Numpy

Βιβλιοθήκη Python για μαθηματικούς υπολογισμούς

Numpy

λύση γραμμικού συστήματος

$$3x + \ \ y = 9$$

$$x + 2y = 8$$

Λύση γραμμικού συστήματος

$$x = 2, \quad y = 3$$

Numpy

```
A = np.array([[3,1], [1,2]])
b = np.array([[9],[8]])
```

Λύση γραμμικού συστήματος
Βιβλιοθήκη `linalg`

```
x = np.linalg.solve(A, b)
```

Επιστρέφει

```
[[2],
 [3]]
```

Numpy

- Έλεγχος λύσης

`np.dot(A,x)` δίνει

```
array([ [9.],
        [8.]])
```

Numpy

Αντίστροφος

- Υπολογισμός αντιστρόφου

`B=np.linalg.inv(A)` επιστρέφει

```
[[ 0.4 -0.2]
 [-0.2  0.6]]
```

- Έλεγχος

`np.dot(A,B)` επιστρέφει

```
[[ 1. 0.]
 [0.  1.]]
```

Numpy

λύση συστήματος

- Λύση συστήματος $Ax = b$ $x = A^{-1}b$

- `np.dot(B,b)` επιστρέφει

```
[[2],
 [3]]
```

Numpy

Πιο “δύσκολα” συστήματα

$$x + 3y + 5z = 10$$
$$2x + 5y + \ \ z = \ \ 8$$
$$2x + 3y + 8z = \ \ 3$$

Λύση γραμμικού συστήματος

$$x = -\frac{232}{25}, \quad y = \frac{129}{25}, \quad z = \frac{19}{25}$$

Numpy

- Υπολογισμός αντιστρόφου

`B=np.linalg.inv(A)` τότε

`np.dot(A,B)` επιστρέφει

```
[[  1.00000000e+00   1.11022302e-16  -2.22044605e-16]
 [ -1.38777878e-16   1.00000000e+00   0.00000000e+00]
 [ -2.22044605e-16   0.00000000e+00   1.00000000e+00]]
```

- `np.dot(B,b)` επιστρέφει

```
[[-9.28]
 [ 5.16]
 [ 0.76]]
```

επίσης **`np.linalg.solve(A,b)`** επιστρέφει

```
[[-9.28]
 [ 5.16]
 [ 0.76]]
```

Αν `x=np.dot(B,b)` η `np.dot(A,x)-b` επιστρέφει

```
[[ -3.5527136788e-15]
 [ -7.1054273576e-15]
 [ -7.1054273576e-15]]
```

Αν `x=np.linalg.solve(A,b)` η `np.dot(A,x)-b` επιστρέφει

```
[[  0.0000000000e+00]
 [  0.0000000000e+00]
 [ -1.7763568394e-15]]
```

Η χρησιμοποίηση του αντίστροφου για τον υπολογισμό της λύσης ενός γραμμικού συστήματος πρέπει να αποφεύγετε.

Numpy

Περισότερα “προβληματα”

- Θεωρούμε τον πίνακα

```
A=np.array([[0, 1],[1,1]])
```

και `b=np.array([[1],[2]])`

- Η λύση είναι

```
x=[[ 1.]
   [ 1.]]
```

Numpy

Περισότερα “προβληματα”

- Θεωρούμε τον πίνακα

```
A=np.array([[0.001, 1],[1,1]])
```

και `b=np.array([[1],[2]])`

- Με απαλοιφή Gauss η λύση που παίρνουμε είναι

```
x=[[ 1.001001]
   [ 0.998999]]
```

Το σφάλμα `Ax-b` είναι

```
[[  0.00000000e+00]
 [ -2.26485497e-14]]
```

Numpy

Περισότερα “προβληματα”

- Θεωρούμε τον πίνακα

```
A=np.array([[0.001, 1],[1,1]])
```

και `b=np.array([[1],[2]])`

- Με χρήση της `linalg.solve` η λύση που παίρνουμε είναι

```
x=[[ 1.001001]
  [ 0.998999]]
```

Το σφάλμα `Ax-b` είναι

```
[[  2.22044605e-16]
  [  0.00000000e+00]]
```

Numpy

Περισότερα “προβληματα”

- Θεωρούμε τον πίνακα

```
A=np.array([[1,1],[0.001, 1]])
```

και `b=np.array([[2],[1]])`

- Με απαλοιφή Gauss η λύση που παίρνουμε είναι

```
x=[[ 1.001001]
   [ 0.998999]]
```

Το σφάλμα `Ax-b` είναι

```
[[ 0.0]
 [ 0.0]]
```

Numpy

- Έλεγχος ψηφίων στην εκτύπωση
  `np.set_printoptions(precision=10)`