# Мататематички методи за машинско учење 2023

## Домаћи задатак број 3

```
import numpy as np
import numpy.random as rndm
```

**Задатак 1.** Конструисати матрицу $A \in M_{7\times 6}$ чију базу простора колона чине прве три колоне матрице $B \in M_{7\times 6}$, а базу језгра чине последње три врсте матрице $B$. Матрица $B$ се генерише наредном кодном ћелијом. Кодом се такође проверава да ли случајно дефинисана матрица има прве три линеарно независне колоне и последње три линеарно независне врсте.

(10 поена)

```
# assert je ključna reč u Pythonu koja se koristi za proveru uslova. Ako je uslov zadovoljen,
# assert neće imati efekta i program će nastaviti sa izvršavanjem. Ako uslov nije zadovoljen,
# assert će izazvati izuzetak (AssertionError) i prekinuti izvršavanje programa.

B=rndm.randn(7,6)-5*np.eye(7,6)
print(np.linalg.matrix_rank(B[:,:3]))
print(np.linalg.matrix_rank(B[-3:]))
```

```
3
3
```

```
import numpy as np

# Generišemo matricu B
np.random.seed(42)  # Postavljamo seed za reprodukciju rezultata
B = np.random.randn(7, 6) - 5 * np.eye(7, 6)

# Proveravamo da li prve tri kolone matrice B čine bazu
assert np.linalg.matrix_rank(B[:, :3]) == 3

# Proveravamo da li zadnje tri vrste matrice B čine bazu jezgra
assert np.linalg.matrix_rank(B[-3:]) == 3

# Konstruišemo matricu A tako što uzimamo prve tri kolone matrice B kao bazu prostora kolona
A_column_space = B[:, :3]

# Da bismo pronašli bazu jezgra matrice B, moramo prvo pronaći njen null prostor
```

```
# Rešavamo sistem jednačina Bx = 0
_, _, Vh = np.linalg.svd(B)
B_null_space = Vh[-3:].T

# A je konstruisana tako da njen jezgro ima bazu koja se sastoji od
zadnje tri vrste matrice B
# Dakle, A je rešenje Ax = 0, gde je x jedan od vektora iz baze jezgra
B
# Konstruišemo A tako što rešavamo sistem jednačina za svaki vektor x
iz baze jezgra B
A = np.empty((7, 6))
for i, x in enumerate(B_null_space.T):
    b = np.concatenate((-x[:3], np.zeros(4)))
    A[:, i+3], _, _, _ = np.linalg.lstsq(A_column_space, b,
rcond=None)

# Matrica A
print("Matrica A:")
print(A)
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[10], line 27
     25 for i, x in enumerate(B_null_space.T):
     26     b = np.concatenate((-x[:3], np.zeros(4)))
---> 27     A[:, i+3], _, _, _ = np.linalg.lstsq(A_column_space, b,
rcond=None)
     29 # Matrica A
     30 print("Matrica A:")

ValueError: could not broadcast input array from shape (3,) into shape
(7,)
```

**Задатак 2.** а) Дате су матрице Гаусове трансформације \begin{align} L_1&=\begin{bmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} +\begin{bmatrix}0\\-1\\0.5\\3\end{bmatrix}\begin{bmatrix}1&0&0&0\\ \end{bmatrix}=I_4+v_1e_1^T,\ L_2&=\begin{bmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} +\begin{bmatrix}0\\0\\3\\2\\ \end{bmatrix}\begin{bmatrix}0&1&0&0\end{bmatrix}=I_4+v_2e_2^T,\ L_3&=\begin{bmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} +\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\begin{bmatrix}0&0&1&0\\ \end{bmatrix}=I_4+v_3e_3^T. \end{align} Показати да за њих важи

$$\left(I_4+v_1e_1^T\right)\left(I_4+v_2e_2^T\right)\left(I_4+v_3e_3^T\right)=I_4+\sum_{k=1}^{3}v_ke_k^T.$$

(5 поена)

б) Дате су матрице Гаусове трансформације $L_k = I_n + v_k e_k^T, k = 1, 2, \ldots, m < n$, где је првих $k$ компоненети вектора $v_k$ једнако нули, док су $e_k$ вектори природне базе простора $R^n$. Показати једнакост

$$\left(I_n + v_1 e_1^T\right)\left(I_n + v_2 e_2^T\right) \cdots \left(I_n + v_m e_m^T\right) = I_n + \sum_{k=1}^{m} v_k e_k^T.$$

(5 поена)

**Задатак 3.** На основу неке од обрађених декомпозиција матрица одредити базу простора врста и базу простора колона матрице

$$A = \begin{pmatrix} 1 & 2 & 3 & -1 \\ 2 & -1 & -4 & 8 \\ -1 & 1 & 3 & -5 \\ -1 & 2 & 5 & -6 \\ -1 & -2 & -3 & 1 \end{pmatrix}.$$

(10 поена)

```
# LU
import numpy as np
import scipy.linalg

# Definišemo matricu A
A = np.array([[1, 2, 3, -1],
              [2, -1, -4, 8],
              [-1, 1, 3, -5],
              [-1, 2, 5, -6],
              [-1, -2, -3, 1]])

# Izračunavamo LU dekompoziciju matrice A
P, L, U = scipy.linalg.lu(A)

# Rang matrice A
rang = np.linalg.matrix_rank(A)

# Baza prostora kolona matrice A
baza_kolona = A[:, :rang]

# Baza prostora vrsta matrice A
baza_vrsta = A[:rang, :]

# Ispisujemo bazu prostora kolona matrice A
print("Baza prostora kolona matrice A (LU dekompozicija):")
print(baza_kolona)

# Ispisujemo baza prostora vrsta matrice A
```

```
print("Baza prostora vrsta matrice A (LU dekompozicija):")
print(baza_vrsta)
```

```
Baza prostora kolona matrice A (LU dekompozicija):
[[ 1  2  3]
 [ 2 -1 -4]
 [-1  1  3]
 [-1  2  5]
 [-1 -2 -3]]
Baza prostora vrsta matrice A (LU dekompozicija):
[[ 1  2  3 -1]
 [ 2 -1 -4  8]
 [-1  1  3 -5]]
```

```
# QR
import numpy as np

# Definišemo matricu A
A = np.array([[1, 2, 3, -1],
              [2, -1, -4, 8],
              [-1, 1, 3, -5],
              [-1, 2, 5, -6],
              [-1, -2, -3, 1]])

# Izračunavamo QR dekompoziciju matrice A
Q, R = np.linalg.qr(A)

# Rang matrice A
rang = np.linalg.matrix_rank(A)

# Baza prostora kolona matrice A
baza_kolona = A[:, :rang]

# Baza prostora vrsta matrice A
baza_vrsta = A[:rang, :]

# Ispisujemo bazu prostora kolona matrice A
print("Baza prostora kolona matrice A (QR dekompozicija):")
print(baza_kolona)

# Ispisujemo bazu prostora vrsta matrice A
print("Baza prostora vrsta matrice A (QR dekompozicija):")
print(baza_vrsta)
```

```
Baza prostora kolona matrice A (QR dekompozicija):
[[ 1  2  3]
 [ 2 -1 -4]
 [-1  1  3]
 [-1  2  5]
 [-1 -2 -3]]
```

```
Baza prostora vrsta matrice A (QR dekompozicija):
[[ 1  2  3 -1]
 [ 2 -1 -4  8]
 [-1  1  3 -5]]
```

**Задатак 4.** На основу неке од обрађених декомпозиција матрица одредити матрицу пројекције на простор колона матрице

$$A=\begin{pmatrix} 1 & 2 & 3 & -1 \\ 2 & -1 & -4 & 8 \\ -1 & 1 & 3 & -5 \\ -1 & 2 & 5 & -6 \\ -1 & -2 & -3 & 1 \end{pmatrix}.$$

(10 поена)

```
# LU
# Definišemo matricu A
A = np.array([[1, 2, 3, -1],
              [2, -1, -4, 8],
              [-1, 1, 3, -5],
              [-1, 2, 5, -6],
              [-1, -2, -3, 1]])

# Izračunavamo LU dekompoziciju matrice A^TA
ATA = A.T @ A
P, L, U = scipy.linalg.lu(ATA)

# Izračunavamo inverznu matricu (A^TA)^{-1}
ATA_inv = scipy.linalg.inv(U) @ scipy.linalg.inv(L)

# Izračunavamo matricu projekcije P
P = A @ ATA_inv @ A.T

print("Matrica projekcije P (LU dekompozicija):")
print(P)
```

```
Matrica projekcije P (LU dekompozicija):
[[ -6.54619565  21.9076087  -14.45380435 -18.95380435   6.54619565]
 [ 11.74456522 -35.26086957  23.50543478  31.50543478 -11.74456522]
 [ -8.23097826  25.41304348 -16.89402174 -22.51902174   8.23097826]
 [ 21.03532609 -60.42934783  40.46467391  54.46467391 -21.03532609]
 [  6.54619565 -21.9076087   14.45380435  18.95380435  -6.54619565]]
```

```
import numpy as np

# Definišemo matricu A
A = np.array([[1, 2, 3, -1],
              [2, -1, -4, 8],
              [-1, 1, 3, -5],
```

```
              [-1, 2, 5, -6],
              [-1, -2, -3, 1]])

# Izračunavamo QR dekompoziciju matrice A
Q, R = np.linalg.qr(A)

# Izračunavamo matricu projekcije P
P = Q @ Q.T

print("Matrica projekcije P (QR dekompozicija):")
print(P)
```

```
Matrica projekcije P (QR dekompozicija):
[[ 6.60268491e-01  1.98972502e-01  3.31620837e-01 -1.07552856e-16
  -2.73407342e-01]
 [ 1.98972502e-01  8.83466633e-01 -1.94222279e-01  7.63278329e-17
   1.60128047e-01]
 [ 3.31620837e-01 -1.94222279e-01  6.76296202e-01  5.55111512e-17
   2.66880078e-01]
 [-1.07552856e-16  7.63278329e-17  5.55111512e-17  1.00000000e+00
   1.11022302e-16]
 [-2.73407342e-01  1.60128047e-01  2.66880078e-01  1.11022302e-16
   7.79968674e-01]]
```

```
# Cholesky
# Definišemo matricu A
A = np.array([[1, 2, 3, -1],
              [2, -1, -4, 8],
              [-1, 1, 3, -5],
              [-1, 2, 5, -6],
              [-1, -2, -3, 1]])

# Izračunavamo Cholesky dekompoziciju matrice A^TA
ATA = A.T @ A
L = np.linalg.cholesky(ATA)

# Izračunavamo inverznu matricu (A^TA)^{-1}
ATA_inv = scipy.linalg.inv(L.T) @ scipy.linalg.inv(L)

# Izračunavamo matricu projekcije P
P = A @ ATA_inv @ A.T

print("Matrica projekcije P (Cholesky dekompozicija):")
print(P)
```

```
Matrica projekcije P (Cholesky dekompozicija):
[[-0.73913043  0.91304348 -0.69565217 -1.43478261  0.73913043]
 [ 0.93478261  0.02173913  0.17391304  1.10869565 -0.93478261]
 [-1.35869565  0.86956522 -0.79347826 -1.90217391  1.35869565]
```

```
 [ 0.5         1.         -0.5         0.5        -0.5       ]
 [ 0.73913043 -0.91304348  0.69565217  1.43478261 -0.73913043]]
```

**Задатак 5.** Показати да оротогнална матрица чува норму и углове. Другим речима, за произвољну ортогоналну матрицу $Q$ и векторе $v$ и $u$ показати једнакости: \begin{align} &|Qv|=|v|,\ &(Qv)\cdot(Qu)=v\cdot u,\ &\cos\angle(v,u)=\cos\angle(Qv,Qu). \end{align}

(10 поена)

```
import numpy as np

# Proizvoljna ortogonalna matrica Q
Q = np.array([[0.8, -0.6],
              [0.6, 0.8]])

# Proizvoljni vektori u i v
u = np.array([1, 3])
v = np.array([-2, 4])

# 1. Pokazujemo da ortogonalna matrica čuva normu
norm_u = np.linalg.norm(u)
norm_Q_u = np.linalg.norm(Q @ u)
print(f"||u|| = {norm_u}, ||Qu|| = {norm_Q_u}")

norm_v = np.linalg.norm(v)
norm_Q_v = np.linalg.norm(Q @ v)
print(f"||v|| = {norm_v}, ||Qv|| = {norm_Q_v}")

# 2. Pokazujemo da ortogonalna matrica čuva skalarni proizvod
u_dot_v = u @ v
Q_u_dot_Q_v = (Q @ u) @ (Q @ v)
print(f"u · v = {u_dot_v}, (Qu) · (Qv) = {Q_u_dot_Q_v}")

# 3. Pokazujemo da ortogonalna matrica čuva uglove
cos_angle_uv = u_dot_v / (norm_u * norm_v)
cos_angle_Q_u_Q_v = Q_u_dot_Q_v / (norm_Q_u * norm_Q_v)
print(f"cos(∠(u, v)) = {cos_angle_uv}, cos(∠(Qu, Qv)) =
{cos_angle_Q_u_Q_v}")
```

```
||u|| = 3.1622776601683795, ||Qu|| = 3.16227766016838
||v|| = 4.47213595499958, ||Qv|| = 4.47213595499958
u · v = 10, (Qu) · (Qv) = 10.0
cos(∠(u, v)) = 0.7071067811865475, cos(∠(Qu, Qv)) = 0.7071067811865474
```

Da bismo pokazali da ortogonalna matrica čuva normu i uglove, možemo koristiti svojstva ortogonalnih matrica i svojstva skalarnog proizvoda. Neka je $Q$ ortogonalna matrica i $v$ i $u$ proizvoljni vektori. Tada važi:

$QQ^T=I, Q^TQ=I$ (pošto je $Q$ ortogonalna)

Prvo, pokazaćemo da $|Qv)=|v)$:

$$|Qv)^2=(Qv)\cdot(Qv)=(Qv)^T(Qv)=v^T\left(Q^TQ\right)v=v^T(I)v=v\cdot v=|v)^2$$

Kako je $|Qv)^2=|v)^2$, sledi da je $|Qv)=|v)$.

Sada ćemo pokazati da $(Qv)\cdot(Qu)=v\cdot u$:

$$(Qv)\cdot(Qu)=(Qv)^T(Qu)=v^T\left(Q^TQ\right)u=v^T(I)u=v^Tu=v\cdot u$$

Na kraju, pokazaćemo da $\cos\angle(v,u)=\cos\angle(Qv,Qu)$:

$$\cos\angle(v,u)=\frac{v\cdot u}{|v)|u)}$$

$$\cos\angle(Qv,Qu)=\frac{(Qv)\cdot(Qu)}{|Qv)|Qu)}=\frac{v\cdot u}{|v)|u)}$$

Pošto je $\cos\angle(v,u)=\cos\angle(Qv,Qu)$, zaključujemo da ortogonalna matrica čuva normu i uglove.