Список рекомендованих до використання методів numpy до роботи № 1

	numpy.random.rand	масив випадкових значень заданої форми.
	numpy.mat	інтерпретує вхідні дані як матрицю.
	numpy.savetxt	зберігає масив у текстовий файл
	numpy.linalg.det	обчислює визначник (детермінант) матриці
	numpy.matmul	обчислює матричний твір двох масивів
	numpy.linalg.inv	обчислює зворотну матрицю
	numpy.zeros	повертає новий масив зазначеної форми та типу, заповнений нулями
	numpy.linalg.cond	обчислює число обумовленості матриці
	numpy.linalg.qr	виконує QR-розкладання матриці
	numpy.linalg.eig	обчислює власні числа (значення) та власні вектори квадратної матриці
	numpy.linalg.pinv	обчислює псевдозворотну матрицю
	numpy.linalg.LinAlgError	генерує винятки Python, викликані функціями модуля linalg
	numpy.linalg.solve	вирішує лінійне матричне рівняння (систему лінійних рівнянь)

numpy.random.rand

`numpy.random.rand(d0, d1, ..., dn)`

Масив випадкових значень заданої форми.

Ця функція створює масив зазначеної форми та заповнює його випадковими числами з плаваючою точкою, які рівномірно розподілені в інтервалі [0, 1).

Параметри:

d0, d1, …, dn – цілі позитивні числа, необов'язковий параметр.
Якщо цей параметр не вказано, буде повернено одне випадкове число. Якщо вказано одне число d, то буде повернено одновимірний масив з кількістю елементів, що дорівнює d. Якщо зазначено кілька чисел, кількість цих чисел визначатиме кількість осей масиву, які значення - кількість елементів вздовж кожної їх.

Повертає:

результат - масив NumPy або число
Масив із формою (d0, d1, …, dn) заповнений рівномірно розподіленими випадковими числами з інтервалу [0, 1). Якщо форму масиву не вказано, то повертається одне випадкове число.

Зауваження

Це, мабуть, найзручніша функція для швидкого (у плані стукання по клавіатурі) отримання випадкових даних, оскільки вона вимагає введення форми масиву як кортежу. Однак, якщо потрібно керувати формою масивів за допомогою кортежів, зверніться до функції[numpy.random.random_sample](#).

Дивіться також:
[random](#)

Приклади

```
>>> import numpy as np
>>>
>>> np.random.rand()
0.004859283445131979
>>>
>>> np.random.rand(7)
array([0.58889628, 0.47765745, 0.02935087, 0.330708, 0.80487635,
       0.79638419, 0.87066946])
>>>
>>> np.random.rand(3, 3)
array([[0.92741287, 0.15253227, 0.70188172],
       [0.14482399, 0.61890041, 0.22074184],
       [0.15232495, 0.72143561, 0.86303536]])
```

---

numpy.mat

numpy.mat(object, dtype=None)

Функція mat() інтерпретує вхідні дані як матрицю.

Параметри:

object - подібний до матриці об'єкт
Рядок, список або кортеж, а також будь-яка функція або об'єкт з методом, що повертають список або кортеж, які можуть бути інтерпретовані як матриця.
dtype – тип даних NumPy (необов'язковий)
Визначає тип даних вихідної матриці.

Повертає:

matrix - масив NumPy
Інтерпретація вхідних даних як матриці.

Дивіться також: bmat

Зауваження

Вхідні дані можуть бути інтерпретовані як матриці, тільки якщо з них можливо побудувати одномірний або двовимірний прямокутний (квадратний) масив.

Просто відформатовані рядки, наприклад, такі як '1 1; 2 2', '1, 1; 2, 2' або '1,1;2,2' також можуть інтерпретуватися як матриці.

На відміну від функційmatrixі asmatrix, функція mat не робить копію даних, якщо вхідні дані вже є масивом NumPy або матрицею. Еквівалентна matrix (data, copy = False).

Приклади

```
>>> import numpy as np
>>>
>>> # Створення матриць з рядків
...
>>> np.mat('1 1 1 1')
matrix([[1, 1, 1, 1]])
>>>
>>> np.mat('1, 1, 1, 1')
matrix([[1, 1, 1, 1]])
>>>
>>> np.mat('1, 1, 1, 1')
matrix([[1, 1, 1, 1]])
>>>
>>> np.mat('1, 1; 1, 1')
matrix([[1, 1],
        [1, 1]])
>>>
>>>
>>> a = [1, 2, 3, 4, 5]
>>>
>>> b = np.mat(a)
>>> b
matrix([[1, 2, 3, 4, 5]])
>>>
>>>
>>> # Якщо вхідні дані не є масивом або матрицею,
... # то вхідні дані копіюються.
...
>>> a[0] = 111 # Зміна списку 'a' не відобразиться
>>> a
[111, 2, 3, 4, 5]
>>>
>>> b # не позначиться на матриці 'b'.
matrix([[1, 2, 3, 4, 5]])
>>>
>>>
>>> # Якщо вхідні дані є масивом або матрицею,
... # вхідні дані не копіюються.
...
>>> a = np.array([[1, 1], [2, 2]])
>>> a
array([[1, 1],
       [2, 2]])
>>>
```

```
>>> b = np.mat(a)
>>> b
matrix([[1, 1],
        [2, 2]])
>>>
>>> a[0] = 77 # Зміна масиву 'a' не відобразиться
>>> a
array([[77, 77],
       [2, 2]])
>>>
>>> b # відобразиться на матриці 'b'.
matrix([[77, 77],
        [2, 2]])
```

numpy.savetxt

numpy.savetxt(fname, X, fmt='%.18e', delimiter='', newline='\n', header='', footer='', comments='#', encoding=None)

Функція savetxt() зберігає масив текстового файлу.

Параметри:

fname – рядок, файловий об'єкт (дескриптор).
Ім'я файлу або шлях до файлу, в який буде збережено масив. Якщо вказаного файлу немає, він буде створено. Жодне розширення до файлу не додається, але може бути явно визначено. Якщо вказати розширення .gz, дані будуть збережені в стислому форматі gzip, файли такого формату можуть бути прочитані за допомогою функції`loadtxt()`.
X - 1 або 2-мірний масив NumPy або подібний масиву об'єкт.
Масив, який необхідно зберегти у файл.
fmt - рядок чи послідовність рядків (необов'язковий параметр).

Визначає формат, в якому записуються числові елементи в масив. Якщо вказано один рядок з форматом, наприклад % 10.7i, то цей формат поширюється на всі елементи, що зберігаються. Також можна вказати формат для кожного окремого стовпця у вигляді послідовності рядків-форматів.

Для комплексних чисел допустимі такі способи вказівки форматів:

- один формат для уявної та дійсної частини, наприклад: fmt = '%. 4f', що призведе до форматування чисел у вигляді '(%s+%sj)' % (fmt, fmt);
- рядок визначальний форматування дійсної та уявної частини кожного стовпця, наприклад: '% .4f% +. 4fj% .4f%+. 4fj' для двох стовпців;
- список рядків форматів, але в цьому випадку дійсна та уявна частини повинні мати окремі формати всередині кожного окремого рядка зриску, наприклад: ['% 5.4f +% 5.4ej', '(% 10.10f% +. 10.10ej)'] для двох стовпців.

delimiter – рядок (необов'язковий параметр).
Задає роздільник між елементами (стовпцями) масиву.
newline – рядок (необов'язковий параметр).

Задає роздільник між рядками масивів.
header – рядок (необов'язковий параметр).
Рядок, який буде записано на початку файлу.
footer – рядок (необов'язковий параметр).
Рядок, який буде записано в кінці файлу.
comments – рядок (необов'язковий параметр).
Рядок яка буде додана на початок рядків header та footer. За промовчанням, це '#', який є стандартним символом для рядків коментування та очікується функцією loadtxt().
encoding - None або рядок (необов'язковий параметр).
Визначає кодування, яке використовується під час запису даних у текстовий файл, але не впливає на потоки даних. Якщо кодування відрізняється від 'bytes' або 'latin1', відкрити файли в NumPy версії < 1.14 не вийде. За промовчанням використовується 'latin1'. Доступно у NumPy з версії 1.14.0

Зауваження

Параметр fmt дозволяє задати формат у якому числа будуть записані у файл. Формат задається у вигляді рядка виду %[flag]width[.precision]specifier.

flag

- `-` - Вирівнювання по лівому краю.
- `+` - Запис перед знаками '+' або '-'.
- `0` - Додає зліва перед числом нулі замість пробілу.

width

Мінімальна кількість символів для запису. Якщо число складається з більшої кількості знаків, воно не усікається.

precision

- мінімальна кількість цифр, які будуть використані для запису цілих даних (d,i,o,x);
- кількість цифр після коми для запису дійсних чисел (наприклад, e, E, f);
- максимальна кількість цифр для запису чисел (наприклад, g, G);
- максимальна кількість символів для рядків (s).

specifier

- `c` - Символ;
- `d` або i – десяткове ціле число зі знаком;
- `e` або i - запис у науковій нотації із символами e або E;
- `f` - десяткове число з плаваючою точкою;
- `g, G` - використання укорочених записів із символами e або E для f;
- `o` - вісімкове число зі знаком;
- `s` - рядок символів;
- `u` - десяткове ціле число без знака;
- `x, X` - шістнадцяткове ціле число без знаку.

Дивіться також:
`loadtxt`,`save`,`savez`,`savez_compressed`.
Повний опис рядків форматування(Англ)

Приклади

Зберегти в текстовий файл можна лише одновимірні та двовимірні масиви:

```
>>> import numpy as np
>>>
>>> a = np.arange(10)
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>>
>>> np.savetxt('example/ex_0', a)
```

Вміст файлу ex_0:

```
0.00000000000000000e+00
1.00000000000000000e+00
2.00000000000000000e+00
3.00000000000000000e+00
4.00000000000000000e+00
5.00000000000000000e+00
6.00000000000000000e+00
7.00000000000000000e+00
8.00000000000000000e+00
9.00000000000000000e+00
```

Файл не має розширення, проте він має текстовий формат і відкривається будь-яким текстовим редактором.

Збереження двовимірного масиву:

```
>>> b = a.reshape(5, 2)
>>> b
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>>
>>> np.savetxt('example/ex_1', b)
```

Вміст файлу ex_1:

```
0.00000000000000000e+00 1.00000000000000000ee+00
2.00000000000000000e+00 3.00000000000000000ee+00
4.00000000000000000e+00 5.00000000000000000ee+00
6.00000000000000000e+00 7.00000000000000000ee+00
8.00000000000000000e+00 9.00000000000000000ee+00
```

У записі окремих чисел є дуже багато незначних нулів. Керувати форматом запису чисел можна за допомогою параметра fmt:

```
>>> c = np.random.randint(0, 300, size = (6, 6))
>>> c
array([[ 28, 286, 233, 162, 235, 220],
       [148, 171, 187, 121, 58, 162],
       [24, 176, 142, 202, 58, 275],
       [101, 32, 30, 126, 128, 127],
       [254, 78, 159, 225, 117, 45],
       [125, 289, 109, 8, 284, 10]])
>>>
>>> np.savetxt('example/ex_2', c, fmt = '%3.0d')
```

Вміст файлу ex_2:

```
28 286 233 162 235 220
148 171 187 121 58 162
24 176 142 202 58 275
101 32 30 126 128 127
254 78 159 225 117 45
125 289 109 8 284 10
```

За замовчуванням як роздільник між числами використовується рядок ' ' - один пробіл. Задати довільний роздільник можна за допомогою параметра delimiter:

```
>>> d = 1/np.random.randint(2, 10, size = (6, 5))
>>> d
array([[0.25, 0.125, 0.14285714, 0.125, 0.125],
       [0.5, 0.2, 0.11111111, 0.2, 0.33333333],
       [0.5, 0.25, 0.25, 0.2, 0.11111111],
       [0.33333333, 0.14285714, 0.11111111, 0.25, 0.2],
       [0.16666667, 0.11111111, 0.11111111, 0.2, 0.5],
       [0.11111111, 0.5, 0.14285714, 0.14285714, 0.5]])
>>>
>>> np.savetxt('example/ex_3', d, fmt = '%1.4f', delimiter = ', ')
```

Вміст файлу ex_3:

```
0.2500, 0.1250, 0.1429, 0.1250, 0.1250
0.5000, 0.2000, 0.1111, 0.2000, 0.3333
0.5000, 0.2500, 0.2500, 0.2000, 0.1111
0.3333, 0.1429, 0.1111, 0.2500, 0.2000
0.1667, 0.1111, 0.1111, 0.2000, 0.5000
0.1111, 0.5000, 0.1429, 0.1429, 0.5000
```

За умовчанням як роздільник між рядками масиву використовується символ перенесення рядка '\n'. Задати довільний роздільник можна за допомогою параметра newline:

```
>>> e = 1/np.random.randint(2, 10, size = (6, 5))
>>> e
array([[0.11111111, 0.25, 0.2, 0.14285714, 0.33333333],
       [0.33333333, 0.25, 0.2, 0.11111111, 0.125],
       [0.14285714, 0.11111111, 0.16666667, 0.2, 0.5],
       [0.5, 0.5, 0.5, 0.2, 0.125],
       [0.5, 0.2, 0.5, 0.33333333, 0.25],
       [0.14285714, 0.11111111, 0.14285714, 0.16666667, 0.11111111]])
>>>
>>> np.savetxt('example/ex_4',
               e,
               fmt = '%1.4f',
               delimiter = '| ',
               newline = '\n' + '-'*42 + '\n')
```

Вміст файлу ex_4:

```
0.1111 | 0.2500 | 0.2000 | 0.1429 | 0.3333
------------------------------------------
0.3333 | 0.2500 | 0.2000 | 0.1111 | 0.1250
------------------------------------------
0.1429 | 0.1111 | 0.1667 | 0.2000 | 0.5000
------------------------------------------
0.5000 | 0.5000 | 0.5000 | 0.2000 | 0.1250
------------------------------------------
0.5000 | 0.2000 | 0.5000 | 0.3333 | 0.2500
------------------------------------------
0.1429 | 0.1111 | 0.1429 | 0.1667 | 0.1111
```

Форматування комплексних чисел, можливе трьома різними способами. Найпростіший, один формат для дійсної та уявної частини кожного стовпця:

```
>>> a = np.linspace(1 + 1j, 6 + 6j, 12).reshape(6, 2)
>>> a
array([[1. +1.j , 1.45454545+1.45454545j],
       [1.90909091+1.90909091j, 2.36363636+2.36363636j],
       [2.81818182+2.81818182j, 3.27272727+3.27272727j],
       [3.72727273+3.72727273j, 4.18181818+4.18181818j],
       [4.63636364+4.63636364j, 5.09090909+5.09090909j],
       [5.54545455+5.54545455j, 6. +6.j ]])
>>>
>>> np.savetxt('example/ex_5', a, fmt = '%1.4f')
```

Однак, у цьому випадку, числа у кожному стовпці будуть автоматично поміщені у круглі дужки. Вміст файлу ex_5:

```
(1.0000+1.0000j) (1.4545+1.4545j)
(1.9091+1.9091j) (2.3636+2.3636j)
(2.8182+2.8182j) (3.2727+3.2727j)
(3.7273+3.7273j) (4.1818+4.1818j)
(4.6364+4.6364j) (5.0909+5.0909j)
(5.5455+5.5455j) (6.0000+6.0000j)
```

Можна визначити формат уявної та дійсної частини окремо у кожному стовпці, для цього необхідно вказати рядок з усіма форматами:

```
np.savetxt('example/ex_5', a, fmt = '%1.3f %+0.3ej %1.3f %+0.3ej')
```

Вміст файлу ex_5:

```
1.000 +1.000e+00j 1.455 +1.455e+00j
1.909 +1.909e+00j 2.364 +2.364e+00j
2.818 +2.818e+00j 3.273 +3.273e+00j
3.727 +3.727e+00j 4.182 +4.182e+00j
4.636 +4.636e+00j 5.091 +5.091e+00j
5.545 +5.545e+00j 6.000 +6.000e+00j
```

Визначити формат уявної та дійсної частини кожного стовпця масиву можна за допомогою списку рядків-форматів:

```
>>> np.savetxt('example/ex_5', a, fmt = ['(%0.3e %+0.3ej)',
'%1.3f %+1.3fj']))
```

Вміст файлу ex_5:

```
(1.000e+00 +1.000e+00j) 1.455 +1.455j
(1.909e+00 +1.909e+00j) 2.364 +2.364j
(2.818e+00 +2.818e+00j) 3.273 +3.273j
(3.727e+00 +3.727e+00j) 4.182 +4.182j
(4.636e+00 +4.636e+00j) 5.091 +5.091j
(5.545e+00 +5.545e+00j) 6.000 +6.000j
```

numpy.linalg.det

`numpy.linalg.det(a)`

Функція linalg.det() обчислює визначник (детермінант) матриці.

Параметри:

a - масив NumPy або подібний масиву об'єкт.
Це може бути тільки " квадратний " двомірний масив, тобто. Квадратна матриця.
Якщо це багатовимірний масив, дві його останні осі повинні бути рівні, в цьому випадку він розглядається як масив матриць і детермінант розраховується окремо для кожної з них.

Повертає:

результат - масив NumPy або число
Якщо a це багатовимірний масив, то повертається масив визначників кожної підматриці вихідного масиву. Якщо на вхід подано одну квадратну матрицю, то її визначник повертається у вигляді одного числа.

Дивіться також:

det,slogdet,svd

Приклади

```
>>> import numpy as np
>>> від numpy import linalg as LA
>>>
>>> a = np.random.randint(10, size = (3, 3))
>>> a
array([[4, 9, 9],
       [6, 9, 8],
       [9, 5, 7]])
>>>
>>> LA.det(a)
-97.0
>>>
>>>
>>> b = np.random.randint(10, size = (3, 2, 2))
>>> b
array([[[6, 0],
        [0, 9]],

       [[3, 6],
        [5, 0]],

       [[9, 5],
        [7, 8]]])
>>>
>>> LA.det(b)
array ([54., -30., 37.])
>>>
>>> LA.det(b[0])
54.00000000000001
```

numpy.matmul

numpy.matmul(a, b, out=None)

Функція matmul() обчислює матричний добуток двох масивів. Доступна NumPy з версії 1.10.0.

Поведінка функції залежить від вхідних масивів таким чином:

- Якщо обидва аргументи це двовимірні масиви, всі вони множаться як звичайні матриці;
- Якщо один з масивів має розмірність більше двох, то він вважається послідовністю матриць з формою, що дорівнює останнім двом індексам. Далі спрацьовує механізм транслювання;
- Якщо перший аргумент це одномірний масив (вектор), він вважається матрицею у якій кожен рядок повторюється і дорівнює вихідному вектору. Після множення повертається один рядок результуючої матриці;
- Якщо другий аргумент це одномірний масив, він вважається матрицею у якій кожен стовпець повторюється і дорівнює вихідному вектору. Після множення повертається один стовпець результуючої матриці;
- Добуток послідовності матриць на вектор і навпаки, поверне послідовність векторів;
- Аргументом може бути число, при цьому використовуйте a*b.

Параметри:

a, b - масиви NumPy або подібні до масивів об'єкти.
Вхідні дані
out – масив NumPy, необов'язковий параметр.
Масив, в який можна помістити результат функції. Даний масив повинен відповідати формі та типу даних результуючого масиву функції, а також обов'язково бути C-суміжним, тобто. зберігати дані у рядковому З стилі. Вказівка даного параметра дозволяє уникнути зайвої операції присвоювання тим самим трохи прискорюючи роботу вашого коду. Корисний параметр якщо ви часто звертаєтеся до функції в циклі.

Повертає:

результат - масив NumPy або число
Якщо обидва аргументи це вектори, то повертається число. Інакше повертається масив.

Зауваження

Якщо один із аргументів це число, або якщо довжина останньої осі a не дорівнює довжині всіх осей крім першої масиву b, виникає виняток ValueError.

Дивіться також:

dot,vdot,tensordot,einsum

Приклади

Розмноження двовимірних масивів:

```
>>> a = [[1, 2], [3, 4]]
>>> b = [[2, 4], [1, 3]]
>>>
>>> np.matmul(a, b)
array([[ 4, 10],
```

```
       [10, 24]])
```

Розмноження одновимірного масиву на двовимірний:

```
>>> c = [5, 7]
>>>
>>> np.matmul(c, a)
array([26, 38])
>>>
>>> np.matmul(a, c)
array([19, 43])
```

Масиви з розмірністю більше двох вважаються послідовністю матриць:

```
>>> d = np.arange(12).reshape(3, 2, 2)
>>> d
array([[[ 0, 1],
        [2, 3]],

       [[ 4, 5],
        [6, 7]],

       [[ 8, 9],
        [10, 11]]])
>>>
>>> np.matmul(a, d)
array([[[ 4, 7],
        [8, 15]],

       [[16, 19],
        [36, 43]],

       [[28, 31],
        [64, 71]]])
>>>
>>> np.matmul(a, d[0])
array([[ 4, 7],
       [8, 15]])
```

Для комплексних векторів повертається внутрішній твір:

```
>>> np.matmul([1+2j, 1+3j], [1+2j, 1+3j])
(-11+10j)
>>>
>>> np.inner([1+2j, 1+3j], [1+2j, 1+3j])
(-11+10j)
```

numpy.linalg.inv

numpy.linalg.inv(a)

Функція linalg.inv() обчислює зворотну матрицю.

Для квадратної матриці A

, зворотною є така матриця A-1

, Для якої виконується умова:

$$AA^{-1}=A^{-1}A=E$$

Якщо вхідна матриця a не є квадратною або обчислення зворотної матриці неможливо, то викликається виключення LinAlgError.

Параметри:

a - масив NumPy або подібний масиву об'єкт.
Це може бути тільки " квадратний " двомірний масив, тобто. Квадратна матриця.
Якщо це багатовимірний масив, дві його останні осі повинні бути рівні, в цьому випадку він розглядається як масив матриць і зворотна матриця обчислюється окремо для кожної з них.

Повертає:

результат - масив NumPy
Зворотна матриця чи масив зворотних матриць.

Дивіться також:

linalg.pinv,linalg.tensorinv

Приклади

```
>>> import numpy as np
>>> від numpy import linalg as LA
>>>
>>> a = np.random.randint(10, size = (6, 6))
>>> a
array([[8, 9, 7, 6, 3, 0],
       [6, 3, 1, 9, 8, 1],
       [9, 5, 9, 7, 3, 6],
       [9, 1, 3, 2, 4, 9],
       [1, 5, 8, 4, 0, 0],
       [9, 2, 0, 4, 7, 5]])
>>>
>>> a_inv = LA.inv(a)
>>> a_inv
array([[-0.14883721, -0.39186047, 0.53837209, -0.69069767, -0.16744186,
0.6755814 ],
       [ 0.53333333, 0.55 , -0.78333333, 1.1 , -0.06666667, -1.15 ],
       [-0.41627907, -0.51395349, 0.4627907, -0.83023256, 0.34418605,
1.04186047],
       [0.20310078, 0.43837209, -0.08100775, 0.45813953, -0.31317829, -
0.81511628],
       [-0.30465116, -0.28255814, -0.00348837, -0.4372093, 0.46976744,
0.84767442],
       [0.31860465, 0.53023256, -0.58604651, 1.04883721, -0.07906977, -
1.09069767]])
>>>
>>>
>>> np.dot(a, a_inv)
array([[ 1.00000000e+00, -1.11022302e-16, -9.85322934e-16, -4.44089210e-16,
2.22044605e-16, 4.44089210
       [-5.55111512e-17, 1.00000000e+00, 3.33066907e-16, -1.77635684e-15,
0.00000000e+00, 1.99840144e-15],
       [ 0.00000000e+00, 0.00000000e+00, 1.00000000e+00, -8.88178420e-16,
0.00000000e+00, 0.00000000ee+00],
       [0,00000000e+00, -8.88178420e-16, 8.88178420e-16, 1.00000000e+00,
0.00000000e+00, 1.77635684e-15],
       [-3.33066907e-16, 6.66133815e-16, -9.99200722e-16, 1.33226763e-15,
1.00000000e+00, -2.22044605e-15],
```

```
       [2.22044605e-16, -4.44089210e-16, -4.44089210e-16, -1.77635684e-15,
4.44089210e-16, 1.00000000ee+00]
```

Ми можемо перевірити правильність результату, спираючись основне властивість зворотної матриці, тобто. np.dot(a, ainv) = np.dot(ainv, a) = np.eye(a.shape[0])

```
>>> np.allclose(np.dot(a, a_inv), np.eye(a.shape[0]))
True
>>>
>>> np.allclose(np.dot(a_inv, a), np.eye(a.shape[0]))
True
```

Поняття зворотної матриці може бути узагальнено і для матриць над полем комплексних чисел:

```
>>> b = np.random.randint(0, 10, size = (4, 4))
>>> c = np.random.randint(0, 10, size = (4, 4))
>>>
>>> z = b + c * 1j
>>> z
array([[7.+0.j, 7.+4.j, 6.+1.j, 9.+9.j],
       [2.+6.j, 4.+8.j, 5.+8.j, 8.+4.j],
       [7.+1.j, 7.+4.j, 1.+6.j, 5.+5.j],
       [9.+1.j, 8.+6.j, 4.+4.j, 6.+8.j]])
>>>
>>> z_inv = LA.inv(z)
>>> z_inv
array([[ 0.04248703-0.30126338j, -0.13290916+0.01529899j, 0.07682157-
0.35652787j, 0.08277916+0.57908
       [-0.04071282+0.36279118j, 0.07957304-0.10205249j,
0.08653401+0.46482539j, -0.14389804-0.66769836j],
       [-0.01060869+0.0444729j , 0.04056649-0.01735018j, -0.1520231 -
0.11146967j, 0.09741184+0.03498778j],
       [0.11237768-0.17060791j, 0.0031617 +0.03600685j, 0.06850577-
0.09999216j, -0.10577338+0.20449824j]])
>>>
>>> np.allclose(np.dot(z, z_inv), np.eye(z.shape[0]))
True
>>>
>>> np.allclose(np.dot(z_inv, z), np.eye(z.shape[0]))
True
```

numpy.zeros

`numpy.zeros(shape, dtype=float, order='C')`

Функція zeros() повертає новий масив зазначеної форми та типу, заповнений нулями.

Параметри:

shape - ціле число, список чи кортеж цілих чисел
Задає розміри необхідного масиву – ціле число чи кортеж цілих чисел.
dtype – тип даних NumPy (необов'язковий)
Визначає тип даних вихідного масиву.
order - 'C' або 'F' (необов'язковий)

Цей параметр визначає, у якому порядку масиви повинні зберігатися в пам'яті: рядковому C-стилі або стовпчастому стилі Fortran.

Повертає:

результат - масив NumPy

Масив з нулів, зазначеної форми, типу та порядку.

Дивіться також: zeros like,ones,empty,full

Приклади

```
>>> import numpy as np
>>>
>>> np.zeros(4) # Масив-рядок з нулів
array([0., 0., 0., 0.])
>>>
>>> np.zeros((4, 1)) # Масив-стовпець з нулів
array([[ 0.],
       [0.],
       [0.],
       [0.]]))
>>>
>>> np.zeros((3, 3)) # Масив з нулів зазначених розмірів
array([[ 0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
>>>
>>> np.zeros((3, 3), dtype = np.int8) # Вказуємо форму та тип даних
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]], dtype = int8)
>>>
>>>
>>> # zeros може застосовуватися для створення та
... # заповнення структурованих масивів:
... f = np.zeros((5), dtype=[('x', 'i2'), ('y', 'f4')])
>>>
>>> f
array([(0, 0.0), (0, 0.0), (0, 0.0), (0, 0.0), (0, 0.0)],
      dtype=[('x', '<i2'), ('y', '<f4')])
>>>
>>> f['x']
array([0, 0, 0, 0, 0], dtype=int16)
>>>
>>> f['y']
array([ 0., 0., 0., 0., 0.], dtype = float32)
```

numpy.linalg.cond

numpy.linalg.cond(x, p=None)

Функція linalg.cond() обчислює кількість обумовленості матриці.

Дане число дозволяє оцінити, наскільки матриця далека від матриці повного рангу або від невиродженості для квадратної матриці. Число обумовленості може бути обчислено на підставі однієї із семи норм, яку можна вказати в параметрі p.

Параметри:

x - масив NumPy або подібний до масиву об'єкт.
Це може бути як двовимірний так багатовимірний масив. У разі багатовимірних масивів, він розглядається як масив матриць щодо двох останніх осей, і число обумовленості обчислюється для кожної такої підматриці окремо.
p - {None, 1, -1, 2, -2, np.inf, -np.inf, 'fro'} (необов'язковий параметр).
Дозволяє вказати необхідну при обчисленні норму матриці:
None
Встановлений за умовчанням і обчислює 2 норму, тобто. найбільше сингулярне число матриці;
'fro'
Норма Фробеніуса;
np.inf
np.max(np.sum(np.abs(x), axis=1));
-np.inf
np.min(np.sum(np.abs(x), axis=1));
1
np.max(np.sum(np.abs(x), axis=0));
-1
np.min(np.sum(np.abs(x), axis=0));
2
Найбільше сингулярне число матриці;
-2
Найменше сингулярне число матриці;

Повертає:

результат - дійсне число або inf
Число обумовленості матриці, яке може дорівнювати inf.

Зауваження

Чим ближче число обумовленості до 1, тим ближче матриця до невироджених або повного рангу.

Дивіться також:
norm

Приклади

```
>>> import numpy as np
>>> від numpy import linalg as LA
>>>
>>> a = [[1, 10], [100, 1001]]
>>>
>>> LA.cond(a)
1012102.000015063
>>>
>>> b = [[10, 3], [2, 11]]
>>>
>>> LA.cond(b)
1.6403882032022075
>>>
>>> c = [[1, 0], [0, 1]]
>>>
>>> LA.cond(c)
1.0
```

numpy.linalg.qr

`numpy.linalg.qr(a, mode = 'reduced')`

Функція linalg.qr() виконує QR-розкладання матриці.

Дане розкладання дозволяє уявити матрицю A як твори QR, де Q - це ортогональна (унітарна) матриця, а R - це верхньотрикутна матриця.

Параметри:

a - масив NumPy або подібний до масиву об'єкт.
Двовимірний масив з формою (N, M), який може складатися як з комплексних, так і речових чисел.
mode - {'reduced', 'complete', 'r', 'raw', 'full', 'economic'} (необов'язковий параметр).

Дозволяє задати розмір матриць Q і R, що повертаються. Якщо L = min(N, M), то можливі наступні варіанти:

- *'reduced'*- повертає Q та R з розмірами (M, L), (L, N) (встановлений за замовчуванням);
- *'complete'*- повертає Q та R з розмірами (M, M), (M, N);
- *'r'*- Повертає тільки R з формою (L, N);
- *'raw'*- Повертає h, tau з розмірами (N, M), (L,);
- *'full'*- псевдонім 'reduced' (є для зворотної сумісності);
- *'economic'*- Повертає h з raw (вважається застарілим).

Повертає:

результат - кортеж із двох масивів NumPy
Перший масив у кортежі - це масив Q, другий - масив R. У режимі 'raw' повертається кортеж з масивів h, tau; h містить відбиття Хаусхолдера, які використовуються для генерації масивів Q і R, а масив tau містить коефіцієнти масштабування для відбиття. У режимі економічний повертається тільки h.

Зауваження

У разі неможливості розкладання зазначеного масиву виникає виняток LinAlgError.

Дивіться також:
cholesky,svd

Приклади

```
>>> import numpy as np
>>>
>>> A = [[12, -51, 4],
... [6, 167, -68],
... [-4, 24, -41]]
>>>
>>> Q, R = np.linalg.qr(A)
>>>
>>> Q
array([[-0.85714286, 0.39428571, 0.33142857],
       [-0.42857143, -0.90285714, -0.03428571],
       [0.28571429, -0.17142857, 0.94285714]])
>>>
```

```
>>> np.dot(QT, Q)
array([[ 1.00000000e+00, -2.08166817e-17, -5.55111512e-17],
       [-2.08166817e-17, 1.00000000e+00, 2.77555756e-17],
       [-5.55111512e-17, 2.77555756e-17, 1.00000000e+00]])
>>>
>>> R
array([[-14., -21., 14.],
       [0., -175., 70.],
       [0., 0., -35.]])
>>>
>>> np.dot(QT, A)
array([[-1.40000000e+01, -2.10000000e+01, 1.40000000e+01],
       [5.55111512e-16, -1.75000000e+02, 7.00000000e+01],
       [8.88178420e-16, 0.00000000e+00, -3.50000000e+01]])
>>>
>>> np.dot(Q, R)
array([[ 12., -51., 4.],
       [6., 167., -68.],
       [-4., 24., -41.]])
>>>
>>>
>>> np.linalg.qr(A, mode = 'raw')
(array([[-1.40000000e+01, 2.30769231e-01, -1.53846154e-01],
       [-2.10000000e+01, -1.75000000e+02, 5.55555556e-02],
       [ 1.40000000e+01, 7.00000000e+01, -3.50000000e+01]]),
array([1.85714286, 1.99384615, 0.]))
>>>
>>>
>>> np.linalg.qr(A, mode = 'economic')
array([[-1.40000000e+01, -2.10000000e+01, 1.40000000e+01],
       [2.30769231e-01, -1.75000000e+02, 7.00000000e+01],
       [-1.53846154e-01, 5.55555556e-02, -3.50000000e+01]])
```

numpy.dot

numpy.dot(a, b, out=None)

Функція dot() обчислює скалярний добуток двох масивів.

Перед використанням цієї функції необхідно враховувати такі нюанси:

- Якщо a або b є числом, то np.dot(a, b) стає еквівалентною функції np.multiply(a, b) або команді a*b, які є кращими;
- Якщо a чи b є одномірними масивами, тобто. векторами, то виконується внутрішній (по суті скалярний) добуток векторів і np.dot(a, b) стає еквівалентною функції np.inner(a, b), але без підтримки комплексних чисел.
- Якщо a - це N-вимірний масив, b - це одновимірний масив, то сума творів елементів a та b обчислюється по останній осі a.
- Якщо a та b є двовимірними масивами, тобто. матрицями, то виконується матричне множення і np.dot(a, b) стає еквівалентною функції np.matmul(a, b) або команді a@b, які є кращими;
- Якщо a - це N-вимірний масив, а b - це M-вимірний масив (за умови, що M >= 2), то обчислюється сума творів елементів останньої осі a та елементів з другої по останню осей b, тобто. еквівалентно команді: dot(a, b)[i,j,k,m] = sum(a[i,j,:] * b[k,:,m])

Параметри:

a, b - числа, масиви NumPy або подібні до масивів об'єкти.
Вхідні дані
out – масив NumPy, необов'язковий параметр.
Масив, в який можна помістити результат функції. Даний масив повинен відповідати формі та типу даних результуючого масиву функції, а також обов'язково бути C-суміжним, тобто. зберігати дані у рядковому 3 стилі. Вказівка даного параметра дозволяє уникнути зайвої операції присвоювання тим самим трохи прискорюючи роботу вашого коду. Корисний параметр якщо ви часто звертаєтеся до функції в циклі.

Повертає:

результат - масив NumPy або число
Скалярний твір a та b. Якщо це числа чи одновимірні масиви, то повертається число, інакше повертається масив.

Зауваження

Довжина останньої осі масиву a, а як і довжина всіх осей масиву b крім першої повинні збігатися, інакше виникне виняток ValueError.

Дивіться також:

vdot,linalg.multi_dot,matmul

Приклади

Розмноження числа на число або числа на масив:

```
>>> import numpy as np
>>>
>>> np.dot(2, 2)
4
>>>
>>> np.dot(2, [2, 3, 4])
array([4, 6, 8]) # Краще використовувати np.multiply() або a*b
```

Розмноження одновимірних масивів (векторів):

```
>>> a = np.array([2, 3, 4])
>>> b = np.array([5, 6, 7])
>>>
>>> np.dot(a, b)
56
>>> # Еквівалентно:
... np.inner(a, b) # краще
56
>>>
>>> # Еквівалентно:
... np.sum(a*b)
56
```

Розмноження двовимірних масивів (матриць):

```
>>> a = np.arange(1, 7).reshape(2, 3)
>>> a
array([[1, 2, 3],
       [4, 5, 6]])
>>>
>>> b = np.arange(1, 7).reshape(3, 2)
```

```
>>> b
array([[1, 2],
       [3, 4],
       [5, 6]])
>>>
>>> np.dot(a, b)
array([[22, 28],
       [49, 64]])
>>>
>>> np.dot(b, a)
array([[ 9, 12, 15],
       [19, 26, 33],
       [29, 40, 51]])
>>>
>>> # Еквівалентно:
...np.matmul(a, b) # краще
array([[22, 28],
       [49, 64]])
```

Розмноження одновимірного масиву на двовимірні та багатовимірні:

```
>>> a = np.array([2, 3])
>>> b = np.array([[1, 2], [3, 4]])
>>>
>>> np.dot(a, b)
array([11, 16])
>>>
>>> c = np.arange(1, 7).reshape(2, 3)
>>> c
array([[1, 2, 3],
       [4, 5, 6]])
>>>
>>> np.dot(a, c)
array([14, 19, 24])
>>>
>>> np.dot(c, a)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: shapes (2,3) and (2,) no aligned: 3 (dim 1) != 2 (dim 0)
>>>
>>> # Довжина останньої осі 'c' повинна збігатися з довжиною 'a':
>>> c = c.reshape(3, 2)
>>> c
array([[1, 2],
       [3, 4],
       [5, 6]])
>>>
>>> np.dot(c, a)
array([ 8, 18, 28])
>>> a
array([2, 3])
>>>
>>>
>>> d = np.arange(1, 13).reshape(3, 2, 2)
>>> d
array([[[ 1, 2],
        [3, 4]],

       [[ 5, 6],
        [7, 8]],

       [[ 9, 10],
        [11, 12]]])
```

```
>>>
>>> np.dot(a, d)
array([[11, 16],
       [31, 36],
       [51, 56]])
>>> np.dot(d, a)
array([[ 8, 18],
       [28, 38],
       [48, 58]])
>>>
>>>
>>> d = np.arange(1, 13).reshape(2, 3, 2)
>>> d
array([[[ 1, 2],
        [3, 4],
        [5, 6]],

       [[7, 8],
        [9, 10],
        [11, 12]]])
>>>
>>> np.dot(a, d)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: shapes (2,) and (2,3,2) не виключено: 2 (dim 0) != 3 (dim 1)
>>> np.dot(d, a)
array([[ 8, 18, 28],
       [38, 48, 58]])
>>>
>>>
>>> d = np.arange(1, 13).reshape(2, 2, 3)
>>> d
array([[[ 1, 2, 3],
        [4, 5, 6]],

       [[7, 8, 9],
        [10, 11, 12]]])
>>>
>>> np.dot(a, d)
array([[14, 19, 24],
       [44, 49, 54]])
>>> np.dot(d, a)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: shapes (2,2,3) and (2,) no aligned: 3 (dim 2) != 2 (dim 0)
```

Скалярний добуток багатовимірних масивів:

```
>>> a = np.arange(3*2*4).reshape(3, 2, 4)
>>> a
array([[[ 0, 1, 2, 3],
        [4, 5, 6, 7]],

       [[ 8, 9, 10, 11],
        [12, 13, 14, 15]],

       [[16, 17, 18, 19],
        [20, 21, 22, 23]]])
>>>
>>> b = np.arange(2*4*4).reshape(2, 4, 4)
>>> b
array([[[ 0, 1, 2, 3],
        [4, 5, 6, 7],
```

```
         [8, 9, 10, 11],
         [12, 13, 14, 15]],

        [[16, 17, 18, 19],
         [20, 21, 22, 23],
         [24, 25, 26, 27],
         [28, 29, 30, 31]]])
>>>
>>> np.dot(a, b)
array([[[[ 56, 62, 68, 74],
         [152, 158, 164, 170]],

         [[ 152, 174, 196, 218],
          [504, 526, 548, 570]]],


        [[[ 248, 286, 324, 362],
          [856, 894, 932, 970]],

         [[ 344, 398, 452, 506],
          [1208, 1262, 1316, 1370]]],


        [[[ 440, 510, 580, 650],
          [1560, 1630, 1700, 1770]],

         [[ 536, 622, 708, 794],
          [1912, 1998, 2084, 2170]]]])
```

numpy.linalg.eig

`numpy.linalg.eig(a)`

Функція linalg.eig() обчислює власні числа (значення) та власні вектори квадратної матриці.

Власне число квадратної матриці a та її власний вектор пов'язані рівністю np.dot(a, v) = w*v.

примітка:
dot(a,v) – скалярний добуток матриці a на вектор v, w*v – матриця добуток матриць w та v)

Якщо у матриці кілька власних чисел і кілька власних векторів, то ця функція повертає масив власних значень і масив власних векторів (необов'язково впорядкованих). І тут виконуються рівності np.dot(a[:,:], v[:,i]) = w[i] * v[:,i], всім і з інтервалу [0, ..., M - 1], де M - розмір матриці.

Параметри:

a - масив NumPy або подібний масиву об'єкт.
Це може бути тільки " квадратний " двомірний масив, тобто. Квадратна матриця.
Якщо це багатовимірний масив, дві його останні осі повинні бути рівні, в цьому випадку він розглядається як масив матриць і всі обчислення виконуються окремо для кожної з них.

Повертає:

w - масив NumPy
Власні числа вхідної матриці.
v - масив NumPy
Власні вектори вхідної матриці.

Зауваження

Дивіться також:

eigh,eigvals,eigvalsh

Приклади

```
>>> import numpy as np
>>> від numpy import linalg as LA
>>>
>>> a = np.array([[2, 3, 5], [7, 11, 13], [17, 19, 23]])
>>> w,v = LA.eig(a)
>>> w
array([36.81172799, -1.91702763, 1.10529963])
>>> v
array([[-0.16510917, -0.60574007, 0.36952037],
       [-0.4789958, -0.3769456, -0.8244886],
       [-0.86214963, 0.70070748, 0.42857116]])
>>> np.dot(a,v)
array([[-6.07795389, 1.16122046, 0.40843073],
       [-17.6326632, 0.72261513, -0.91130695],
       [-31.73721763, -1.3432756, 0.47369955]])
>>>
>>> w*v
array([[-6.07795389, 1.16122046, 0.40843073],
       [-17.6326632, 0.72261513, -0.91130695],
       [-31.73721763, -1.3432756, 0.47369955]])
```

Іноді візуально здається, що вищеописані рівності не виконуються:

```
>>> a = np.array([[1, 1, 1], [2, 2, 2], [3, 3, 3]]))
>>>
>>> w,v = LA.eig(a)
>>> np.dot(a,v)
array([[ 1.60356745e+00, 1.11022302e-16, 0.00000000e+00],
       [3.20713490e+00, 2.22044605e-16, 0.00000000e+00],
       [ 4.81070235e+00, 0.00000000e+00, -2.22044605e-16]])
>>> w*v
array([[ 1.60356745e+00, -4.10074414e-16, -3.96241817e-17],
       [3.20713490e+00, -1.03002928e-16, 1.46033608e-16],
       [4.81070235e+00, 5.13077342e-16, -1.06409427e-16]])
```

Однак значення перших стовпців рівні, а значення другого і третього стовпців настільки малі, що можуть бути прирівняні до 0.

Завжди слід пам'ятати про помилки округлення:

```
>>> a = np.array([[1 + 1e-9, 0], [0, 1 - 1e-9]])
>>>
>>> w,v = LA.eig(a)
>>>
>>> w
array([1., 1.])
>>>
>>> v
```

```
array([[1., 0.],
       [0., 1.]])
```

Але насправді w = [1.000000001, 0.999999999].

---

numpy.linalg.pinv

`numpy.linalg.pinv(a, rcond=1e-15)`

Функція linalg.pinv() обчислює псевдозворотну матрицю (Мура-Пенроуза).

Для матриці AA псевдозворотною матрицею є така матриця A+A+, для якої виконуються такі умови:

AA+A=AAA+A=A;

A+AA+=A+A+AA+=A+;

(AA+)*=AA+(AA+)*=AA+, тобто. AA+AA+ теж ермітова;

(A+A)*=A+A(A+A)*=A+A, тобто A+AA+A теж ермітова

Обчислення відбувається на основі сингулярного (SVD) розкладання матриці a, якщо розкладання неможливе, повертається помилка LinAlgError.

Параметри:

a- *масив NumPy або подібний масиву об'єкт.*
Двовимірний чи багатовимірний масив. Якщо це багатовимірний масив, він розглядається як масив матриць з розміром рівним довжині останніх двох осей.

rcond- *масив NumPy або подібний масиву об'єкт.*
Коефіцієнт "відкидання" малих значень масиву a. Якщо якийсь елемент з a менше ніж`rcond*max(a)` цей елемент буде оброблятися як 0.

Повертає:

результат- *масив NumPy або число*
Псевдоботна матриця або масив Псевдооборотних матриць.

Дивіться також:
linalg.inv,linalg.tensorinv

Приклади

```
>>> importnumpyasnp
>>> fromnumpyimportlinalgasLA
>>>
>>>a=np.random.randint(0,10, size=(3,3))
```

```
>>>a
array([[0,9,1],
       [8,2,2],
       [1,6,3]])
>>>
>>>a_pinv=LA.pinv(a)
>>>a_pinv
array([[0.03947368,0.13815789,-0.10526316],
       [0.14473684,0.00657895,-0.05263158],
       [-0.30263158,-0.05921053,0.47368421]])
>>>
>>>
>>> # Перевіримо виконання 1-ї умови:
... np.allclose(np.dot(a, np.dot(a_pinv, a)), a)
True
>>>
>>> # Перевіримо виконання 2-ї умови:
... np.allclose(np.dot(a_pinv, np.dot(a, a_pinv)), a_pinv)
True
>>>
>>> # Перевіримо виконання 3-ї умови:
... np.allclose(np.dot(a, a_pinv).T, np.dot(a, a_pinv))
True
>>>
>>> # Перевіримо виконання 4-ї умови:
... np.allclose(np.dot(a_pinv, a).T, np.dot(a_pinv, a))
True
```

---

numpy.linalg.LinAlgError

`exception numpy.linalg.LinAlgError`

Об'єкт linalg.LinAlgError генерує винятки Python, викликані функціями модуля linalg.

Цей об'єкт утворений класом винятків загального призначення Python і є похідним від нього. Цей клас викликається, тільки якщо подальша робота будь-якої функції модуля linalg неможлива.

Дивіться також:
linalg.inv,linalg.pinv

Приклади

```
>>> importnumpyasnp
>>> fromnumpyimportlinalgasLA
>>>
>>>a=[[0,0], [0,0]]
>>>
>>> try:
... LA.inv(a)
...exceptLA.LinAlgError:
...print('Матриця "a" або вироджена або прямокутна')
...
Матриця"a"є або виродженою або прямокутною
```

---

numpy.linalg.solve

`numpy.linalg.solve(a, b)`

Функція linalg.solve() вирішує лінійне матричне рівняння (систему лінійних рівнянь).

Ця функція обчислює значення невідомих лише квадратних, невироджених матриць з повним рангом, тобто. тільки якщо матриця розміром {m, m} має ранг рівний m. Якщо хоча б одна з цих умов не виконується, повертається помилка LinAlgError.

Система лінійних рівнянь може бути записана в матричній формі:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Або у більш короткій формі$Ax=b$, де A – це матриця коефіцієнтів (матриця системи), x – стовпець невідомих, b – стовпець вільних членів.

Параметри:

a- *масив NumPy або подібний масиву об'єкт.*
Матриця коефіцієнтів - квадратний масив або багатовимірний масив, у якого дві останні осі рівні.

b- *масив NumPy або подібний масиву об'єкт.*
Стовпець вільних членів - одномірний масив, довжина якого збігається з довжиною`a`. Це може бути багатовимірний масив, але в цьому випадку його остання або передостання вісь повинна дорівнювати останній осі масиву a.

Повертає:

x- *масив NumPy.*
Вирішення матричного рівняння (системи рівнянь) Ax=b. Форма масива, що повертається, залежить від форми масивів a і b.

Дивіться також:
linalg.lstsq,dot

Приклади

Розв'яжемо систему рівнянь

$$\begin{cases} x_0 + 2x_1 - 3x_2 = 4 \\ 2x_0 + x_1 + 2x_2 = 3 \\ 3x_0 - 2x_1 - x_2 = 9 \end{cases}$$

```
>>> importnumpyasnp
>>> fromnumpyimportlinalgasLA
>>>
>>>a=[[1,2,-3],
      [2,1,2],
      [3,-2,-1]]
>>>
>>>b=[4,3,9]
>>>
>>>x=LA.solve(a, b)
>>>x
array([2.475,-0.4  ,-0.775])
```

Виконаємо перевірку, помножимо матрицю коефіцієнтів на отриманий стовпець рішень. Даний твір повинен виявитися рівним стовпцю b:

```
>>>np.dot(a, x)
array([4.,3.,9.])
```

Ми можемо вирішувати декілька систем одночасно.

Допустимо у нас є кілька систем рівнянь:

$$\begin{cases} x_0 + 3x_1 = 7 \\ 2x_0 - 5x_1 = -2 \end{cases}$$

$$\begin{cases} 2x_0 + x_1 = 5 \\ 5x_0 - 2x_1 = -1 \end{cases}$$

$$\begin{cases} 4x_0 - 3x_1 = 1 \\ 2x_0 + x_1 = 2 \end{cases}$$

```
>>>a=np.array([[[1,3], [2,-5]],
... [[2,1], [5,-2]],
... [[4,-3], [2,1]]])
>>>
>>>a.shape
(3,2,2)
>>>
>>>b=np.array([[7,-2],
... [5,-1],
... [1,2]])
>>>
>>>b.shape
(3,2)
>>>
>>>
>>>np.dot(a[0], x[0])
array([7.,-2.])
>>>np.dot(a[1], x[1])
array([5.,-1.])
>>>np.dot(a[2], x[2])
array([1.,2.])
```