

Sveučilište u Zagrebu  
FAKULTET STROJARSTVA I BRODOGRADNJE  
Zavod za robotiku i automatizaciju proizvodnih sustava  
Katedra za strojarsku automatiku

Objektno programiranje:  
**Linearna algebra i konveksna optimizacija  
u Python-u**

M. Essert; V. Milić

Zagreb, 2009.



# Sadržaj

- 1 Linearna algebra: `linalg.py`
- 2 Konveksna optimizacija: `cvxopt.py`
- 3 Literatura

# Kreiranje matrica

Matrica  $\mathbf{A}$  sa  $n$  redaka,  $m$  stupaca i s elementima  $a_{ij}$  zapisuje se kao

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nm} \end{bmatrix} = [a_{ij}].$$

```
A = mat('[1 3 5; 2 5 1; 2 3 8]') #slicno kao u MATLAB-u
```

```
matrix([[1, 3, 5],  
        [2, 5, 1],  
        [2, 3, 8]])
```

```
B = matrix([[2,4,6], [8,10,12], [14,16,18]])
```

```
matrix([[ 2,  4,  6],  
        [ 8, 10, 12],  
        [14, 16, 18]])
```

```
nul_matrica = mat(zeros((5,5))) #nul-matrica
```

```
matrica_jedinica = mat(ones((3,2))) #matrica jedinicna
```

```
jedinicna_matrica = mat(identity(5)) #jedinicna matrica ili  
jedinicna_matrica = eye(5)
```

```
dijagonalna_matrica = mat(diag([1.,3,5]))
```

# Jednakost matrica

Matrica **A** dimenzije  $n \times m$  je jednaka matrici **B** dimenzije  $p \times q$  ako vrijedi  $n = p$  i  $m = q$  i

$$a_{ij} = b_{ij}, \text{ za sve } 1 \leq i \leq n, 1 \leq j \leq m$$

U tom slučaju pišemo

$$\mathbf{A} = \mathbf{B}.$$

U Python-u vrlo jednostavno:

`A == B`

```
matrix([[False, False, False],
        [False, False, False],
        [False, False, False]], dtype=bool)
```

# Zbrajanje matrica

Neka je  $\mathbf{A} \in \mathbb{R}^{n \times m}$  i  $\mathbf{B} \in \mathbb{R}^{p \times q}$ . Ako je  $n = p$  i  $m = q$  onda matricu  $\mathbf{C} \in \mathbb{R}^{n \times m}$  s elementima

$$c_{ij} = a_{ij} + b_{ij}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq m,$$

zovemo zbrojem ili sumom matrica  $\mathbf{A}$  i  $\mathbf{B}$  i pišemo

$$\mathbf{C} = \mathbf{A} + \mathbf{B}.$$

U Python-u vrlo jednostavno:

$\mathbf{C} = \mathbf{A} + \mathbf{B}$

```
matrix([[ 3,  7, 11],
        [10, 15, 13],
        [16, 19, 26]])
```

# Umnožak matrice sa skalarom

Ako je  $\mathbf{A} \in \mathbb{R}^{n \times m}$  i  $c \in \mathbb{R}$ , onda matricu  $\mathbf{D} \in \mathbb{R}^{n \times m}$  s elementima

$$d_{ij} = c a_{ij}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq m,$$

zovemo umnožak matrice  $\mathbf{A}$  s skalarom  $c$  i označavamo

$$\mathbf{D} = c \mathbf{A}.$$

U Python-u vrlo jednostavno:

```
c = 3
```

```
D = c*A
```

```
matrix([[ 3,  9, 15],
        [ 6, 15,  3],
        [ 6,  9, 24]])
```

# Umnožak matrica

Neka je  $\mathbf{A} \in \mathbb{R}^{n \times m}$  i  $\mathbf{B} \in \mathbb{R}^{m \times p}$ , umnožak ili produkt matrica  $\mathbf{A}$  i  $\mathbf{B}$  je matrica

$$\mathbf{C} = \mathbf{A} \mathbf{B} \in \mathbb{R}^{n \times p}$$

čiji elementi su određeni formulom

$$c_{ij} = a_{i1} b_{1j} + a_{i2} b_{2j} + a_{i3} b_{3j} + \dots + a_{im} b_{mj} = \sum_{k=1}^m a_{ik} b_{kj}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq p,$$

U Python-u vrlo jednostavno:

$$\mathbf{C} = \mathbf{A} * \mathbf{B}$$

```
matrix([[ 96, 114, 132],
        [ 58,  74,  90],
        [140, 166, 192]])
```

# Transponirana matrica

Neka je  $\mathbf{A} \in \mathbb{R}^{n \times m}$ . Matrica  $\mathbf{A}^T \in \mathbb{R}^{m \times n}$  se naziva transponirana matrica  $\mathbf{A}$ , ako je svaki redak od  $\mathbf{A}^T$  jednak odgovarajućem stupcu matrice  $\mathbf{A}$ .

U Python-u vrlo jednostavno:

$\mathbf{A.T}$

```
matrix([[1, 2, 2],  
        [3, 5, 3],  
        [5, 1, 8]])
```

# Determinanta matrice

Neka su  $a_{ij}$  elementi kvadratne matrice i neka je  $M_{ij} = |\mathbf{A}_{ij}|$  determinanta matrice pomicanjem u lijevo  $i$ -tog retka i  $j$ -tog stupca. Tada za bilo koji redak  $i$

$$|\mathbf{A}| = \sum_j (-1)^{i+j} a_{ij} M_{ij}$$

U Python-u vrlo jednostavno:

```
linalg.det(A)
```

```
-25.0
```

# Inverz matrice

Inverz kvadratne matrice  $\mathbf{A}$  je matrica  $\mathbf{A}^{-1}$  takva da je  $\mathbf{A} \mathbf{A}^{-1} = \mathbf{I}$  gdje je  $\mathbf{I}$  jedinična matrica.

U Python-u vrlo jednostavno:

```
linalg.inv(A)
```

```
matrix([[ -1.48,  0.36,  0.88],
        [ 0.56,  0.08, -0.36],
        [ 0.16, -0.12,  0.04]])
```

A.I

```
matrix([[ -1.48,  0.36,  0.88],
        [ 0.56,  0.08, -0.36],
        [ 0.16, -0.12,  0.04]])
```

# Sustavi linearnih algebarskih jednažbi

Primjer:

$$x_1 + 3x_2 + 5x_3 = 10,$$

$$2x_1 + 5x_2 + x_3 = 8,$$

$$2x_1 + 3x_2 + 8x_3 = 3.$$

U Python-u vrlo jednostavno:

```
A = mat('[1 3 5; 2 5 1; 2 3 8]')
```

```
b = mat('[10;8;3]')
```

```
linalg.solve(A,b)
```

```
matrix([[ -9.28],
        [  5.16],
        [  0.76]])
```

A.I\*b

```
matrix([[ -9.28],
        [  5.16],
        [  0.76]])
```

# Svojsstvene (vlastite) vrijednosti i vektori

Problem svojsstvenih vrijednosti (vektora):

$$\mathbf{A} \mathbf{v} = \lambda \mathbf{v}.$$

Svojsstvene vrijednosti su rješenja karakterističnog polinoma

$$|\mathbf{A} - \lambda \mathbf{I}| = 0$$

```
from scipy import*
from scipy import linalg

A = mat('[1 5 2; 2 4 1; 3 6 2]')
la,v = linalg.eig(A)
l1,l2,l3 = la

print l1, l2, l3
      (7.95791620491+0j) (-1.25766470568+0j) (0.299748500767+0j)

print v[:,0]
      [-0.5297175  -0.44941741 -0.71932146]

print v[:,1]
      [-0.90730751  0.28662547  0.30763439]

print v[:,2]
      [ 0.28380519 -0.39012063  0.87593408]

print sum(abs(v**2),axis=0)
      [ 1.  1.  1.]
```

# Rješavanje homogenog sustava linearnih običnih diferencijalnih jednačbi

Običnu linearnu Diferencijalnu jednačbu drugog reda

$$\ddot{y}(t) + \dot{y}(t) + 10y(t) = 0,$$

možemo zapisati kao sustav od dvije diferencijalne jednačbe prvog reda

$$\begin{aligned}\dot{x}_1 &= x_2, \\ \dot{x}_2 &= -10x_1 - x_2,\end{aligned}$$

gdje su  $x_1 = y$  i  $x_2 = \dot{y}$ . U matričnom zapisu gornji sustav je sljedećeg oblika

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -10 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

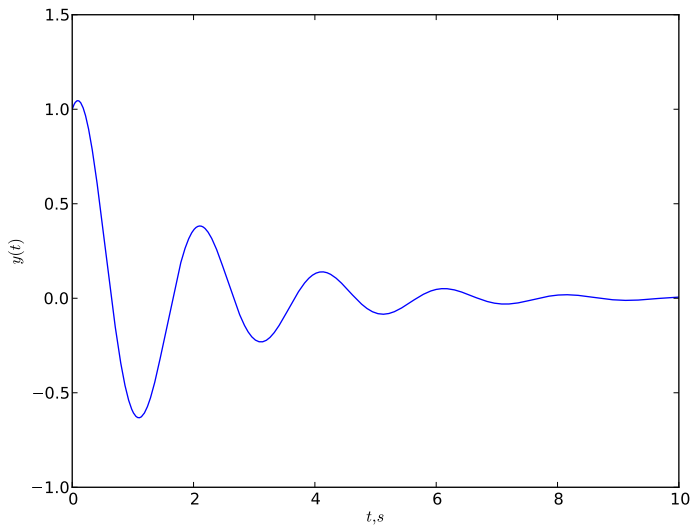
Rješenje je oblika:

$$\mathbf{x}(t) = e^{t\mathbf{A}} \mathbf{x}(0)$$

```
from scipy import*
from scipy.linalg import expm
import matplotlib.pyplot as plt

A = mat('[0 1;-10 -1]')
x0 = mat('[1;1]')
xp = mat('[1 0]')
t_val=arange(0,10,0.01)
y = [xp*expm(t*A)*x0 for t in t_val]

plt.plot(t_val,array(y).reshape(10/0.01,))
plt.xlabel('$t, s$')
plt.ylabel('$y(t)$')
plt.show()
```



# Kreiranje matrica

```
from cvxopt import matrix

A = matrix([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], (2,3))
print A
    [ 1.00e+000  3.00e+000  5.00e+000]
    [ 2.00e+000  4.00e+000  6.00e+000]

B = matrix([ [1.0, 2.0], [3.0, 4.0] ])
print B
    [ 1.00e+00  3.00e+00]
    [ 2.00e+00  4.00e+00]
```

# Problem linearnog programiranja

Definicija problema:

$$\begin{aligned}
 \min \quad & \mathbf{c}^T \mathbf{x} \\
 \text{s.t.} \quad & \mathbf{G}\mathbf{x} + \mathbf{s} = \mathbf{h} \\
 & \mathbf{A}\mathbf{x} = \mathbf{b} \\
 & \mathbf{s} \succeq \mathbf{0}
 \end{aligned}$$

Primjer:

$$\begin{aligned}
 \min \quad & -4x_1 - 5x_2 \\
 \text{s.t.} \quad & 2x_1 + x_2 \leq 3 \\
 & x_1 + 2x_2 \leq 3 \\
 & x_1 \geq 0, \quad x_2 \geq 0
 \end{aligned}$$

```
from cvxopt import matrix, solvers
```

```
c = matrix([-4., -5.])
```

```
G = matrix([[2., 1., -1., 0.], [1., 2., 0., -1.]])
```

```
h = matrix([3., 3., 0., 0.])
```

```
sol = solvers.lp(c, G, h)
```

	pcost	dcost	gap	pres	dres	k/t
0:	-8.1000e+000	-1.8300e+001	4e+000	0e+000	8e-001	1e+000
1:	-8.8055e+000	-9.4357e+000	2e-001	7e-017	4e-002	3e-002
2:	-8.9981e+000	-9.0049e+000	2e-003	2e-016	5e-004	4e-004
3:	-9.0000e+000	-9.0000e+000	2e-005	1e-016	5e-006	4e-006
4:	-9.0000e+000	-9.0000e+000	2e-007	5e-016	5e-008	4e-008






Optimal solution found.

```
print sol['x']
```

```
[ 1.00e+000]
```

```
[ 1.00e+000]
```

# Literatura

-  NumPy Reference, Release 1.3. *Written by the NumPy community*, dostupno na: <http://docs.scipy.org/doc/numpy-1.3.x/numpy-ref.pdf>
-  SciPy Reference Guide, Release 0.7. *Written by the SciPy community*, dostupno na: <http://docs.scipy.org/doc/scipy-0.7.x/scipy-ref.pdf>
-  K. Horvatić. *Linearna algebra*. Golden marketing–Tehnička knjiga, Zagreb, 2004.
-  H. P. Langtangen. *Python Scripting for Computational Science*. Springer-Verlag, Berlin Heidelberg, 2008.
-  C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, 2000.

## WEB stranice

<http://docs.scipy.org/doc/scipy/reference/tutorial/linalg.html>

<http://abel.ee.ucla.edu/cvxopt>

<http://cvxmod.net/index.html>