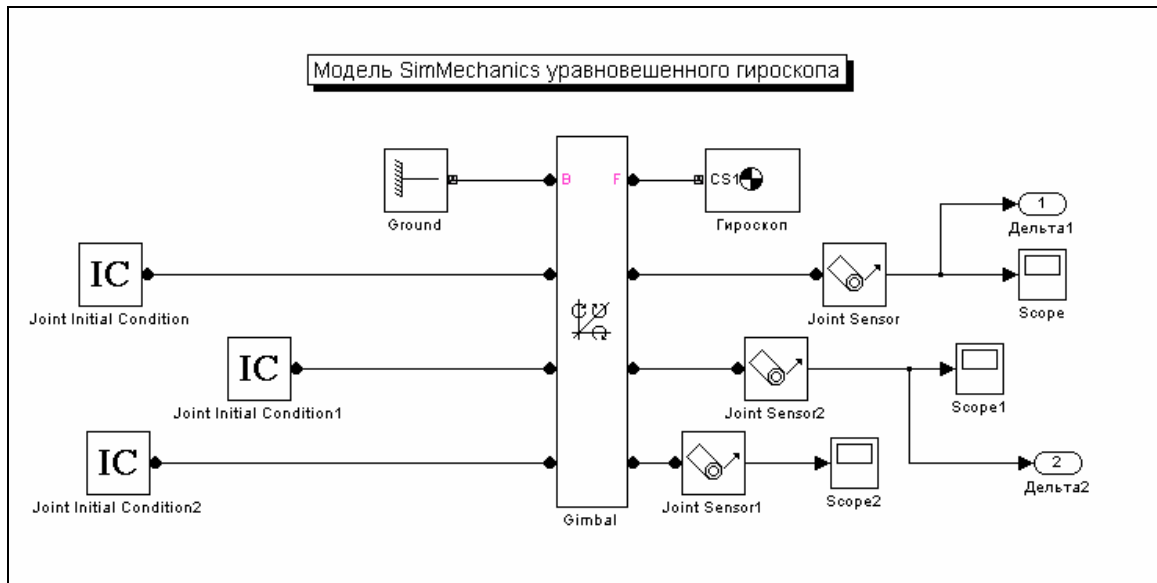


Рис. 12.35. Модель SimMechanics зрівноваженого гіроскопа

Для зв'язку з робочим простором передбачені вихідні порти Дельта1 і Дельта2, на які надходять сигнали, пропорційні поточним значенням кутів повороту гіроскопа навколо осей Z і X відповідно.

Позначимо:

m	маса гіроскопа
J	матриця моментів інерції гіроскопа
Ω	власна кутова швидкість гіроскопа (навколо осі Y)
ω_x, ω_z	початкові кутові швидкості гіроскопа навколо осей X і Z відповідно
$\text{delta10}, \text{delta20}$	початкові кути відхилення осі власного обертання гіроскопа від осі Y інерціальної системи координат (навколо осей Z і X відповідно)

Нижче (рис. 12.36 ... 12.39) наведені вікна настроювання основних блоків

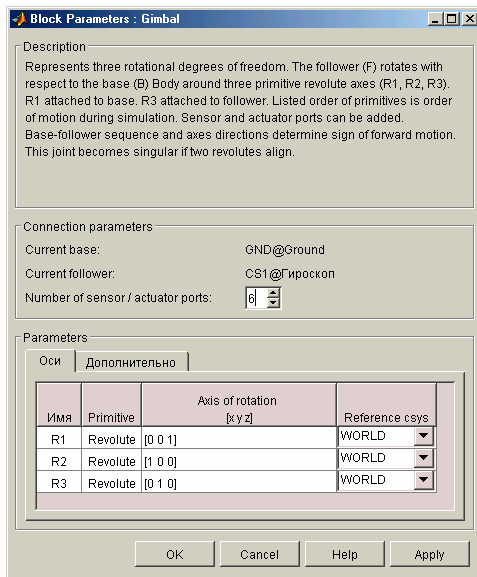


Рис. 12.36. Вікно налаштування блоку Gimbal

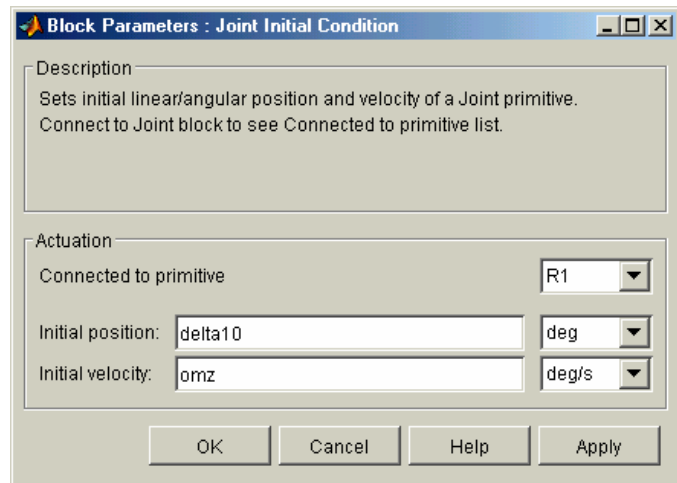


Рис. 12.37. Вікно налаштування блоку Joint Initial Condition

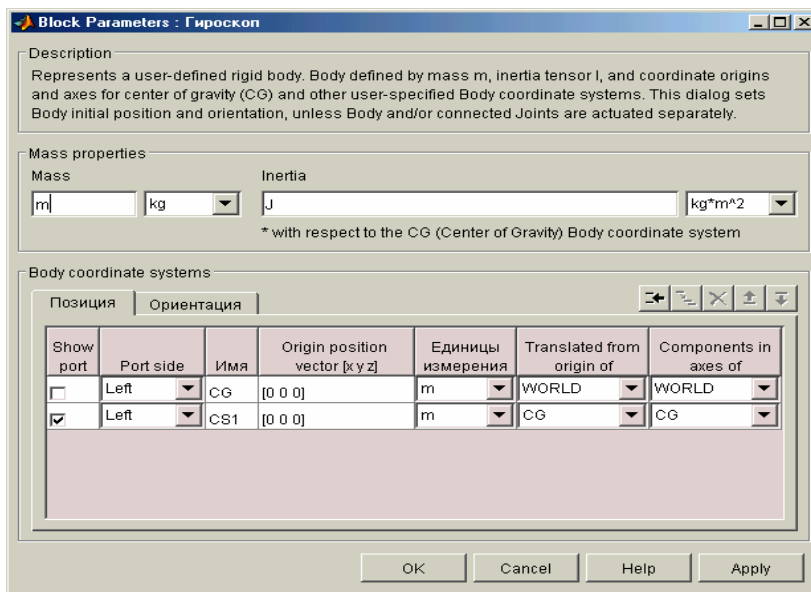


Рис. 12.38. Вікно налаштування блоку Гіроскоп

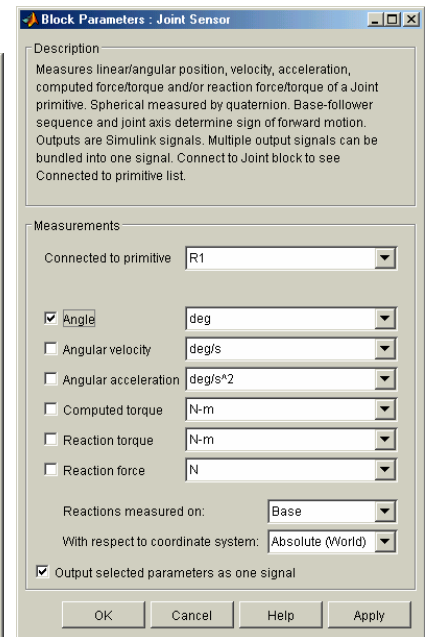


Рис. 12.39. Вікно налаштування блоку Joint Sensor

Текст керуючої програми *SUG_simMech_upr*, яка здійснює присвоєння значень інерційним характеристикам гіроскопа, введення початкових умов, запуск на виконання S-моделі і виведення в графічній формі результатів на екран, наведений нижче

```
% SUG_simMech_upr
```

```
% Лазарев Ю. Ф. 19-04-2004
```

```
clc
```

```
clear all
```

```
% Установка инерционных характеристик гироскопа
```

```
m=1;
```

```
J=[3 0 0;0 5 0;0 0 3]; %J=[4 0 0;0 5 0;0 0 3]; %J=[4 -0.2 0.1;-0.2 5 -0.2;0.1 -0.2 3];
```

```

% Установка начальных условий
OM=20; omx=1; omz=0; delta10=0; delta20=0;
% Моделирование на модели SimMechanics
sim('SUG_simMech');
% Извлечение данных
t=tout; D1=yout(:,1); D2=yout(:,2);
% Построение графического вывода результатов
subplot(2,2,1)
plot(D1,D2), grid
set(gca,'FontSize',12)
title('Траектория в картинной плоскости');
xlabel('\delta1 (градусы)'), ylabel('\delta2 (градусы)')
subplot(2,2,2)
axis('off');
h=text(-0.3,1.1,'Уравновешенный гироскоп (модель SimMechanics)','FontSize',14);
h=text(0.1,0.9,'I', 'FontSize',12);
h=text(0.2,0.9,num2str(J(1,1)),'FontSize',12);
h=text(0.4,0.9,num2str(J(1,2)),'FontSize',12);
h=text(0.6,0.9,num2str(J(1,3)),'FontSize',12);
h=text(0.8,0.9,'J', 'FontSize',12); h=text(-0.1,0.8,'J = ', 'FontSize',12);
h=text(0.1,0.8,'I', 'FontSize',12);
h=text(0.2,0.8,num2str(J(2,1)),'FontSize',12);
h=text(0.4,0.8,num2str(J(2,2)),'FontSize',12);
h=text(0.6,0.8,num2str(J(2,3)),'FontSize',12);
h=text(0.8,0.8,'J', 'FontSize',12);
h=text(0.1,0.7,'I', 'FontSize',12);
h=text(0.2,0.7,num2str(J(3,1)),'FontSize',12);
h=text(0.4,0.7,num2str(J(3,2)),'FontSize',12);
h=text(0.6,0.7,num2str(J(3,3)),'FontSize',12);
h=text(0.8,0.7,'J', 'FontSize',12);
h=text(-0.1,0.5,'Начальные углы (градусы)','FontSize',12);
h=text(0.1,0.4,['\delta10 = ',num2str(delta10)],'FontSize',12);
h=text(0.4,0.4,['\delta20 = ',num2str(delta20)],'FontSize',12);
h=text(-0.1,0.2,'Начальные угловые скорости (рад/с)','FontSize',12);
h=text(0.1,0.1,['omx0 = ',num2str(omx)],'FontSize',12);
h=text(0.4,0.1,['omy0 = ',num2str(OM)],'FontSize',12);
h=text(0.7,0.1,['omz0 = ',num2str(omz)],'FontSize',12);
h=text(-0.1,-0.05,'-----');
h=text(-0.1,-0.1,'Программа SUG-simMech-upr Лазарев Ю. Ф. 19-04-2004');
h=text(-0.1,-0.15,'-----');

subplot(2,2,[3,4])
plot(t,D1,t,D2,'. '), set(gca,'FontSize',12)
title('Изменение углов поворота оси гироскопа со временем');
xlabel('Время (сек)'), ylabel('Углы (градусы)')
legend('\delta1', '\delta2',0), grid

```

Далі наводяться результати роботи цієї програми.

На рис. 12.40 подані результати моделювання для випадку, коли гіроскоп є динамічно симетричним тілом з віссю симетрії, збіжній з віссю власного обертання.

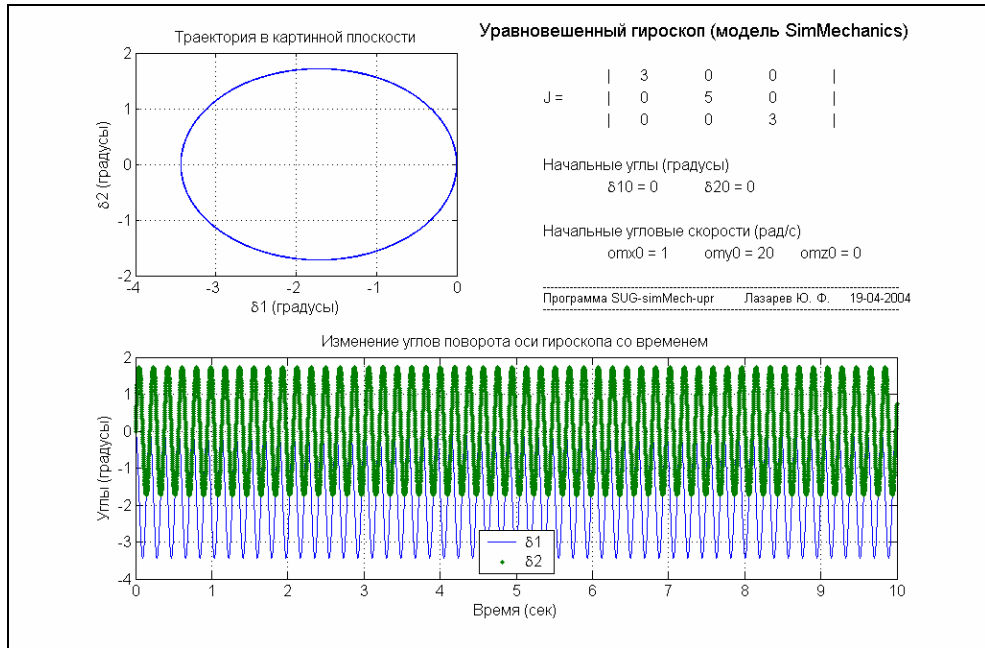


Рис. 12.40. Вільний рух симетричного зрівноваженого гіроскопа

Рис. 12.41 показує результати для випадку несиметричного гіроскопа. На рис. 12.42 подано рух несиметричного і динамічно незбалансованого гіроскопа.

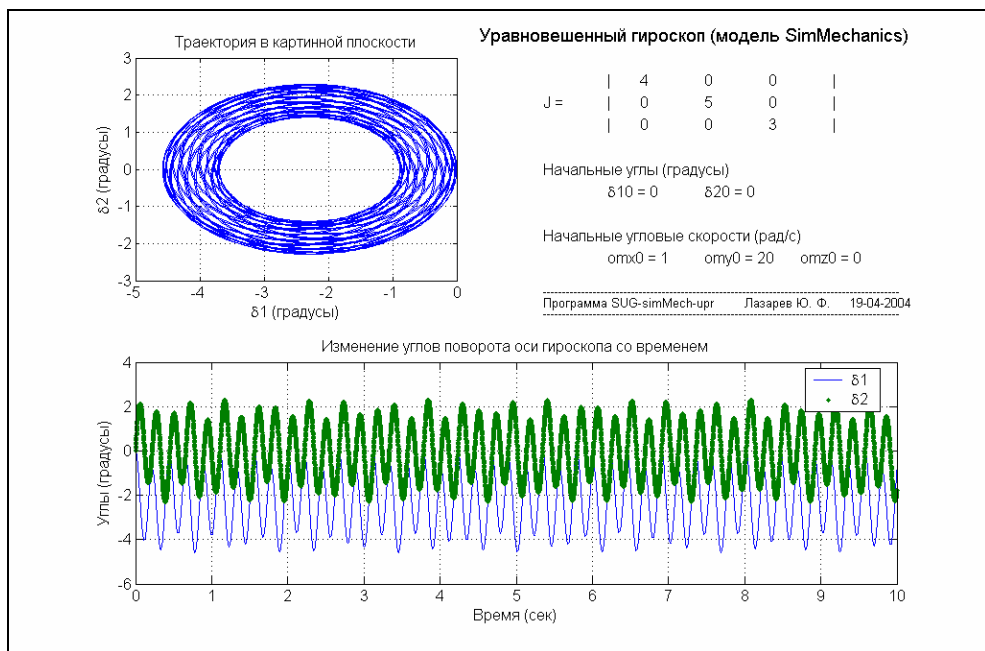


Рис. 12.41. Вільний рух несиметричного зрівноваженого гіроскопа

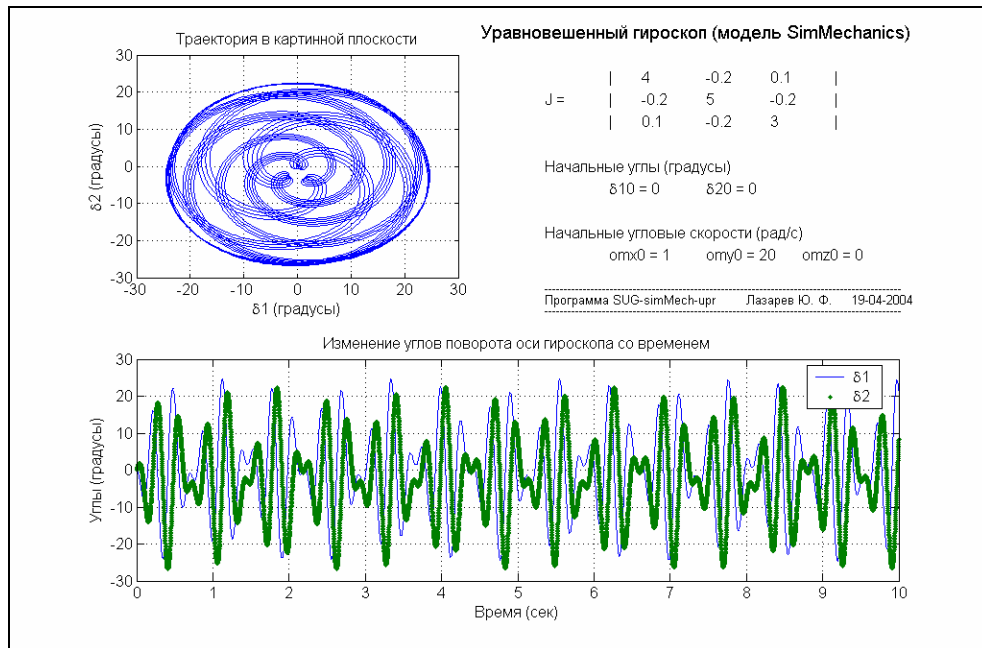


Рис. 12.42. Вільний рух незбалансованого гіроскопа

Результати моделювання добре узгоджуються з результатами теоретичного аналізу [14].

12.3. Модель кривошипно-шатунного механізму

Перейдемо до складання моделі кривошипно-шатунного механізму. Він складається з провідної ланки (повідця, кривошипа), яка обертається із заданою кутовою швидкістю навколо осі Z , веденої ланки (шатуна) і повзуна, який переміщується поступально в напрямних, паралельних осі X . Всі три тіла разом утворюють замкнений механічний ланцюг.

Модель може бути реалізована у вигляді схеми, зображеної на рис. 12.43.

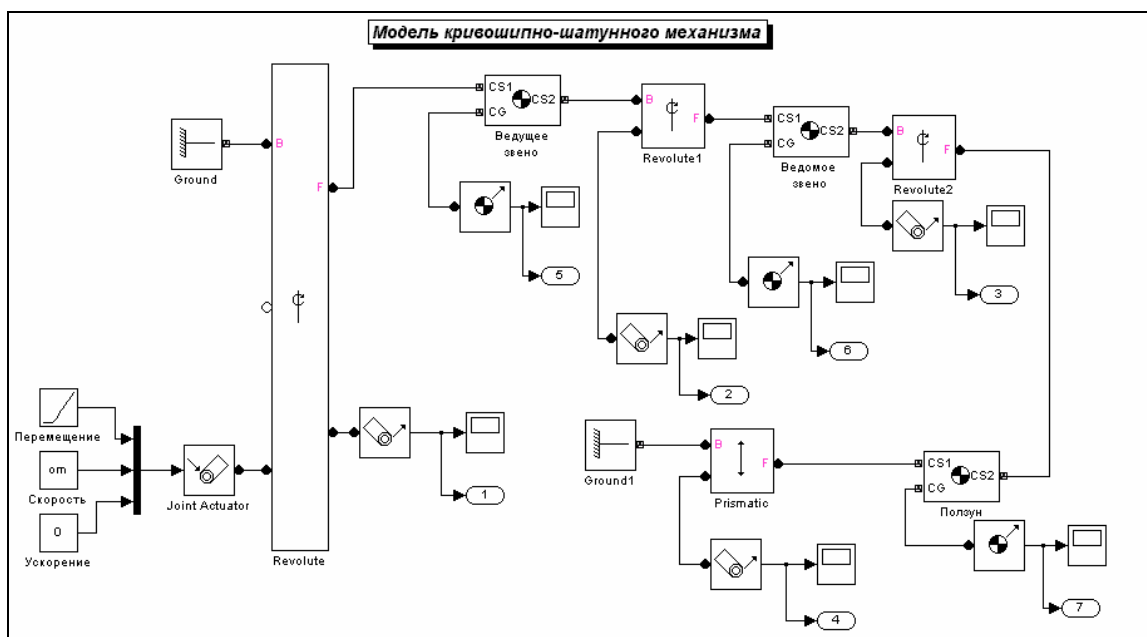


Рис. 12.43. Модель SimMech_KSM1 кривошипно-шатунного механізму

Вона складається з двох блоків типу **Ground**, двох блоків типу **Body** (Ведуще звено, Ведомое звено і Ползун), чотирьох блоків примітивів зчленувань (три блоки **Revolute** і один блок **Prismatic**), блоку **Joint Actuator** збудника обертання провідної ланки, чотирьох блоків типу **Joint Sensor** для вимірювання відносна руху частин чотирьох зчленувань і трьох блоків типу **Body Sensor** для вимірювання параметрів руху тіл.

Надалі використовуємо такі позначення

- m1, m2, m3** Маса провідної, веденої ланок і повзуна відповідно
J1, J2, J3 Матриці моментів інерції зазначених тіл щодо їх центрів мас
L1 Відстань від осі обертання кривошипа до шарніра, що зв'язує його з шатуном
L2 Відстань між двома шарнірами шатуна (веденої ланки) X
om Кутова швидкість обертання кривошипа навколо осі Z
fi0 Початковий кут відхилення кривошипа від горизонтальної осі
E Найкоротша відстань від осі обертання кривошипа до осі переміщення повзуна
A1 Початкове значення кута нахилу осі шатуна до осі X

С врахуванням їх встановлені параметри настроювання блоків тіл, показані на рис. 12.44...12.48.

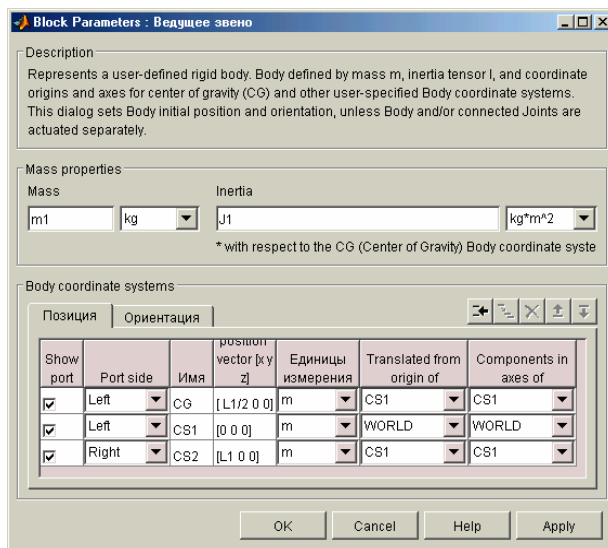


Рис. 12.44. Вікно настроювання блоку Ведуще звено (вкладка *Позиция*)

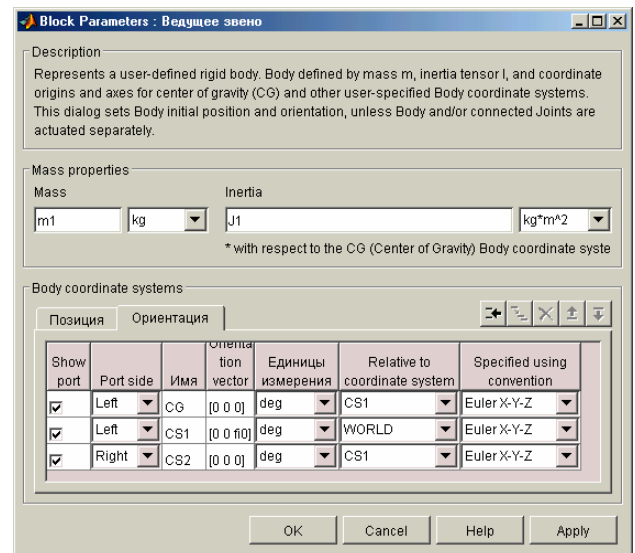


Рис. 12.45. Вікно настроювання блоку Ведуще звено (вкладка *Ориентация*)

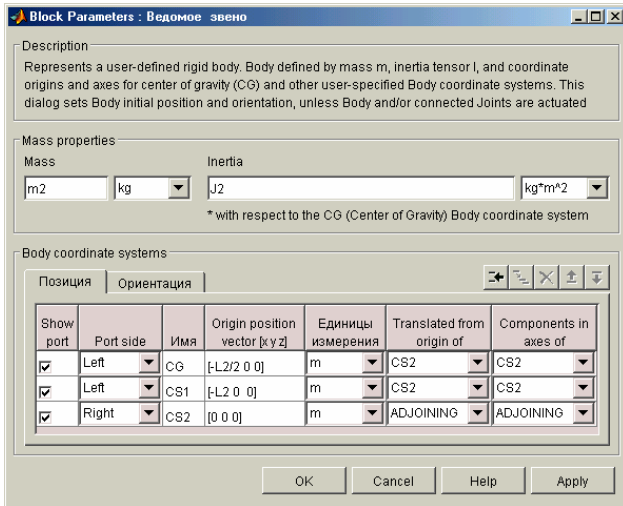


Рис. 12.46. Вікно налаштування блоку Ведомое звено (вкладка Позиция)

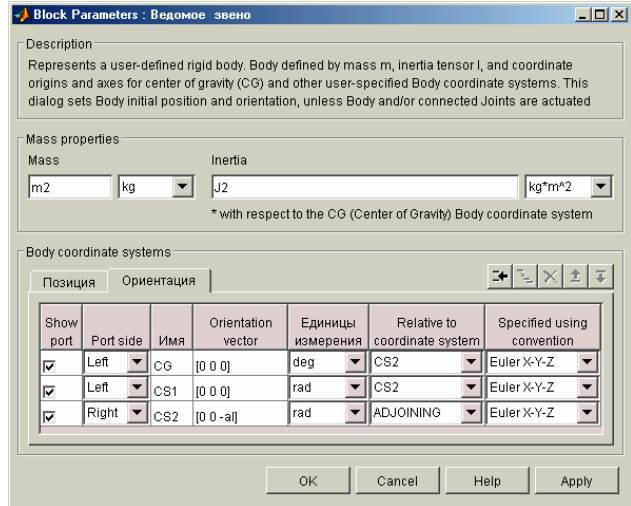


Рис. 12.47. Вікно налаштування блоку Ведомое звено (вкладка Ориентация)

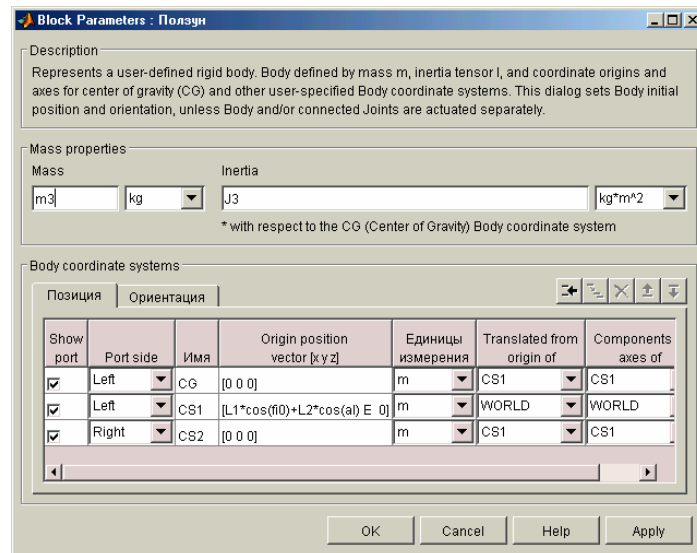


Рис. 12.48. Вікно налаштування блоку Ползун

Програма *SimMech_KSM1_upr* керування роботою цієї моделі наведена НИЖЧЕ.

```
% SimMech_KSM1_upr
% Лазарев Ю. Ф. 6-05-2004

clear all,      clc
% Задание параметров ведущего звена
m1=1; J1=[1 0 0;0 1 0; 0 0 1]; L1=0.5; fi0=0; om=5;
% Задание параметров ведомого звена
m2=1; J2=[1 0 0;0 1 0; 0 0 1]; L2=1;
% Задание параметров ползуна
m3=1; J3=[1 0 0;0 1 0; 0 0 1]; E=0.1;
% Предварительные расчеты
al=asin((L1*sin(fi0)-E)/L2);
% Моделирование
sim('SimMech_KSM1')
% Получение данных
t=tout;
% 1. С приводного шарнира
```

```

Ug1=unwrap(yout(:,1)*pi/180)*180/pi; UgSk1=yout(:,2); UgUsk1=yout(:,3);
Mc1=yout(:,4);
% 2. С ведомого шарнира
Ug2=unwrap(yout(:,5)*pi/180)*180/pi; UgSk2=yout(:,6); UgUsk2=yout(:,7);
Mc2=yout(:,8);
% 3. С шарнира ползуна
Ug3=yout(:,9); UgSk3=yout(:,10); UgUsk3=yout(:,11);
Mc3=yout(:,12);
% 4. С направляющей ползуна
Xp=yout(:,13); Skp=yout(:,14); Uskp=yout(:,15);
% 5. С центра масс ведущего звена
Xz1=yout(:,16); Yz1=yout(:,17); SkXz1=yout(:,19); SkYz1=yout(:,20);
UgSkZ1=yout(:,24); UgUskZ1=yout(:,27);
% 6. С центра масс ведомого звена
Xz2=yout(:,28); Yz2=yout(:,29); SkXz2=yout(:,31); SkYz2=yout(:,32);
UgSkZ2=yout(:,36); UgUskZ2=yout(:,39);
% 7. С центра масс ползуна
Xz3=yout(:,40); Yz3=yout(:,41); SkXz3=yout(:,43); SkYz3=yout(:,44);
UgSkZ3=yout(:,48); UgUskZ3=yout(:,51);
% Вывод графиков
subplot(2,3,1)
plot(t,Ug1,'o',t,Ug1+Ug2,'.',t,Ug3), grid, title('Углы поворотов шарниров')
xlabel('Время (с)'), ylabel('Градусы')
h=text(0.0,1200,'Кривошипно-шатунный механизм','FontSize',14);
legend('Шарнир1','Шарнир1-2','Шарнир3',0)
subplot(2,3,4)
plot(t,Mc1,'o'), grid, title('Момент сил на ведущем шарнире')
xlabel('Время (с)'), ylabel('Ньютон*м')
subplot(2,3,2)
plot(t,UgSk1,'o',t,UgSk2,'.',t,UgSk3), grid
title('Угловые скорости поворотов шарниров')
xlabel('Время (с)'), ylabel('Градусы в секунду')
legend('Шарнир1','Шарнир2','Шарнир3',0)
subplot(2,3,3)
plot(t,UgSkZ1,'o',t,UgSkZ2,'.',t,UgSkZ3), grid
title('Угловые ускорения поворотов шарниров')
xlabel('Время (с)'), ylabel('Градусы в секунду^2')
legend('Шарнир1','Шарнир2','Шарнир3',0)
subplot(2,3,5)
plot(t,Xp,'r'), grid, title('Отн. перемещение призматич. сочленения')
xlabel('Время (с)'), ylabel('Метры')
subplot(2,3,6)
plot(t,Skp,'.',t,Uskp), grid, title('Скорость и ускорение ползуна')
xlabel('Время (с)'), ylabel('М/с и м/с^2'), legend('скорость','Ускорение',0)

figure
subplot(2,3,1)
plot(t,Xz1,'o',t,Xz2,'.',t,Xz3), grid, title('Перемещения X ц.м. звеньев')
xlabel('Время (с)'), ylabel('Метры'), legend('ведущее','ведомое','ползун',1)
subplot(2,3,3)
plot(t,UgSkZ1,'o',t,UgSkZ2,'.',t,UgSkZ3), grid,
title('Угловая скорость ведущего звена')
xlabel('Время (с)'), ylabel('Градусы в сек.')
subplot(2,3,4)
plot(t,Yz1,'o',t,Yz2,'.',t,Yz3), grid, title('Перемещения Y ц.м. звеньев')
xlabel('Время (с)'), ylabel('Метры')
subplot(2,3,2)
plot(t,SkXz1,'o',t,SkXz2,'.',t,SkXz3), grid, title('Скорость Vx звеньев')
xlabel('Время (с)'), ylabel('Градусы в сек.')
subplot(2,3,5)
plot(t,SkYz1,'o',t,SkYz2,'.',t,SkYz3), grid, title('Скорость Vy звеньев')
xlabel('Время (с)'), ylabel('Градусы в сек.')
subplot(2,3,6)
plot(t,UgUskZ1,'o',t,UgUskZ2,'.',t,UgUskZ3), grid
title('Угловые ускорения звеньев'), xlabel('Время (с)'), ylabel('Градусы в сек.')

```

Результати моделювання по цій програмі подані на рис. 12.49 і 12.50.

Як бачимо, засоби бібліотеки SimMechanics дозволяють отримати достатньо обширну інформацію про механічний рух тіл, як відносний, так і абсолютний, включаючи реакції у в'язях між тілами. Остання обставина особливо важлива при проектуванні механізмів і машин.

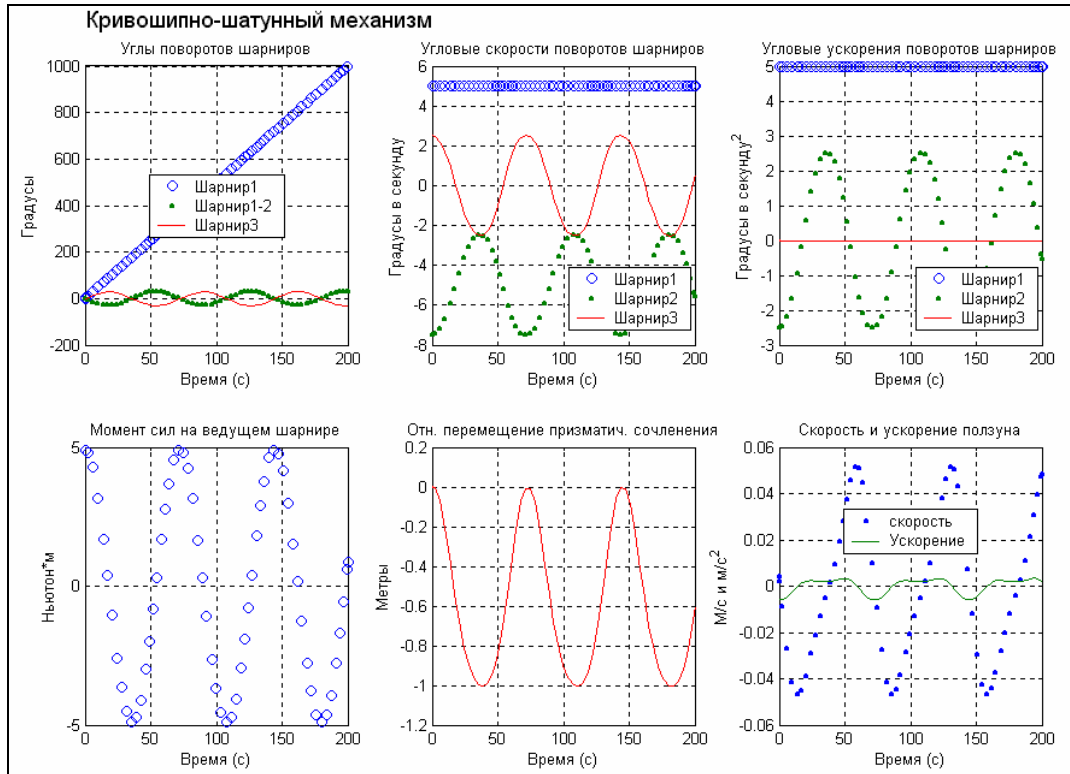


Рис. 12.49. Результати моделювання за моделлю SimMech_KSM1

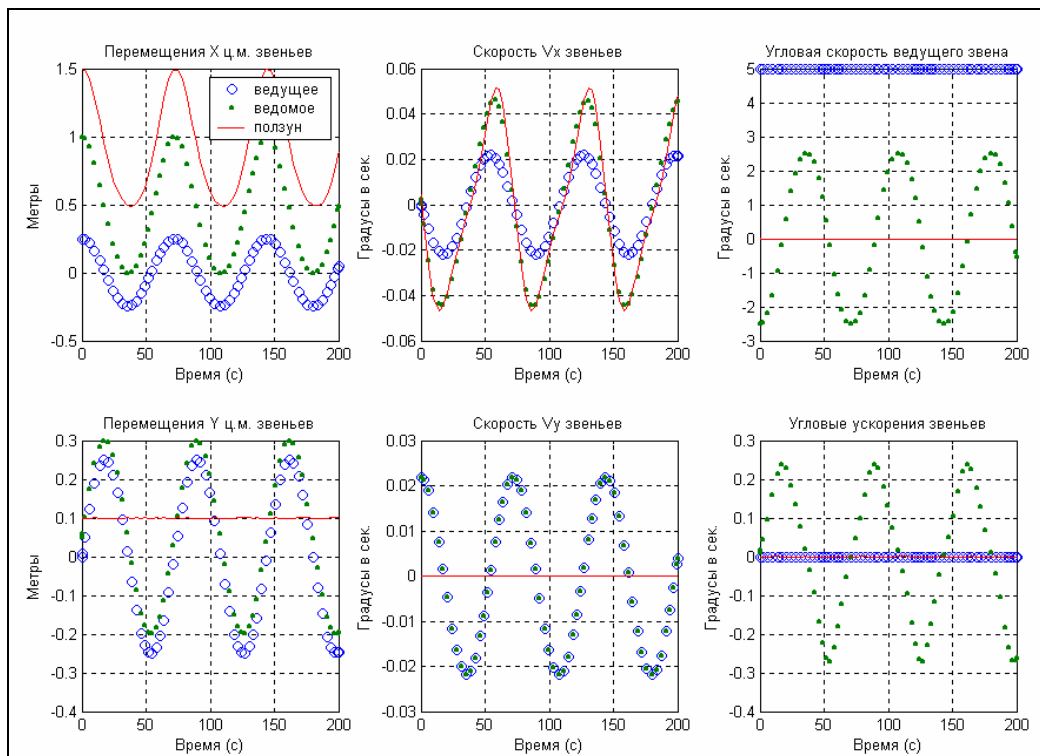


Рис. 12.50. Результати моделювання кривошипно-шатунного механізму

12.4. Модель руху маятника

Розглянемо модель руху маятника на поступально віброуючій основі, поведження якого було досліджено раніше іншими засобами.

Для початку відзначимо, що за замовчуванням моделі бібліотеки SimMechanics автоматично враховують дію сили тяжіння на всі ланки механізму. При цьому передбачається, що вісь Y земної (інерціальної) системи координат напрямлена вздовж вертикалі вгору, а, отже, сила тяжіння, прикладена в центрі мас, спрямована по ній у протилежному напрямку (вниз). Тому осі Z і X лежать у площині горизонту.

Надалі будемо припускати, що вісь обертання маятника напрямлена уздовж осі Z , а коливання маятника відбуваються у площині YX .

Для імітації поступального руху основи у цій площині зв'яжемо (рис. 12.51) тіло маятника (блок *Тело*) з інерціальною системою відліку (блок *Основание*) через зчленування *Planar*, яке являє собою ланцюг з трьох послідовно з'єднаних примітивів – P1, P2 і R3.

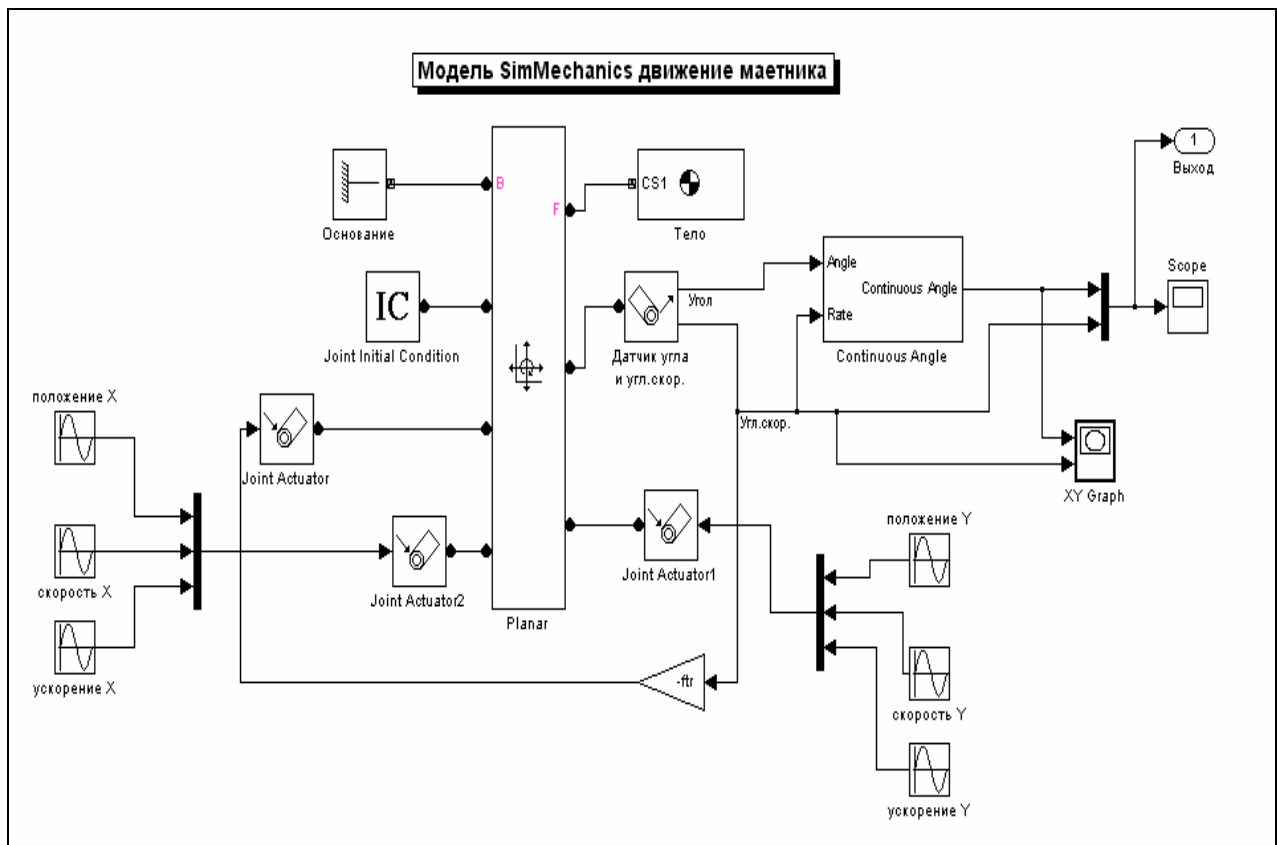


Рис. 12.51. Блок-схема моделі SimMech_FMr маятника

Перші два (типу *Prismatic*) реалізують поступальний рух основи відповідно уздовж осей X і Y (див рис. 12.52). Третій примітив (типу *Revolute*) реалізує обертальну ступінь вільності маятника навколо осі Z .

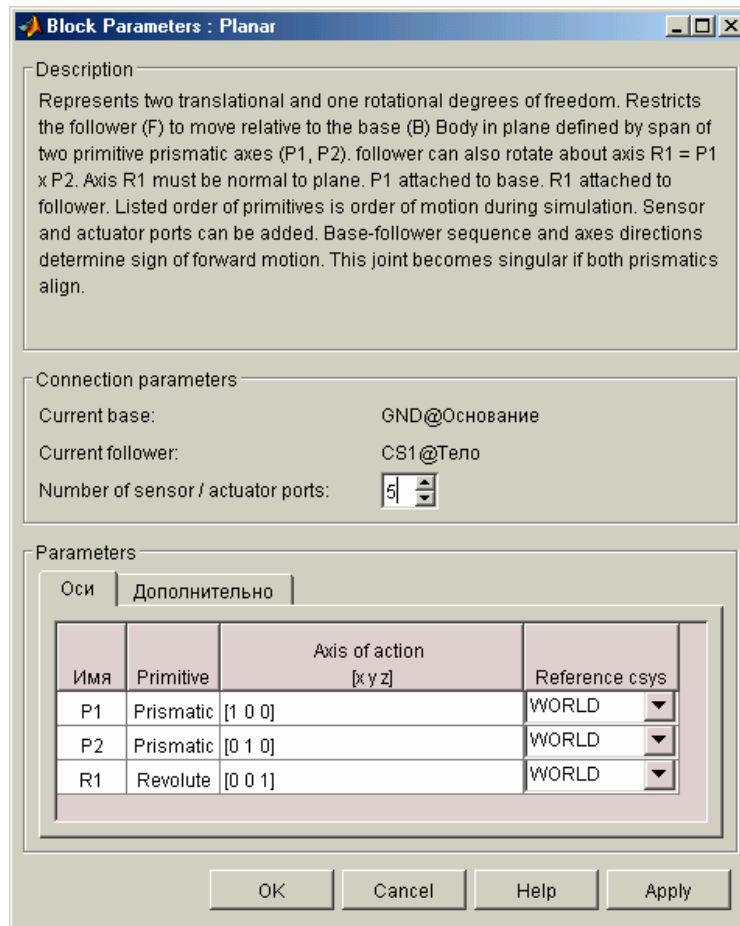


Рис. 12.52. Налаштування блоку Planar

Для задання параметрів використані такі позначення.

m	Маса маятника
J	Матриця моментів інерції маятника відносно його центру мас
l	Зміщення центру мас маятника відносно осі його обертання
ftr	Коефіцієнт в'язкого тертя в осі обертання маятника
fi0	Початковий кут відхилення маятника від вертикалі
fit0	Початкова кутова швидкість маятника
nxm, nym	Амплітуди віброперевантажень основи відповідно в горизонтальному і вертикальному напрямках
om	Частота коливань основи
erx, ery	Початкові фази коливань основи
g	Прискорення сили тяжіння

Введення початкових умов руху маятника здійснюється блоком **Joint Initial Condition**, налаштування якого наведено на рис. 12.53.

Вікно налаштування блоку Тіло показані на рис. 12.54.

Збудження коливань основи здійснюється двома блоками **Joint Actuator2** (по осі X) і **Joint Actuator1** (по осі Y). На рис. 12.55 відображені налаштування одного з них.

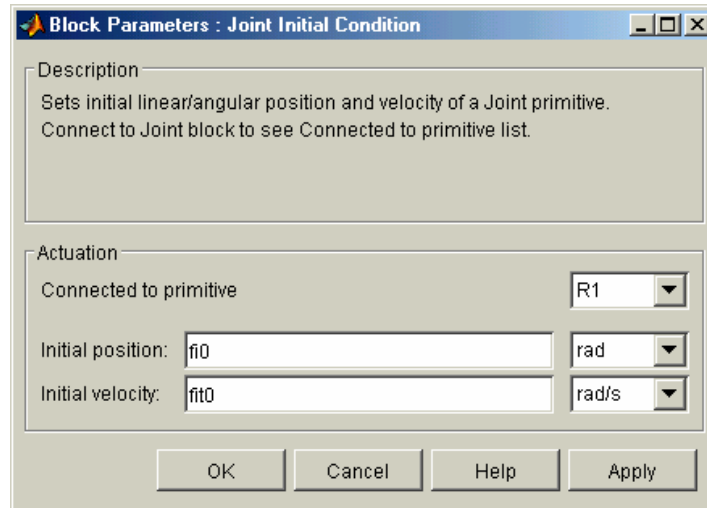


Рис. 12.53. Налаштування блоку Joint Initial Condition

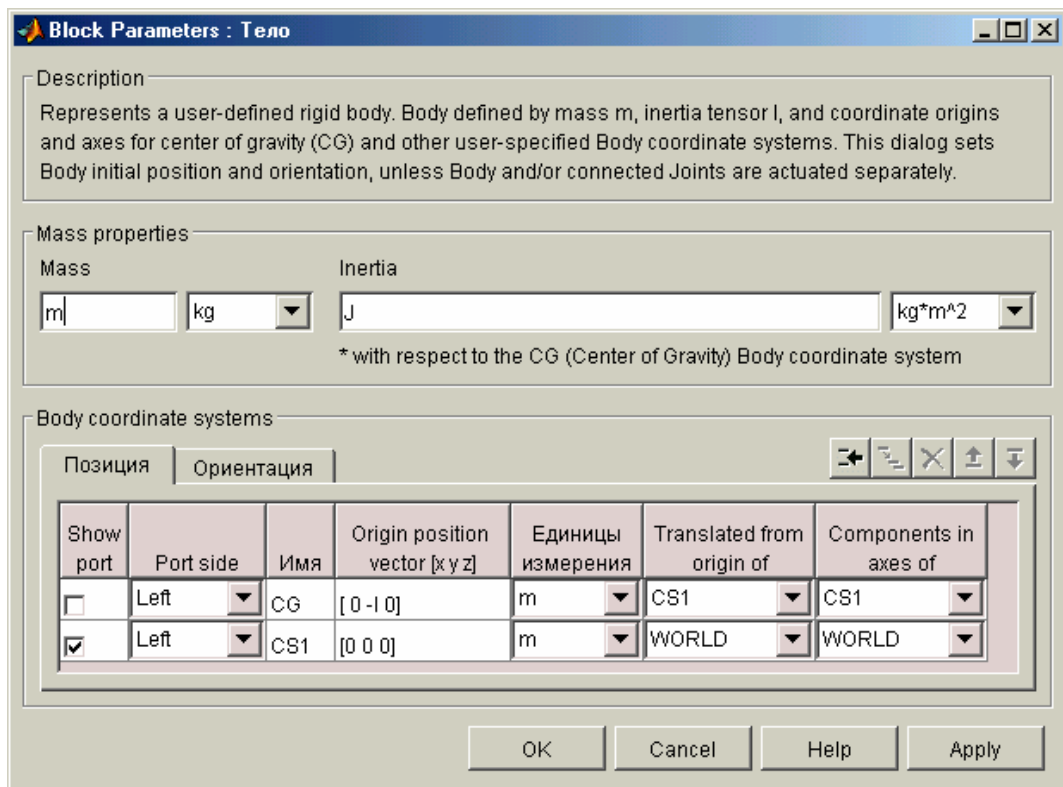
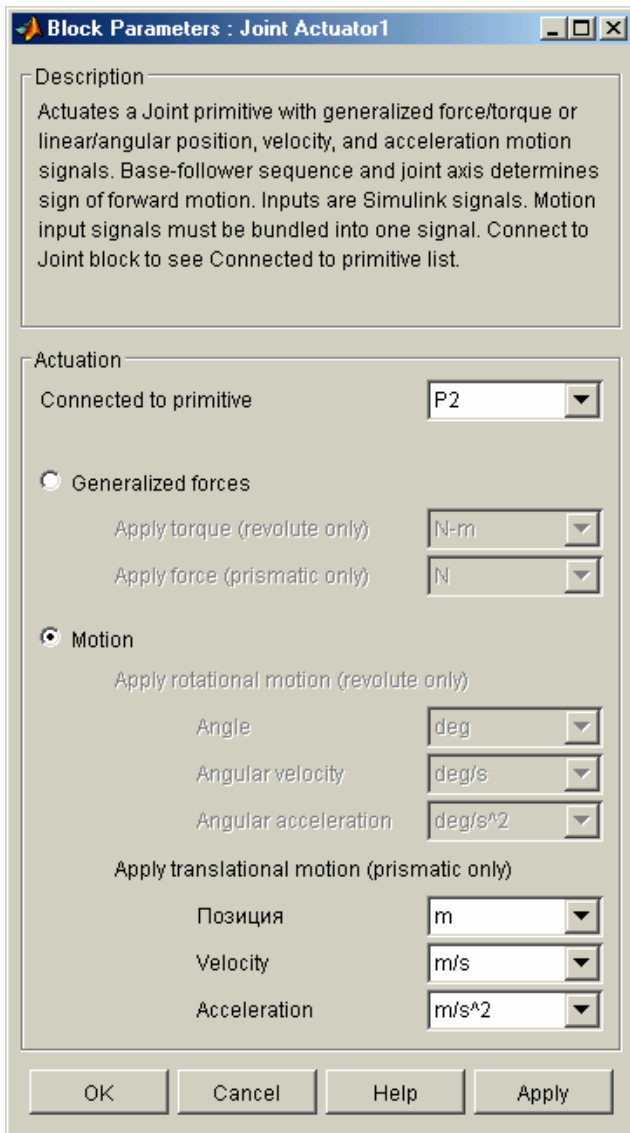
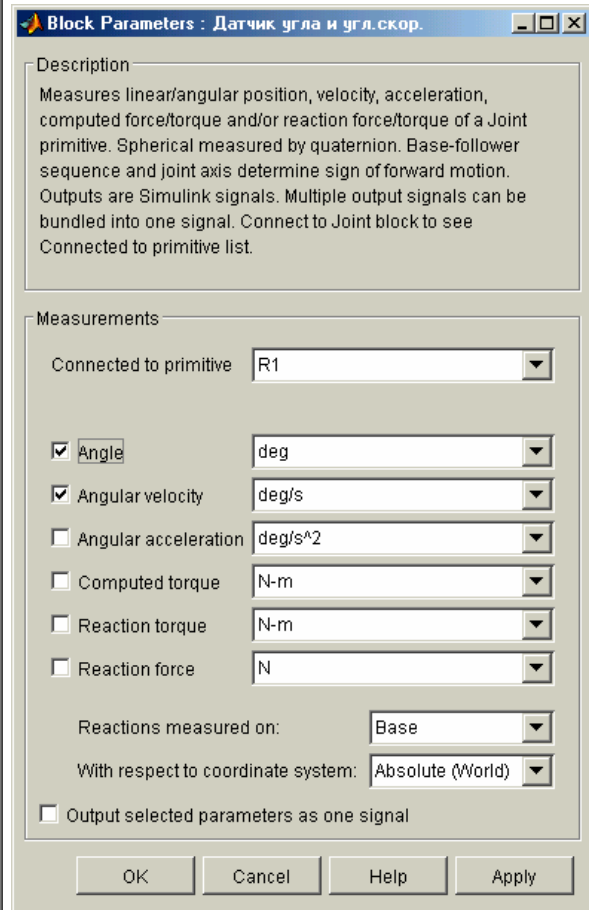


Рис. 12.54. Налаштування блоку Тело

Вимірювання кута відхилення маятника від вертикалі та його кутової швидкості здійснюється за допомогою блоку *Датчик угла и угл. скор.* типу *Joint Sensor*, налаштування якого наведені на рис. 12.56.

Рис. 12.55. Налаштування блоку *Joint Actuator1*Рис. 12.56. Налаштування блоку *Датчик угла и угл. скор.*

Для отримання безперервного і більшого за 180° за модулем кута повороту маятника використаний блок *Continuous Angle*.

Сигнал кутової швидкості, отриманий на виході блоку *Датчик кута і кут. скор.* використовується для формування моменту сил в'язкого тертя на осі обертання маятника за допомогою блоку *Joint Actuator*, налаштування якого наведені на рис. 12.57.

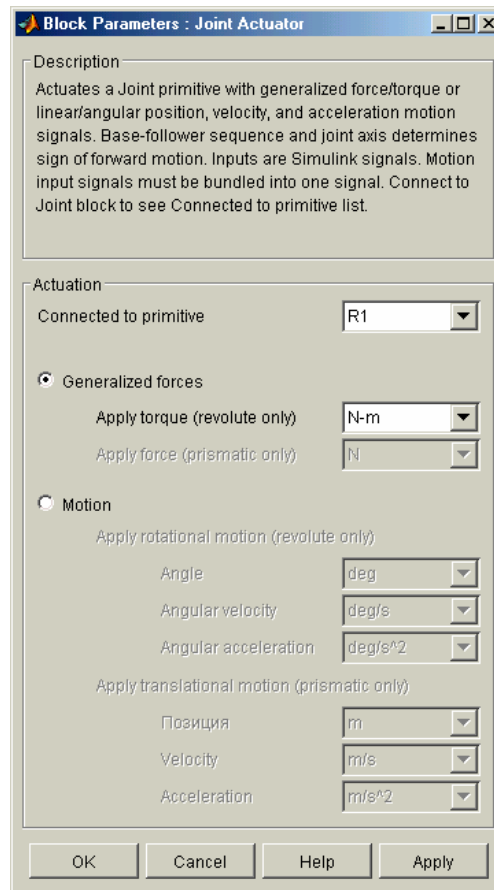


Рис. 12.57. Налаштування блоку Joint Actuator

Управління введенням даних, запуском моделі і виведенням результатів можна здійснити за допомогою керувальної програми *SimMech_FM_upr*, приведеною нижче.

```
% SimMech_FM_upr
% Управляющая программа для S-модели SimMech_FMr

% Лазарев Ю. Ф., 11-05-2004

clear all, clc
% Ввод параметров маятника
m=1; Jx=0.01; Jy=0.01; Jz=0.02; I=0.5;
ftr= 0.0%05;
J=[Jx 0 0;0 Jy 0;0 0 Jz];
% Параметры внешних воздействий
pxm=0; om=10; ерх=0;
пум=0; om=21.5; еру=0; g=9.81;
% Ввод начальных условий
fi0=179.9*pi/180; fit0=0;
% Моделирование на S-модели
sim('SimMech_FMr')
% Получение результатов
t=tout; fi=yout(:,1); fit=yout(:,2);
% Вывод графиков
subplot(2,2,1)
plot(fi*pi/180,fit*pi/180), grid
xlabel('Угол (радиан)'), ylabel('Угл. скорость (рад/с)')
set(gca,'FontSize',12), title('Фазовый портрет')
subplot(2,2,[3 4])
plot(t,fi), grid
xlabel('Время (секунды)'), ylabel('Угол (градусы)')
```

```

set(gca,'FontSize',12), title('Отклонение от вертикали')
subplot(2,2,2)
axis('off');
h=text(0.1,1.1,'Маятник (модель SimMechanics)','FontSize',14);
h=text(-0.2,1.0,['Масса (кг) m= ',num2str(m)],'FontSize',12);
h=text(0.5,1.0,['Смещение ц.м (м) L= ',num2str(l)],'FontSize',12);
h=text(-0.2,0.9,'Матрица моментов инерции относительно центра масс (кг м^2)','FontSize',12);
h=text(0.1,0.8,'| ','FontSize',12);
h=text(0.2,0.8,num2str(J(1,1)), 'FontSize',12);
h=text(0.4,0.8,num2str(J(1,2)), 'FontSize',12);
h=text(0.6,0.8,num2str(J(1,3)), 'FontSize',12);
h=text(0.8,0.8,'| ','FontSize',12);
h=text(-0.1,0.7,'J = ','FontSize',12);
h=text(0.1,0.7,'| ','FontSize',12);
h=text(0.2,0.7,num2str(J(2,1)), 'FontSize',12);
h=text(0.4,0.7,num2str(J(2,2)), 'FontSize',12);
h=text(0.6,0.7,num2str(J(2,3)), 'FontSize',12);
h=text(0.8,0.7,'| ','FontSize',12);
h=text(0.1,0.6,'| ','FontSize',12);
h=text(0.2,0.6,num2str(J(3,1)), 'FontSize',12);
h=text(0.4,0.6,num2str(J(3,2)), 'FontSize',12);
h=text(0.6,0.6,num2str(J(3,3)), 'FontSize',12);
h=text(0.8,0.6,'| ','FontSize',12);
h=text(-0.2,0.5,'Начальный угол (градусы)','FontSize',12);
h=text(0.7,0.5,['\phi_0 = ',num2str(phi0*180/pi)], 'FontSize',12);
h=text(-0.2,0.4,'Начальная угловая скорость (рад/с)','FontSize',12);
h=text(0.7,0.4,['\phi''_0 = ',num2str(phi0)], 'FontSize',12);
h=text(-0.2,0.3,'К-нт вязкого трения (Н м с)','FontSize',12);
h=text(0.7,0.3,['ftr = ',num2str(ftr*180/pi)], 'FontSize',12);
h=text(-0.2,0.2,'Движение основания:', 'FontSize',12);
h=text(0.5,0.2,['om = ',num2str(om)], 'FontSize',12);
h1=text(-0.2,0.1,' вдоль горизонтали', 'FontSize',12);
h=text(0.5,0.1,['nxm = ',num2str(nxm)], 'FontSize',12);
h=text(0.8,0.1,['epx = ',num2str(epx)], 'FontSize',12);
h2=text(-0.2,0.0,' вдоль вертикали', 'FontSize',12);
h=text(0.5,0.0,['nym = ',num2str(nym)], 'FontSize',12);
h=text(0.8,0.0,['epy = ',num2str(epy)], 'FontSize',12);
h=text(-0.1,-0.05,'-----');
h=text(-0.1,-0.1,'Программа SimMech-FM-upr Лазарев Ю. Ф. 26-04-2004');
h=text(-0.1,-0.15,'-----');

```

Далі наведено два приклади роботи програми та моделі. На рис. 12.58 зображені результати роботи моделі для вільних (без тертя і коливань основи) коливань маятника з амплітудою, близькою до 180° .

Рис. 12.59 презентує результати моделювання маятника при інтенсивній вертикальній вібрації основи та ілюструє стійкість за цих умов верхнього положення рівноваги маятника.

Нарешті, на рис. 12.60 проілюстрований випрямний ефект маятника при інтенсивній горизонтальній вібрації основи.

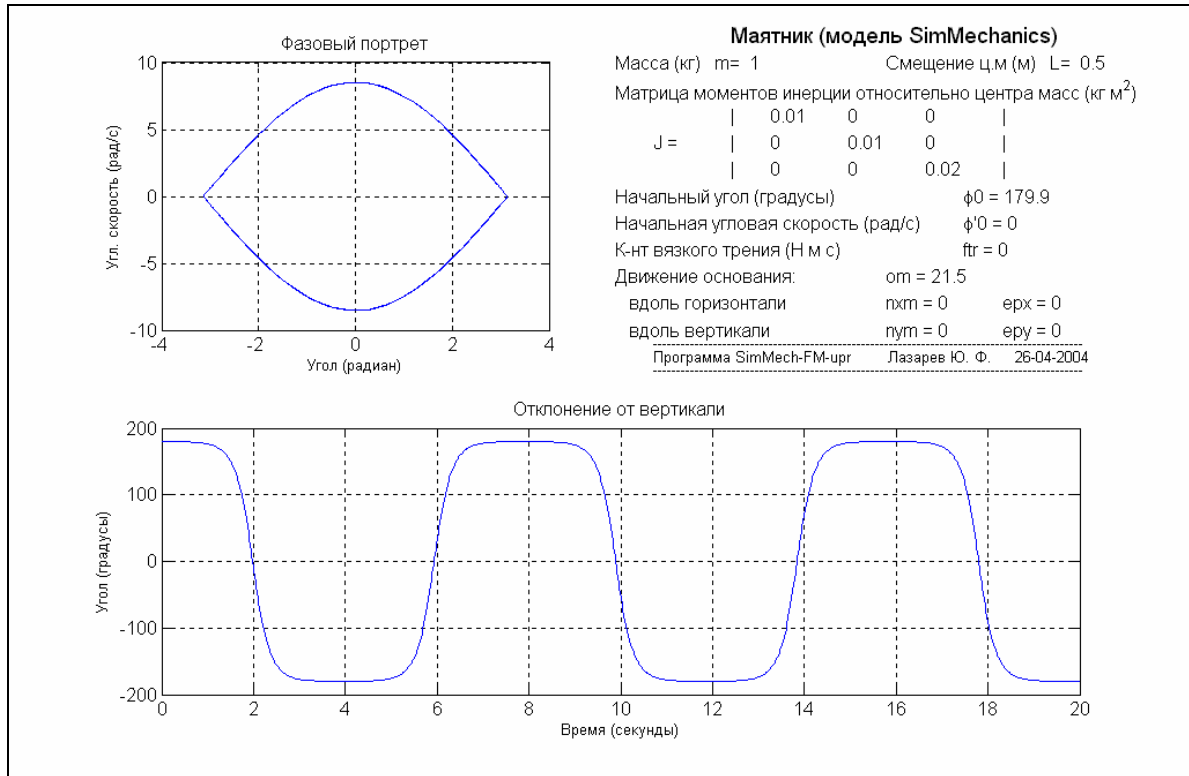


Рис. 12.58. Вільні коливання маятника з великою амплитудою

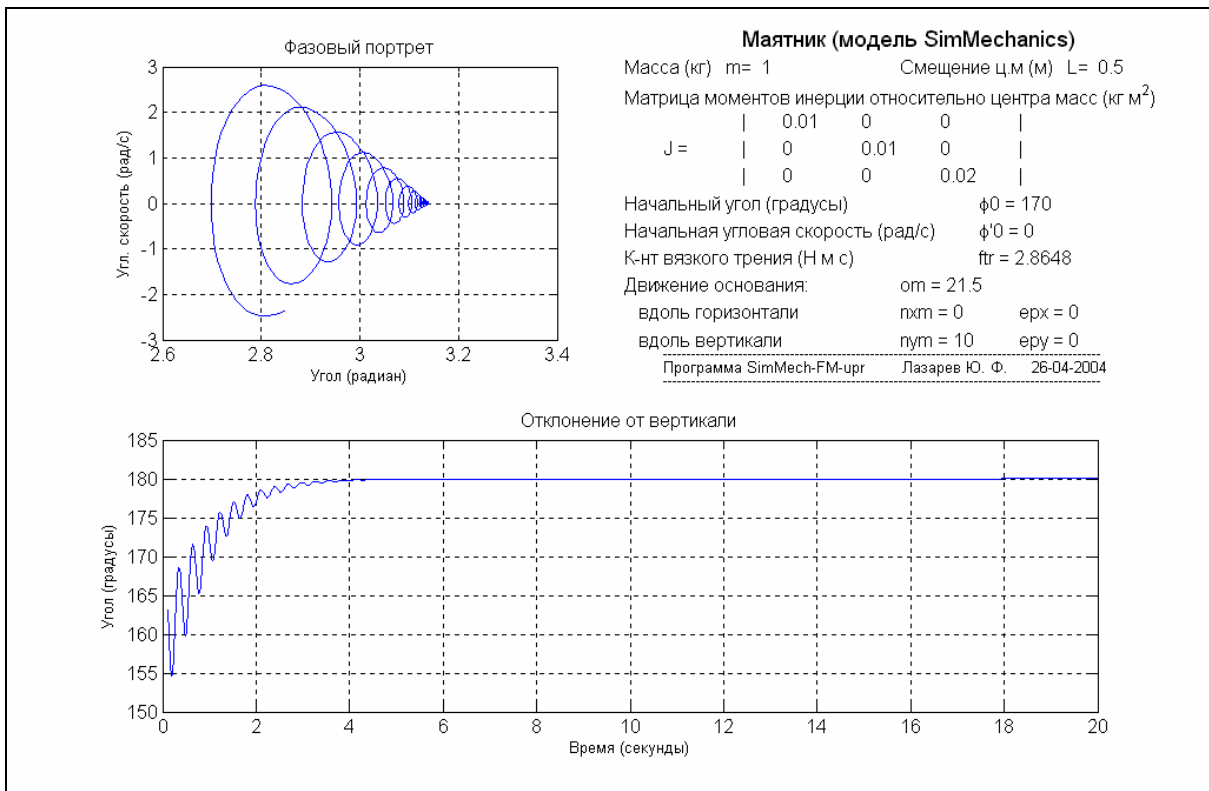


Рис. 12.59. Стійкість верхнього положення рівноваги маятника

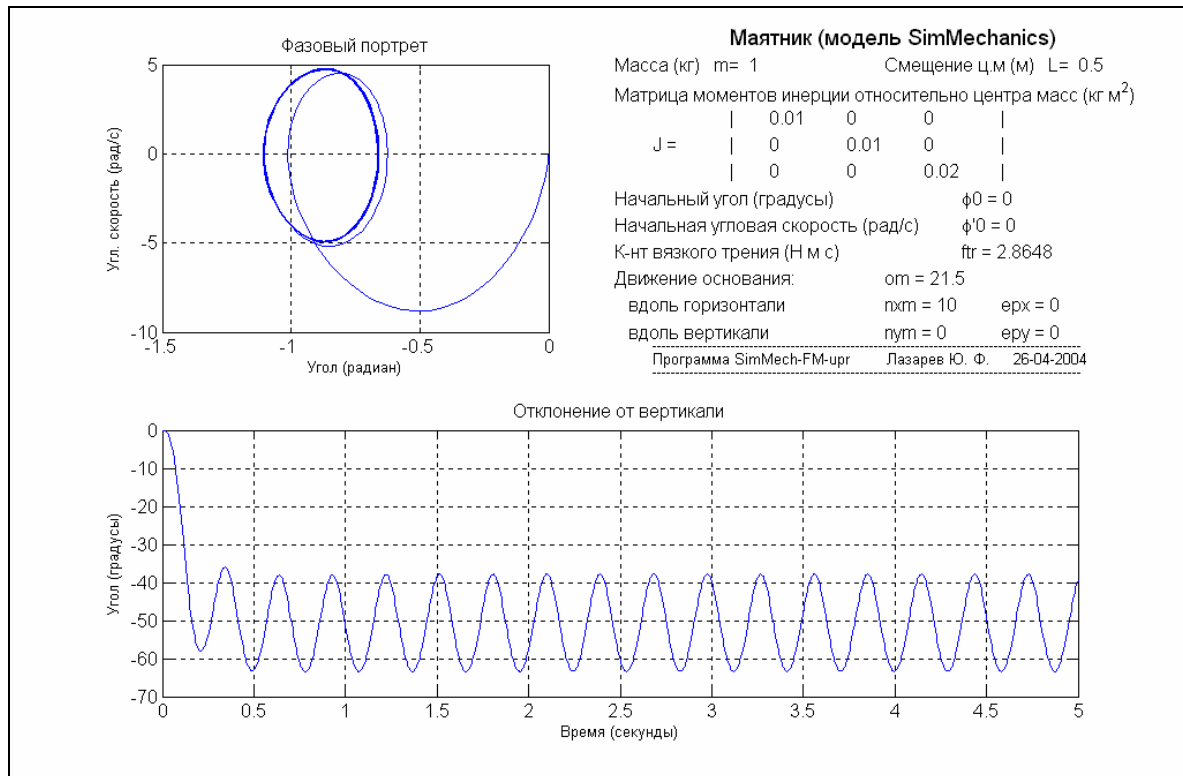


Рис. 12.60. Відхилення положення рівноваги маятника від вертикаліза горизонтальної вібрації основи

12.5. Запитання для самоперевірки

1. Для чого призначено бібліотеку SimMechanics?
2. Які основні принципи формування блок-схеми, що створюється на основі бібліотеки SimMechanics? Чи відмінні вони від принципів утворення S-моделей за допомогою бібліотеки Simulink?
3. З яких основних розділів складається бібліотека SimMechanics?
4. У чому полягають головні особливості блоків бібліотеки SimMechanics у порівнянні зі звичайними S-блоками? Які переваги та недоліки такої побудови блоків?
5. Які блоки бібліотеки SimMechanics здійснюють зв'язок з блоками бібліотеки Simulink і з системою MATLAB?
6. За допомогою яких блоків бібліотеки SimMechanics здійснюється імітація джерел механічного руху? імітація забезпечення ступенів вільності? імітація накладення в'язів?

ЛІТЕРАТУРА

1. Лазарєв Ю. Ф. Моделирование процессов и систем в Matlab. Учебный курс. – СПб.: Питер; Киев: Издат. группа ВНУ, 2005. – 512 с.
2. Лазарєв Ю. Ф. Matlab 5.x. – К.: Издат. группа ВНУ, 2000. – 384 с.
3. Лазарєв Ю. Ф. Моделювання на ЕОМ. Навчальний посібник. – К.: Корнійчук, 2007. – 290 с.
4. Лазарєв Ю. Ф. Математичні моделі та методи теоретичного дослідження стаціонарних лінійних динамічних систем. Конспект лекцій. – К.: КПІ, 1991. – 156 с.
5. Лазарєв Ю. Ф. Методи теоретичного дослідження нелінійних і нестаціонарних динамічних систем. Конспект лекцій. – К.: КПІ, 1991. – 180 с.
6. Герман-Галкин С. Г. Компьютерное моделирование полупроводниковых систем в Matlab 6.0 / Учебн. пособие. – СПб.: Корона-принт, 2006.

АБЕТКОВИЙ ПОКАЖЧИК

З	С
<i>3 × 3 Cross Product</i> 340	<i>cart2pol</i> 77
<i>3-Phase Dynamic Load</i> 366	<i>cart2sph</i> 77
<i>3-Phase Mutual Inductance</i> 366	case 143
<i>3-Phase Parallel RLC Branch</i> 366	<i>ceil</i> 77
<i>3-Phase Parallel RLC Load</i> 366	<i>cell</i> 316, 317
<i>3-Phase Programmable Voltage Source</i> 362, 365, 366	<i>char</i> 316, 317
<i>3-Phase Series RLC Branch</i> 366	Chirp Signal 197, 205
<i>3-Phase Series RLC Load</i> 366	<i>chol</i> 130
<i>3-Phase Source</i> 362, 365	Clock 192, 197, 198
	COESA Atmosphere Model 335
6	<i>colon</i> 319
6DoF (Euler Angles) 330	Combinatorial Logic 225
6DoF (Quaternion) 330	<i>comet</i> 97
	cond 129
A	<i>conj</i> 80
<i>abs</i> 77, 79	Connection Port 396
Abs 223, 262	Connections 218, 238
<i>AC Current Source</i> 362	Constant 193, 198
<i>AC Voltage Source</i> 362	Continuous 185, 209
<i>acos</i> 76, 223	Continuous Angle 397
<i>acosh</i> 76	<i>Controlled Current Source</i> 362
<i>acot</i> 77	<i>Controlled Voltage Source</i> 362
<i>acoth</i> 77	conv 89
<i>acsc</i> 77	corrcoeff 121
<i>acsch</i> 77	cos 76, 223
angle 80	<i>cosh</i> 76, 223
<i>Angle Driver</i> 395	cot 77
<i>ans</i> 74, 79, 195	coth 77
array 317	Coulomb & Viscous Friction 218
<i>asec</i> 77	<i>cov</i> 120
<i>asech</i> 77	cplxpair 80
<i>asin</i> 76, 223	cross 87, 291
<i>asinh</i> 76	csc 77
<i>Asynchronous Machine pu Units</i> 373	<i>csch</i> 77
<i>Asynchronous Machine SI Units</i> 373	<i>ctranspose</i> 319
atan 76, 223	cumprod 118, 119
atan2 76, 223	cumsum 118, 119
atanh 76	Current Measurement 376
axis 99, 161	Custom Joint 387
	Cylindrical 387
	Constant 196
B	D
BackLash 217, 262	Data Store Memory 219
Band-Limited White Noise 197, 207	Data Store Read 219
bar 95	Data Store Write 219
<i>besseli</i> 78	<i>DC Machine</i> 373
<i>besselj</i> 78	<i>DC Voltage Source</i> 362
<i>besselk</i> 78	Dead Zone 216, 262
<i>bessely</i> 78	deconv 89
beta 78	Demux 218
betainc 78	Derivative 209
betaln 78	det 130
Body 379	<i>Detailed Thyristor</i> 371
<i>Body Actuator</i> 391	diag 113
<i>Body Sensor</i> 391	diff 118, 119
break 145	Digital clock 197
Breaker 370	<i>Diode</i> 371
Bushing 389	Direction Cosine Matrix to Euler Angles 341

<i>Direction Cosine Matrix to Quaternions</i>	341
<i>Discontinuities</i>	185
<i>Discrete</i>	185, 213
<i>Discrete DC Machine</i>	373
<i>Discrete Derivative</i>	214
<i>Discrete Filte</i>	214
<i>Discrete State-Space</i>	214
<i>Discrete Transfer Fcn</i>	214
<i>Discrete Zero-Pole</i>	214
<i>Discrete-Time Integrator</i>	214, 215
<i>Diskret Pulse Generator</i>	200
<i>disp</i>	79, 145, 150, 152
<i>display</i>	319
<i>Display</i>	187, 193, 198
<i>Distance Driver</i>	395
<i>Distributed Parameters Line</i>	369
<i>double</i>	316, 317
<i>Driver Actuator</i>	391
<i>Driver&Constraint Sensor</i>	391

E

<i>eig</i> 132	
<i>ellipj</i>	78
<i>ellipke</i>	78
<i>else</i> 142	
<i>elseif</i>	143
<i>Enable</i>	227
<i>end</i> 115, 142, 145	
<i>eq</i> 319	
<i>Equations of Motion (Body Axes)</i>	332
<i>erf</i> 78	
<i>erfc</i>	78
<i>erfcx</i>	78
<i>erfinv</i>	78
<i>Euler Angles to Direction Cosine Matrix</i>	341
<i>Euler Angles to Quaternions</i>	341
<i>Excitation system</i>	376
<i>exp</i> 77, 87, 223	
<i>expint</i>	78
<i>expm</i>	127
<i>expm1</i>	127
<i>expm2</i>	127
<i>expm3</i>	127
<i>eye</i> 111	

F

<i>Fcn</i>	225
<i>feval</i>	166, 170
<i>fft</i> 105, 107	
<i>fftshift</i>	108
<i>figure</i>	101
<i>filter</i>	104
<i>find</i>	317
<i>First-Order Hold</i>	215
<i>fix</i> 77	
<i>fliplr</i>	112
<i>flipud</i>	112
<i>floor</i>	77
<i>fmin</i>	165
<i>fmins</i>	165
<i>for</i> 142, 145	
<i>fplot</i>	165
<i>From</i>	219
<i>From File</i>	197
<i>From Workspace</i>	197, 266, 323
<i>function</i>	141, 146

<i>fzero</i>	165
--------------------	-----

G

<i>gamma</i>	78
<i>gammainc</i>	78
<i>gammainl</i>	78
<i>gcd</i> 78	
<i>ge</i> 319	
<i>Gear Constraint</i>	396
<i>Generic Power System Stabilizer</i>	376
<i>get_param</i>	279, 327
<i>Gimbal</i>	386
<i>global</i>	150, 169
<i>Goto</i>	219
<i>Goto Tag Visibility</i>	219
<i>grid</i>	93
<i>Ground</i>	379, 381
<i>gt</i> 319	
<i>gtext</i>	101
<i>Gto371</i>	

H

<i>hadamard</i>	111
<i>hankel</i>	113
<i>help</i>	77, 82, 140, 149
<i>hess</i>	133
<i>hilb</i>	111
<i>hist95</i>	
<i>Hit Crossing</i>	217, 262
<i>hold off</i>	101
<i>hold on</i>	
<i>horzcat</i>	319
<i>HTG</i>	376
<i>hypot</i>	223

I

<i>IC</i> 264	
<i>Ideal Switch</i>	371
<i>if</i> 142	
<i>ifft</i> 105, 106	
<i>IGBT</i>	372
<i>imag</i>	80, 317
<i>In</i> 238, 267	
<i>Incidence & Airspeed</i>	332
<i>inf</i> 196	
<i>inline</i>	317
<i>In-plane</i>	386
<i>input</i>	150, 152
<i>Integrator</i>	209, 215, 262
<i>interp1</i>	103
<i>inv</i> 123, 131	
<i>invhilb</i>	112
<i>ISA Atmosphere Model</i>	335

J

<i>Joint Actuator</i>	391
<i>Joint Initial Condition</i>	394
<i>Joint Sensor</i>	391
<i>Joint Stiction Actuator</i>	394

K

<i>keyboard</i>	150, 152
-----------------------	----------

L

<i>lcm</i>	78
<i>ldivide</i>	318
<i>le</i>	319
<i>legendre</i>	78
<i>length</i>	317
<i>Linear</i>	209, 218
<i>Linear Driver</i>	395
<i>Linear Transformer</i>	370
<i>log</i>	77, 223
<i>log10</i>	77, 223
<i>log2</i>	78
<i>Logic & Bit Operations</i>	224
<i>Logic & Bit Operations</i>	185
<i>Logical Operator</i>	224
<i>loglog</i>	97, 98, 99
<i>logm</i>	128
<i>logspace</i>	97
<i>lower</i>	317
<i>lt</i>	319
<i>lti</i>	317
<i>ltiview</i>	321
<i>lu</i>	130

M

<i>Machine Measurement Demux</i>	373
<i>Manual Switch</i>	219
<i>Math Function</i>	223
<i>Math Operations</i>	185
<i>Math Operations</i>	220
<i>Math Operations</i>	220
<i>MATLAB Fcn</i>	225
<i>MATLAB Function</i>	292
<i>max</i>	117, 119
<i>mdlDerivatives</i>	325
<i>mdlGetTimeOfNextVarHit</i>	325
<i>mdlInitializeSizes</i>	325
<i>mdlOutputs</i>	325
<i>mdlTerminate</i>	325
<i>mdlUpdate</i>	325
<i>mean</i>	117, 119
<i>Memory</i>	215
<i>menu</i>	152, 154,
<i>Merge</i>	219
<i>min</i>	117, 119
<i>MinMax</i>	223, 262
<i>minus</i>	318
<i>mldivide</i>	318
<i>mod</i>	223
<i>Mosfet</i>	372
<i>mpower</i>	318
<i>mrdivide</i>	318
<i>mtimes</i>	318
<i>Multi-band Power System Stabilizer</i>	376
<i>Multimeter</i>	376
<i>Multiport Switch</i>	219
<i>Mutual Inductance</i>	366
<i>Mux</i>	218

N

<i>ndims</i>	317
<i>norm</i>	129
<i>null</i>	130
<i>num2str</i>	151

O

<i>ode23</i>	164, 171
<i>ode45</i>	164, 171, 173
<i>odeset</i>	164
<i>ones</i>	111
<i>orth</i>	130
<i>otherwise</i>	144
<i>Out</i>	238, 268

P

<i>Parallel Constraint</i>	396
<i>Parallel RLC Branch</i>	366
<i>Parallel RLC Load</i>	366
<i>pascal</i>	
<i>pause</i>	150, 152
<i>Permanent Magnet Synchronous Machine</i>	372
<i>permute</i>	317
<i>PI Section Line</i>	368
<i>pinv</i>	131
<i>Planar</i>	387
<i>plot</i>	91, 99, 158, 166
<i>plus</i>	318
<i>Point Curve Constraint</i>	396
<i>pol2cart</i>	77
<i>poly</i>	89, 132
<i>polyder</i>	91
<i>polyeig</i>	136
<i>polyfit</i>	101
<i>polynom</i>	319
<i>polyval</i>	90, 98
<i>polyvalm</i>	136
<i>Ports & Subsystems</i>	185
<i>pow</i>	223
<i>pow2</i>	78
<i>power</i>	319
<i>Prismatic</i>	380, 384
<i>prod</i>	118, 119
<i>Product</i>	222
<i>Pulse Generator</i>	196

Q

<i>qr</i>	131
<i>quad</i>	163
<i>quad8</i>	163
<i>Quantizer</i>	217
<i>Quaternions to Direction Cosine Matrix</i>	341
<i>Quaternions to Euler Angles</i>	340
<i>qz</i>	134

R

<i>Ramp</i>	196, 201
<i>rand</i>	111, 114
<i>randn</i>	97, 111
<i>Random Number</i>	197, 206
<i>rank</i>	129
<i>rat</i>	78
<i>rats</i>	78
<i>rcond</i>	
<i>rdivide</i>	318
<i>real</i>	80, 317
<i>reciprocal</i>	223
<i>Relational Operator</i>	224, 262
<i>Relay</i>	217
<i>rem</i>	77, 223

<i>Repeating Sequence</i>	197, 204
<i>reshape</i>	113, 317
<i>Revolute</i>	380, 385
<i>roots</i>	89
<i>rot90</i>	112
<i>round</i>	77
<i>Rounding Function</i>	223
<i>rref130</i>	
<i>rsf2csf</i>	134

S

<i>Saturation</i>	216, 262
<i>schur</i>	
<i>Scope</i>	187, 188
<i>Screw</i>	390
<i>sec</i> 76	
<i>sech</i>	77
<i>Second Order Linear Actuator</i>	338
<i>Second Order Nonlinear Actuator</i>	338
<i>semilogx</i>	97, 98, 99
<i>semilogy</i>	97, 99, 158
<i>Series RLC Load</i>	366
<i>set_param</i>	279, 327
<i>SfinTMPL.m</i>	270
<i>S-function</i>	272, 289
<i>SfunTMPL</i>	272
<i>SfunTMPL.m</i>	325
<i>sign</i>	77
<i>Sign</i>	223, 262
<i>Signal Generator</i>	196, 198
<i>Signal Routing</i>	185
<i>Signal Routing</i>	218
<i>Signals & Systems</i>	292
<i>sim</i> 324	
<i>Simplified Synchronous Machine pu Units</i>	372
<i>Simplified Synchronous Machine SI Units</i>	372
<i>simset</i>	269, 324
<i>sin</i> 76, 87, 223	
<i>Sine Wave</i>	192, 197, 202
<i>sinh</i>	76, 223
<i>Sinks</i>	185, 187
<i>Six-DoF</i>	390
<i>size</i> 117, 119, 317	
<i>Slider Gain</i>	222
<i>sort</i>	118, 119
<i>Sources</i>	185, 196
<i>sparse</i>	316
<i>sph2cart</i>	77
<i>Spherical</i>	380, 386
<i>spline</i>	103
<i>sprintf</i>	150, 151
<i>sqrt</i>	77, 79, 223
<i>sqrtm</i>	128
<i>square</i>	223
<i>ss</i> 317	
<i>State-Space</i>	209
<i>std</i> 117, 119	
<i>stem</i>	95
<i>Step</i>	197, 203, 263
<i>STG</i>	376
<i>Stop Simulation</i>	188
<i>strcmp</i>	317
<i>struct</i>	316, 317, 318
<i>subindex</i>	317
<i>subplot</i>	100, 158, 159
<i>subsasgn</i>	319
<i>subsindex</i>	319

<i>subspace</i>	136
<i>subsref</i>	319
<i>Subsystem</i>	227, 238, 288
<i>SubSystem</i>	292
<i>sum</i>	118, 119
<i>Sum</i>	221, 222
<i>Surge Arrester</i>	366
<i>svd</i> 133	
<i>switch</i>	142, 143
<i>Switch</i>	219
<i>sym</i>	317
<i>Synchronous Machine pu Fundamental</i>	373
<i>Synchronous Machine pu Standart</i>	373
<i>Synchronous Machine SI Fundamental</i>	373
<i>S-функція</i>	325

T

<i>tan</i> 76, 87, 223	
<i>tanh</i>	76, 223
<i>text</i> 101, 161	
<i>tf</i> 317	
<i>Three Level Bridge</i>	372
<i>Thyristor</i>	371
<i>times</i>	318
<i>title93</i>	
<i>To File</i>	187, 194
<i>To Workspace</i>	188, 195, 266, 323
<i>trace</i>	130
<i>Transfer Fcn</i>	209, 212, 213
<i>Transport Delay</i>	213
<i>transpose</i>	317, 319
<i>trapz</i>	118
<i>Trigger</i>	227
<i>Trigonometric Function</i>	223
<i>tril</i> 113	
<i>triu</i> 113	
<i>Turbofan Engine System</i>	336

U

<i>uint8</i>	316, 317
<i>uminus</i>	318
<i>Uniform Random Number</i>	197, 206
<i>Unit Delay</i>	215,
<i>Universal</i>	386
<i>Universal Bridge</i>	372
<i>uplus</i>	318
<i>User-defined Functions</i>	185, 225

V

<i>Variable Transport Delay</i>	213
<i>Velocity Driver</i>	395
<i>vertcat</i>	319
<i>Voltage Measurement</i>	364, 376

W

<i>Weld</i>	388
<i>while</i>	142, 144

X

<i>xlabel</i>	93
<i>xor</i> 143	
<i>XYGraph</i>	187, 192

	У	Моделювання математичне.....	9
		Моделювання фізичне.....	8
<i>ylabel</i>		Модель.....	7
		Модель аналогова.....	8
	Z	Модель знакова.....	9
<i>Zero-Order Hold</i>		Модель математична.....	9, 27
<i>Zero-Pole</i>		Модель математична чисельна.....	9
<i>zeros</i>		Модель програмна.....	9
<i>zpk</i> 317		Модель теоретична.....	9
		Модель фізична.....	8
	A		
Алгоритм методу.....			
Аналіз розмірностей.....			
	B		
Блок-схема алгоритму обчислень.....			
	B		
Величини екстенсивні.....			
Величини інтенсивні.....			
Величини кількісні.....			
Величини якісні.....			
	Д		
Дисипація неповна.....			
Дисипація повна.....			
Додавання векторне.....			
Додавання фізичне.....			
	E		
Енергія потенціальна.....			
	З		
Змінні системи фазові.....			
Змінні стану системи.....			
	I		
Інваріантність системи диференціальних рівнянь.....			
	K		
Координати узагальнені.....			
Критерії подібності визначальні.....			
Критерії подібності визначувані.....			
Критерій подібності.....			
	Л		
ЛСС.....			
	M		
Мантиса числа.....			
Методи обчислювальні наближені.....			
Методи обчислювальні точні.....			
Методи Рунге-Кутта.....			
Моделювання.....			
Моделювання аналогове.....			
	H		
		Нелінійність гладка.....	50
		Нелінійність суттєва.....	50
		Нормалізований вигляд числа.....	41
		<i>нормальною формою Коші</i>	55
	П		
		Параметри процесу.....	12
		Параметри середовища.....	13
		Параметри системи.....	13
		Параметри базові.....	15
		Подібність межових умов.....	14
		Подібність початкових умов.....	14
		Подібність фізична.....	14
		Порядок диференціального рівняння.....	48
		Порядок числа.....	41
		Похибка абсолютна гранична.....	34
		Похибка відносна.....	35
		Похибка відносна гранична.....	35
		Похибка результату обчислень.....	39
		Похибкам абсолютна.....	34
		Похибки математичної моделі.....	33
		<i>Похибки методу</i>	34
		Похибки неусувні.....	34
		Похибки округлення.....	34
		Похибки первісних даних.....	34
		Похибки поширення.....	34
		Похибки усікання.....	34
		Простір стану.....	56
		Простір фазовий.....	56
	C		
		<i>cross</i>	299
		Сила.....	50
		Сила гіроскопічна.....	53
		Сила дисипативна.....	53
		Сила зовнішня.....	50
		Сила інерції.....	52
		Сила консервативна.....	52
		Сила неконсервативна.....	53
		Сила нелінійна.....	52
		Сила потенціальна.....	52
		Сила прискорювальна.....	53
		Сила псевдогіроскопічна.....	53
		Сила радіальної корекції.....	53
		Сила циркуляційна.....	53
		Система динамічна.....	48
		Система динамічна лінійна.....	50, 51
		Система динамічна лінійна стаціонарна.....	51
		Система динамічна нестационарна.....	51
		Система динамічна параметрично збуджувана.....	51
		Система лінійна нестационарна.....	49
		Система лінійна стаціонарна.....	49
		Система нелінійна.....	49

Стан системи 56
Структурна схема алгоритму 30

Т

Точка зображувальна 56
Траекторія фазова 56

У

Умови граничні 13
Умови однозначності 13
Умови початкові 13

Ф

Фаза системи 56
Форма нормальна Коші 55
Функція Релея дисипативна 53
Функція розсіювання енергії 53

Ц

Цифра значуща вірна 35
Цифра значуща сумнівна 35
Цифри числа значущі 35

Ч

Час модельний 21

Ш

Швидкість фазова 56

Я

Явища подібні 14