Univerzitet u Sarajevu
Elektrotehnički fakultet
Predmet: <naziv_predmeta>
Studijska godina: <studijska_godina>
Student: <ime_i_prezime>
Datum: <datum>

Izvještaj za laboratorijsku vježbu 1

Za svaki zadatak je potrebno uraditi sljedeće[1]:

Zadatak <broj>

Ideja za rješenje zadatka

Objasniti vlastitim riječima u nekoliko rečenica šta je osnovna ideja za rješenje zadatka, te objasniti na koji način je zamišljeno njegovo rješenje. (Npr. Uočeno je da za regulaciju ovog sistema je potrebno koristi PID regulator. Njegovi parametri će biti određeni *Ziegler-Nichols*ovom metodom.) U slučaju da je ideja već nametnuta u postavci vježbe (Npr. Za upravljanje nekog sistema mora se koristiti PID regulator.), ovo potpoglavlje se može izostaviti.

Matematski opis zadatka

Ovdje je potrebno prikazati sva matematska izvođenja koja su neophodna za izradu zadatka. U slučaju da postoje neke slike, grafici, tabele i sl. koje olakšavaju razumijevanje tog izvođenja, potrebno ih je prikazati i referencirati se na njih. Poželjno je da sve slike i tabele budu u vektorskom formatu. (Npr. Spasiti sliku iz MATLABa u .eps ili .pdf formatu.) Na sve slike i tabele mora postojati referenca, inače će se smatrati nepostojanim. Također, svaka formula treba imati numeraciju pored, kao npr.

$$\mathscr{L}\{g(t)\} = G(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}, \quad n \geq m. \tag{1}$$

U slučaju da je matematski opis zadatka već dat u postavci vježbe, ovo potpoglavlje se može izostaviti.

Vodič kroz kôd/simulaciju

Ukoliko postoji neka funkcija koja se koristi dodatno u kôdu, pored onih koje su date u postavci vježbe, ukratko objasniti njenu ulogu, koji su joj ulazi, a koji izlazi. Također, potrebno je objasniti na koji način se pokreće kôd, koji parametri se zadaju, koji su konstantni, i sl. Ukoliko u kôdu postoje jasni komentari, onda nema potrebe pisati dodatna pojašnjenja, već je dovoljno izvršiti *copy-paste* tog dijela kôda. Treba napomenuti da ovdje nije potrebno zalijepiti cijeli kôd, već samo one dijelove koji služe za vodič. Npr. Funkcija

```
theta = dajUgao(u, A, B)
```

daje ugao `theta` u stepenima na osnovu izmjerenog napona `u`, gdje `A` i `B` predstavljaju konstante, koje zavise od korištenog senzora.

Ukoliko se koristi neka simulacija, potrebno je dati upute za njeno korištenje. Poželjno je da upute budu grafički objašnjene. Sve blokove koji nisu dati u postavci vježbe, obavezno je prikazati i detaljno objasniti.

[1] U slučaju da je nekoliko zadataka usko povezano, moguće ih je objediniti unutar jednog zadatka, pri čemu je u naslovu naglašeno koji su to zadaci, npr. Zadatak 1, 2 i 3.

Rezultati kôda/simulacije

Ovdje je potrebno prikazati rezultate testiranja/izvršavanja napisanog kôda/simulacije. Prikazati onoliko rezultata koliko je dovoljno da se shvati funkcionalnost kôda/simulacije, te potvrdi njegova/njena ispravnost. Primjeri nad kojim se vrši testiranje trebaju biti dovoljno reprezentativni. Ako se testiranje obavlja nad nekim primjerima koji nisu dati u postavci vježbe, potrebno ih je sve detaljno razmotriti. U slučaju da se dobivaju rezultati stohastičke prirode, potrebno ih je prikazati više, te komentarisati sve karakteristične veličine poput: srednje, minimalne i maksimalne vrijednosti, medijane, varijanse, prve i treće kvartile, itd.

Eksperimentalni rezultati

U slučaju da je na vježbi korišten stvarni sistem, potrebno je prikazati sve dobivene eksperimentalne rezultate, koji moraju biti potkrijepljeni slikama, graficima, tabelama i sl.

Zaključak

Ovdje je potrebno sumirati sve zaključke dobivene tokom izrade ovog zadatka.

Primjedbe i sugestije

Ovdje je moguće napisati eventualne primjedbe i poteškoće sa kojim ste se susreli prilikom izrade ove vježbe, kako bi to bilo uzeto u obzir prilikom ocjenjivanja. (Npr. Osciloskop na radnom mjestu br. 4 nije ispravno radio, pa smo izgubili vrijeme dok smo čekali kolege da oni završe kako bismo od njih posudili.) Također, moguće je navesti i sugestije za poboljšanje ove laboratorijske vježbe. U slučaju da nemate nikakvih primjedbi ili sugestija, ovo potpoglavlje se može izostaviti.

Zadatak 1

Ideja za rješenje zadatka

Za ovaj zadatak, cilj je bio prikazati funkciju $f(x) = x^3 - \cos(5x)$ na intervalima $[-1, -0.2]$, $[-0.2, 0.4]$ i $[0.4, 1]$ te pronaći maksimume i minimume funkcije unutar tih intervala. To ćemo uraditi pomoću matplotlib.pyplot biblioteke pomoću koje crtamo grafik funkcije. Funkciju ćemo definisati u posebnoj funkciji, a maximum i minimum tražimo pomoću argmax i argmin funkcija unutar numpy biblioteke.

Matematski opis zadatka

Funkcija koja se analizira je $f(x) = x^3 - \cos(5x)$. Njena vrijednost je iscrtana na intervalima $[-1, -0.2]$, $[-0.2, 0.4]$ i $[0.4, 1]$. Pomoću NumPy biblioteke, izračunavaju se vrijednosti funkcije za svaki od ovih intervala. Zatim se koristi funkcija 'argmin' i 'argmax' kako bi se pronašli minimumi i maksimumi funkcije unutar svakog intervala.

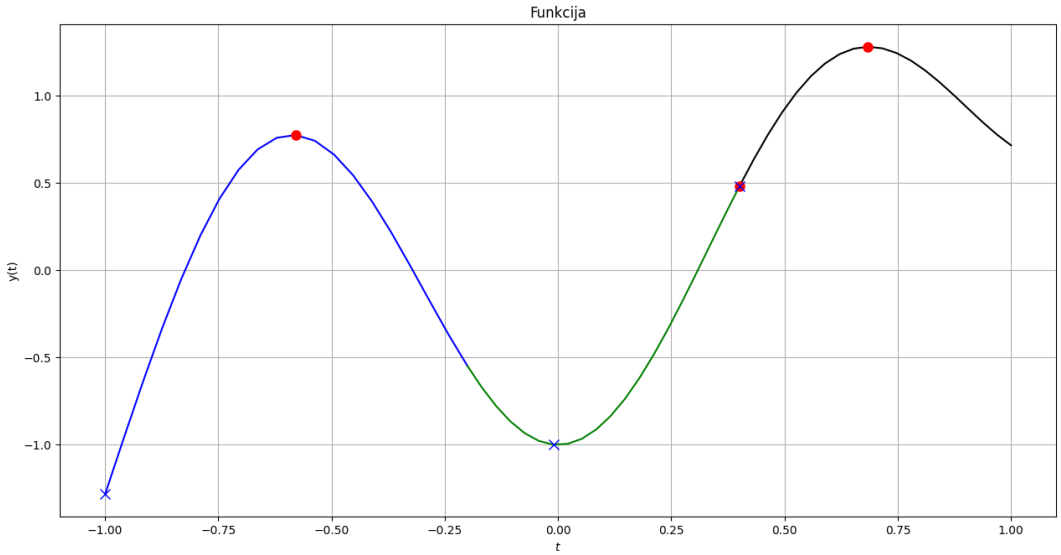
Vodič kroz kod

```
import math
import numpy as np
import matplotlib.pyplot as plt

def f(x):
```

```
    return x**3 - np.cos(5*x)

l1 = np.linspace(-1, -0.2, 20)
l2 = np.linspace(-0.2, 0.4, 20)
l3 = np.linspace(0.4, 1, 20)

y1=f(l1)
y2=f(l2)
y3=f(l3)

plt.plot(l1, y1, color='blue')
plt.plot(l2, y2, color='green')
plt.plot(l3, y3, color='black')

segments = [(l1, y1), (l2, y2), (l3, y3)]
markers = ['x', 'o']

for segment in segments:
    x_values, y_values = segment
    min_index = np.argmin(y_values)
    max_index = np.argmax(y_values)
    min_x, min_y = x_values[min_index], y_values[min_index]
    max_x, max_y = x_values[max_index], y_values[max_index]

    plt.plot(min_x, min_y, color='blue', marker='x', markersize=8)
    plt.plot(max_x, max_y, color='red', marker='o', markersize=8)

plt.xlabel('$t$')
plt.ylabel('y(t)')
plt.title('Funkcija')
plt.grid(True)
plt.show()
```

Python kod koristi NumPy i Matplotlib biblioteke. Koristi se funkcija 'linspace' za generisanje jednakih razmaka između početnog i završnog broja u određenom intervalu. Nakon toga, izračunavaju se vrijednosti funkcije za svaki interval pomoću definisane funkcije $f(x)$ koja vraća vrijednost funkcije za dato x. Za pronalaženje minimuma i maksimuma funkcije korištena je funkcija 'argmin' i 'argmax'. Nakon toga je korištena funkcija plot za crtanje grafika te su ispisane labele za x,y i za naziv.

Rezultati koda



Slika 1: Zadatak 1

Zaključak

Analizom rezultata možemo primijetiti da funkcija $f(x) = x^3 - \cos(5x)$ ima maksimume i minimume unutar zadatih intervala.

Zadatak 2

Ideja za rješenje zadatka

U ovom zadatku cilj je prikazati površinu funkcije $f(x_1, x_2) = 4x_1^2 + x_2^2 + 16x_1^2x_2^2$ u 3D prostoru. Također, potrebno je nacrtati konture funkcije, kao i konture parcijalnih izvoda po varijablama x_1 i x_2.

Matematski opis zadatka

Funkcija koja se analizira je $f(x_1, x_2) = 4x_1^2 + x_2^2 + 16x_1^2x_2^2$. Pomoću NumPy biblioteke, izračunavaju se vrijednosti funkcije za odabrane intervale x_1 i x_2. Zatim se koristi funkcija 'meshgrid' za generisanje 3D prostora, na kojem se prikazuje površina funkcije. Također, koriste se funkcije 'contour' za prikaz kontura funkcije, te kontura parcijalnih izvoda po varijablama x_1 i x_2.

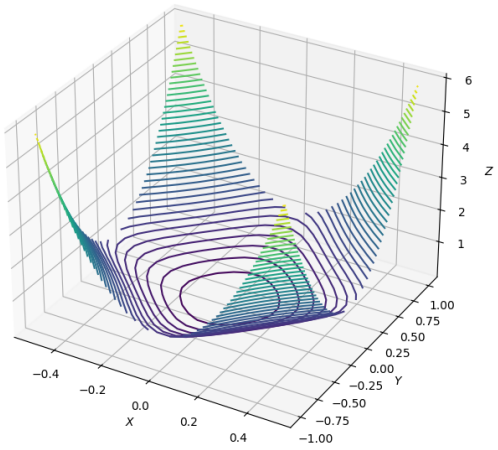
Vodič kroz kod

```
import math
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

def f(x1, x2):
```

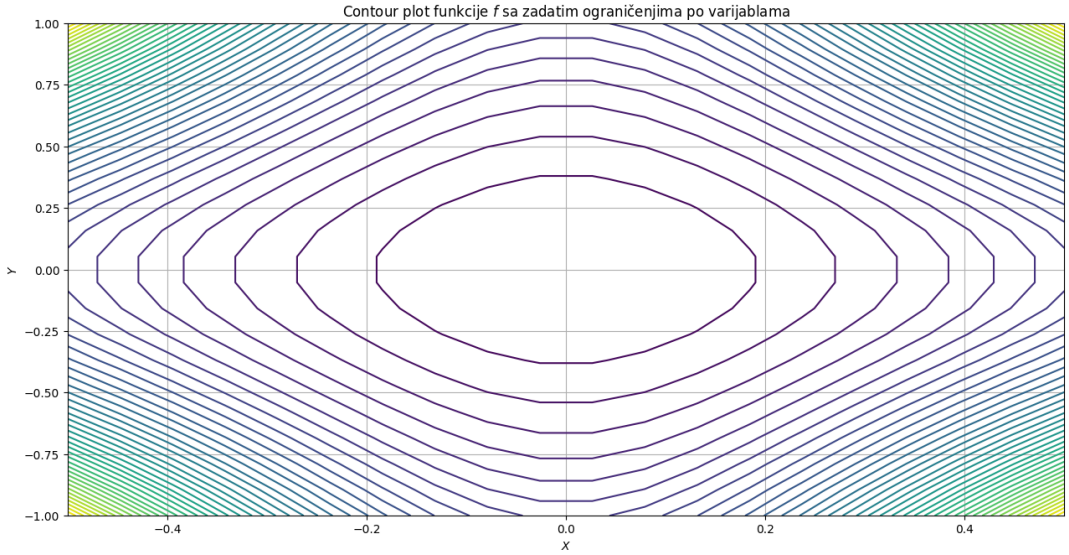
```
    return 4*(x1**2) + (x2**2) + 16*(x1**2)*(x2**2)

x1=np.linspace(-0.5, 0.5, 20)
x2=np.linspace(-1, 1, 20)

X, Y = np.meshgrid(x1, x2)
Z = f(X,Y)

# (a)
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50)
ax.set_xlabel('$X$')
ax.set_ylabel('$Y$')
ax.set_zlabel('$Z$')

# (b)
plt.figure()
plt.contour(X, Y, Z, levels=50)
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.title('Contour plot funkcije $f$ sa zadatim ograni enjima po varijablama')
plt.grid(True)

# (c)
plt.figure()
# Parcijalni izvodi po x1
plt.contour(X, Y, 8*X + 32*X*Y**2, levels=50, colors='r', linestyles='dashed')
plt.xlabel('$X_1$')
plt.ylabel('$X_2$')
plt.title('Contour plot parcijalnog izvoda po $X$')
plt.grid(True)

plt.figure()
# Parcijalni izvodi po x2
plt.contour(X, Y, 2*Y + 32*X**2*Y, levels=50, colors='g', linestyles='dashed')
plt.xlabel('$X_1$')
plt.ylabel('$X_2$')
plt.title('Contour plot parcijalnog izvoda po $Y$')
plt.grid(True)

plt.show()
```

Python kod koristi NumPy i Matplotlib biblioteke. Koristi se funkcija ‘meshgrid‘ za generisanje 3D prostora, a zatim se prikazuje površina funkcije. Nakon toga, koriste se funkcije ‘contour‘ za prikaz kontura funkcije, te kontura parcijalnih izvoda po varijablama x_1 i x_2.

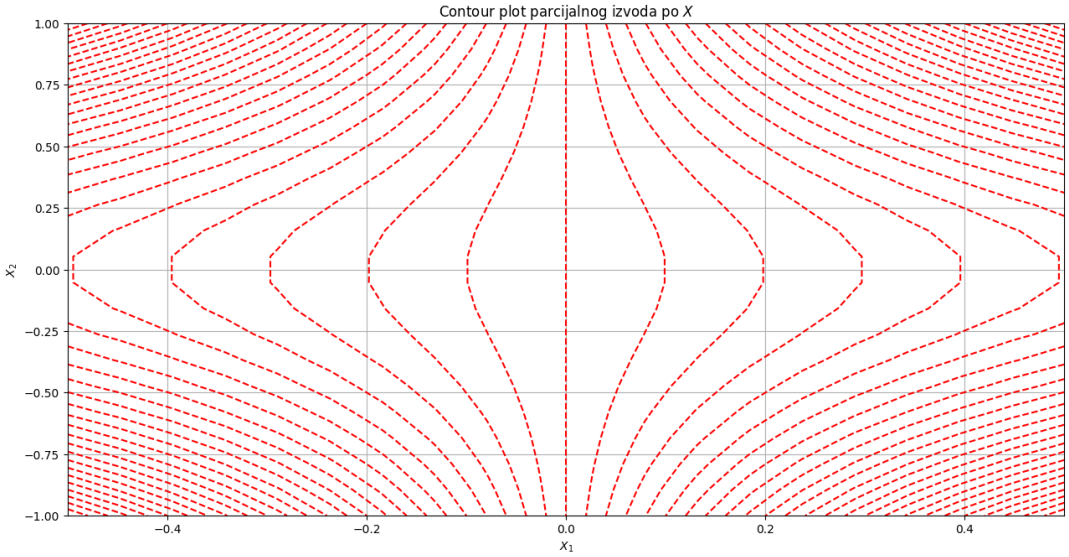
Rezultati koda



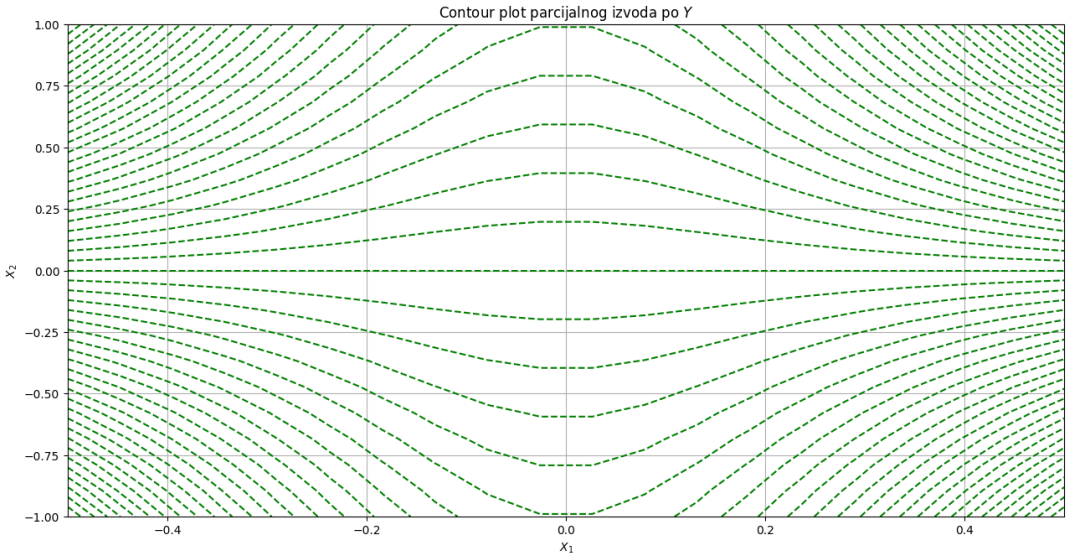
Slika 2: Površina funkcije $f(x_1, x_2) = 4x_1^2 + x_2^2 + 16x_1^2x_2^2$ u 3D prostoru



Slika 3: Contour plot funkcije f sa zadatim ograničenjima po varijablama



Slika 4: Contour plot parcijalnog izvoda po x_1



Slika 5: Contour plot parcijalnog izvoda po x_2

Zaključak

Analizom rezultata možemo zaključiti da funkcija $f(x_1, x_2) = 4x_1^2 + x_2^2 + 16x_1^2x_2^2$ ima karakterističnu površinu u 3D prostoru, kao i odgovarajuće konture i parcijalne izvode po varijablama.

Zadatak 3

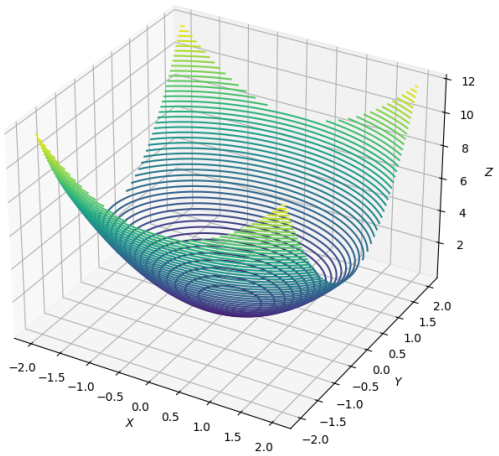
(Radi mi samo pod a)

Vodič kroz kod

```
import math
import numpy as np
import matplotlib.pyplot as plt
from  mpl_toolkits import  mplot3d

def f(x1, x2):
    return (x1**2) + 2*(x2**2)

x1=np.linspace(-2, 2, 50)
x2=np.linspace(-2, 2, 50)

X1, X2 = np.meshgrid(x1, x2)
Z = f(X1,X2)

# (a)
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(X1, X2, Z, 50)
ax.set_xlabel('$X$')
ax.set_ylabel('$Y$')
ax.set_zlabel('$Z$')

# (b)
x1_region = np.linspace(-2, 2, 100)
x2_region = np.linspace(-2, 2, 100)
X1_region, X2_region = np.meshgrid(x1_region, x2_region)
region_mask = (X1_region**2 + X2_region**2 <= 2) & (X1_region - X2_region <= 1) &

# (b) Contour plot funkcije f preklopljen sa region plotom sa ograni enjima
plt.figure()
# Contour plot funkcije f
plt.contour(X1, X2, Z, levels=50)
# Region plot ograni enja
plt.contourf(region_mask, colors=['green'], alpha=0.5)
#plt.plot(x1_region, x1_region - 1, 'r--')
plt.xlabel('$X_1$')
plt.ylabel('$X_2$')
plt.title('Contour␣plot␣funkcije␣$f(x_1,␣x_2)$␣sa␣region␣plotom␣ograni enja')
plt.grid(True)

# (c) Contour plotovi prvih parcijalnih izvoda funkcije f po svim njenim varijablam
plt.figure()
# Prvi parcijalni izvod po x1
Z_partial_x1 = 2 * X1
plt.contour(X1, X2, Z_partial_x1, levels=50, colors='r')
#plt.contourf(region_mask, colors=['green'], alpha=0.5)

plt.xlabel('$X_1$')
plt.ylabel('$X_2$')
plt.title('Contour␣plot␣parcijalnog␣izvoda␣po␣$X_1$␣sa␣ograni enjima')
plt.grid(True)
plt.show()
```

```
plt.figure()
# Prvi parcijalni izvod po x2
Z_partial_x2 = 4*X2**2
plt.contour(X1, X2, Z_partial_x2, levels=50, colors='g')
#plt.contourf(region_mask, colors=['green'], alpha=0.5)

plt.xlabel('$X_1$')
plt.ylabel('$X_2$')
plt.title('Contour plot parcijalnog izvoda po $X_2$ sa ograni enjima')
plt.grid(True)
plt.show()
```

Rezultati koda



Slika 6: Trajektorije stanja

Zadatak 4

Ideja za rješenje zadatka

U ovom zadatku implementirana je funkcija 'LTI' koja simulira trajektoriju stanja i izlaza diskretnog sistema zapisanog u prostoru stanja. Ova funkcija uzima matrice E, F, C, D, početno stanje x0 i ulaz U kako bi izračunala i vratila spojene vektore stanja X i izlaza Y.

Matematski opis zadatka

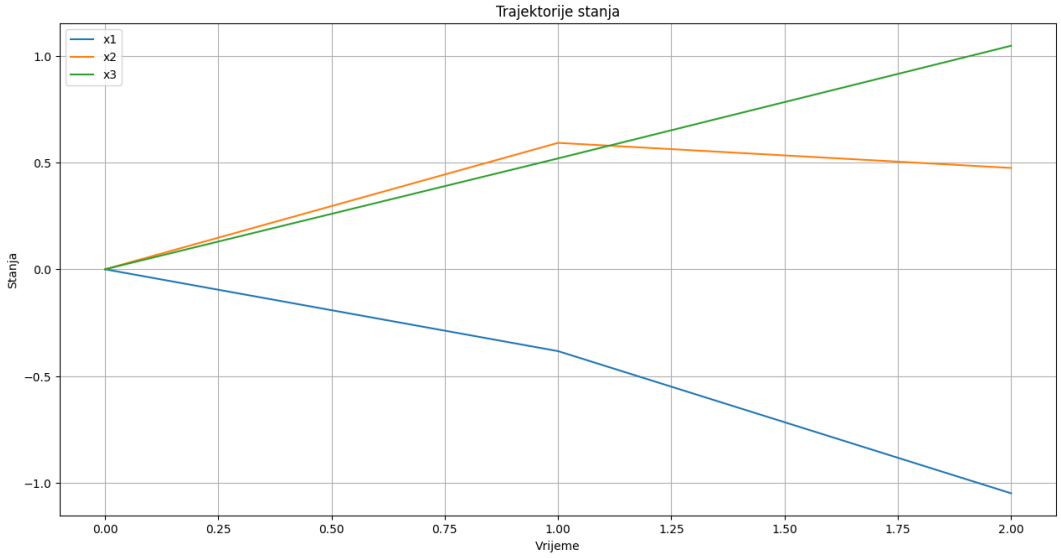
Funkcija 'LTI' simulira trajektoriju stanja i izlaza diskretnog sistema. Matrice E, F, C, i D su dijelovi opisa sistema, dok su x0 i U početno stanje i ulaz sistema. Nakon provjere dimenzija matrica i vektora, inicijaliziraju se spremnici za stanja i izlaze. Zatim se koriste iteracije kako bi se izračunala trajektorija stanja i izlaza koristeći matematičke relacije definirane u postavci zadatka.

Vodič kroz kod

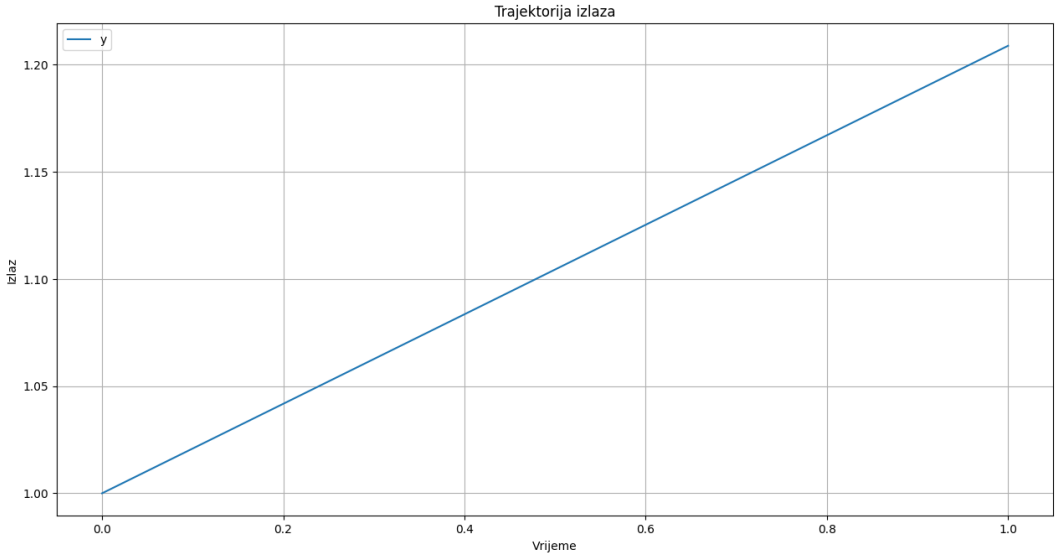
```
import numpy as np
import matplotlib.pyplot as plt

def LTI(E, F, C, D, x0, U):
    # Provjera dimenzija matrica i vektora
    n, m = F.shape
    r, _ = C.shape

    assert E.shape == (n, n), "Dimenzije matrice E nisu ispravne."
    assert D.shape == (r, m), "Dimenzije matrice D nisu ispravne."
    assert x0.shape == (n, 1), "Dimenzije vektora x0 nisu ispravne."
    assert U.shape[1] == m, "Dimenzije vektora U nisu ispravne."

    # Inicijalizacija spremnika za stanja i izlaze
    N = U.shape[0]
    X = np.zeros((N+1, n))
    Y = np.zeros((N, r))

    # Postavljanje pocetnog stanja
    X[0] = x0.flatten()

    # Izracunavanje trajektorije stanja i izlaza
    for k in range(N):
        X[k+1] = np.dot(E, X[k]) + np.dot(F, U[k])
        Y[k] = np.dot(C, X[k]) + np.dot(D, U[k])

    return X, Y

# Testiranje funkcije
E = np.array([[1.1269, -0.4940, 0.1129],
              [1, 0, 0],
              [0, 1, 0]])

F = np.array([[-0.3832, -0.3832],
              [0.5919, 0.8577],
              [0.5191, 0.4546]])

C = np.array([[1, 1, 0]])

D = np.array([[1, 1]])

x0 = np.array([[0],
               [0],
               [0]])

U = np.array([[1, 0],
              [0, 1]])

X, Y = LTI(E, F, C, D, x0, U)

# Iscrtavanje trajektorija stanja i izlaza
plt.plot(X[:, 0], label='x1')
plt.plot(X[:, 1], label='x2')
```

```
plt.plot(X[:, 2], label='x3')
plt.xlabel('Vrijeme')
plt.ylabel('Stanja')
plt.title('Trajektorije stanja')
plt.grid(True)
plt.legend()
plt.show()

plt.plot(Y[:, 0], label='y')
plt.xlabel('Vrijeme')
plt.ylabel('Izlaz')
plt.title('Trajektorija izlaza')
plt.grid(True)
plt.legend()
plt.show()
```

Rezultati koda



Slika 7: Trajektorije stanja



Slika 8: Trajektorija izlaza

Zaključak

Simulacija trajektorije stanja i izlaza diskretnog sistema uspješno je provedena. Grafici prikazuju kako se stanja i izlazi mijenjaju s vremenom, što je karakteristično za dinamičke sisteme.