UNIVERZITET U BEOGRADU

ELEKTROTEHNIČKI FAKULTET

ŠTURMOVA TEOREMA

Drugi projektni zadatak

Mentor:

Branko Malešević, prof. dr

Kandidat:

Filip Kojić 2023/3297

Beograd, Januar 2024.

SADRŽAJ

1. POSTAVKA PROJEKTNOG ZADATKA

Neka je dat realan polinom $P(x)$ nad realnim segmentom $[a,b]$.

(i) 1. Odrediti Euklidovim algoritmom najveći zajednički delilac
$G(x) = GCD(P(x), P'(x))$

2. Za polinom $P(x) := P(x)/G(x)$, upotrebom Šturmove teoreme, odrediti broj nula na $[a,b]$.

(ii) Primena Šturmove teoreme.

Neka je k broj decimala na koji se zaokružuje neki realan broj. Za polinom:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Sa realnim koeficijentima

$$a_n, a_{n-1}, \cdots, a_1, a_0$$

Odrediti niz naniže zaokruženih racionalnih koeficijenata

$$\alpha_n, \alpha_{n-1}, \cdots, \alpha_1, \alpha_0$$

Odrediti po sledećim pravilima:

- Ako je $a_k = a_0.\, a_1 a_2 \cdots a_k a_{k+1} \cdots > 0 \; tada \; \alpha_k = a_0.\, a_1 \cdots a_k$;
- Ako je $a_k = a_0.\, a_1 a_2 \cdots a_k a_{k+1} \cdots < 0 \; tada \; \alpha_k = a_0.\, a_1 \cdots a'_k$, gde je a'_k naviše zaokružena cifra (uz eventualno posledično zaokruživanje i prethodnih cifara za jedan broj naviše).

Odrediti proceduru za formiranje racionalnog polinoma

$$P(x) = \alpha_n x^n + \alpha_{n-1} x^{n-1} + \cdots + \alpha_1 x + \alpha_0$$

Neka je $[a, b]$ segment sa racionalnim rubnim tačkama. Naći takav polinom $P(x)$ sa realnim koeficijentima i broj k, da na osnovu pozitivnosti polinoma P(x) nad segmentom $[a,b]$ imamo dokaz o pozitivnosti polinoma $P(x)$ nad segmentom $[a,b]$.

Teorijska osnova za izradu ovog projektnog zadatka data je u materijalima sa predavanja profersora Maleševića. [1]

2. PREGLED REŠENJA PROJEKTNOG ZADATKA

Programski jezik u kome je rađena implementacija ovog projektnog zadatka je Python (Python 3.9). Implementacija se sastoji iz 2 fajla, projekat2_1.py i projekat2_2.py, čije ćemo detalje ukratko izložiti u nastavku. projekat2_1.py predstavlja implementaciju Šturmovog algoritma i određivanja broja nula na zadatom segmentu, a projekat2_2.py predstavlja proceduru za dobijanje racionalnog polinoma. U izradi projekta su korišćene biblioteke sympy i math.

2.1. Prvi deo projektog zadatka

projekat2_1.py

```python
import sympy as s

# ispis polinoma
def ispisiPolinom(polinom):
    return str(polinom).replace('x**', 'x^')

# trazenje prvog izvoda polinoma
def nadjiPrviIzvod(polinom):
    return s.diff(polinom, x, 1)

# trazenje najveceg zajednickog delioca polinoma
def izracunaj_GCD(polinom1, polinom2):
    return s.gcd(polinom1, polinom2)

# implementacija sturmovog algoritma
def sturmova_teorema(polinom1, polinom2):
    sturm_niz = []
    pol0, rem0 = s.div(polinom1, polinom2)
    pol1 = s.diff(pol0, x, 1)
    sturm_niz.append(pol0)
    sturm_niz.append(pol1)
    while s.degree(pol0) > 1:
        pol, rem = s.div(pol0, pol1)
        sturm_niz.append(-rem)
        pol0 = pol1
        pol1 = -rem
    return sturm_niz

# uvrstavanje granica intervala [a, b] u sturmove polinome
def izracunajVrednostiPolinoma(a, b):
```

projekat2_1.py – Deo 1

```python
# uvrstavanje granica intervala [a, b] u sturmove polinome
def izracunajVrednostiPolinoma(a, b):
    rezultatiA = []
    rezultatiB = []
    for polinom in sturm_niz:
        vrednost1 = polinom.subs(x, a)
        vrednost2 = polinom.subs(x, b)
        rezultatiA.append(vrednost1)
        rezultatiB.append(vrednost2)
    return rezultatiA, rezultatiB

# ispis vrednosti sturmovih polinoma sa a i b granicama intervala
def ispisVrednostiSturmovihPolinoma(rezultatiA, rezultatiB):
    for i in range(len(rezultatiA)):
        val1 = rezultatiA[i]
        val2 = rezultatiB[i]
        print("P" + str(i) + "(" + str(a) + ") = " + str(val1) + "  P" + str(i) + "(" + str(b) + ") = " + str(val2))

# prebrojavanje promena znaka za nizove vrednosti sa a i b granicama intervala
def prebrojPromeneZnaka(rezultatiA, rezultatiB):
    promenaZnakaA = 0
    promenaZnakaB = 0
    duzina = len(rezultatiA)
    for i in range(0, duzina - 1):
        if (rezultatiA[i] < 0 and rezultatiA[i + 1] >= 0) or (rezultatiA[i] >= 0 and rezultatiA[i + 1] < 0):
            promenaZnakaA += 1
        if (rezultatiB[i] < 0 and rezultatiB[i + 1] >= 0) or (rezultatiB[i] >= 0 and rezultatiB[i + 1] < 0):
            promenaZnakaB += 1
    return promenaZnakaA, promenaZnakaB
```

projekat2_1.py – Deo 2

```python
x = s.symbols('x')

# polinom za testiranje
polinom = x**9 - 3*x**7 - x**6 + 3*x**5 + 3*x**4 - x**3 - 3*x**2 + 1

# granice intervala [a, b]
a = 0 ; b = 3;

prviIzvod = nadjiPrviIzvod(polinom)

print("P0(x) = " + ispisiPolinom(polinom))
print("P1(x) = " + ispisiPolinom(prviIzvod))

gcd = izracunaj_GCD(polinom, prviIzvod)
print("GCD(P0(x), P1(x)) = " + ispisiPolinom(gcd))

sturm_niz = []
sturm_niz = sturmova_teorema(polinom, gcd)
print("**********************************************************************")
print("Šturmov niz polinoma:")
print("**********************************************************************")
for i in range(len(sturm_niz)):
    print("P" + str(i) + "(x) = " + ispisiPolinom(sturm_niz[i]))

rezultatiA, rezultatiB = izracunajVrednostiPolinoma(a, b)
print("**********************************************************************")
print("Vrednosti Šturmovih polinoma:")
print("**********************************************************************")
ispisVrednostiSturmovihPolinoma(rezultatiA, rezultatiB)
```

projekat2_1.py – Deo 3

```python
gcd = izracunaj_GCD(polinom, prviIzvod)
print("GCD(P0(x), P1(x)) = " + ispisiPolinom(gcd))

sturm_niz = []
sturm_niz = sturmova_teorema(polinom, gcd)
print("*******************************************************************")
print("Šturmov niz polinoma:")
print("*******************************************************************")
for i in range(len(sturm_niz)):
    print("P" + str(i) + "(x) = " + ispisiPolinom(sturm_niz[i]))

rezultatiA, rezultatiB = izracunajVrednostiPolinoma(a, b)
print("*******************************************************************")
print("Vrednosti Šturmovih polinoma:")
print("*******************************************************************")
ispisVrednostiSturmovihPolinoma(rezultatiA, rezultatiB)

promenaZnakaA, promenaZnakaB = prebrojPromeneZnaka(rezultatiA, rezultatiB)
print("*******************************************************************")
print("Broj promena znaka za granicu a: V(" + str(a) + ") = " + str(promenaZnakaA))
print("Broj promena znaka za granicu b: V(" + str(b) + ") = " + str(promenaZnakaB))
print("*******************************************************************")
brojNula = abs(promenaZnakaA - promenaZnakaB)
print("Broj nula polinoma P(x) = {}, na intervalu [a, b] = [{}, {}] je: {}".format(ispisiPolinom(polinom), a, b, brojNula))
```

projekat2_1.py – Deo 4

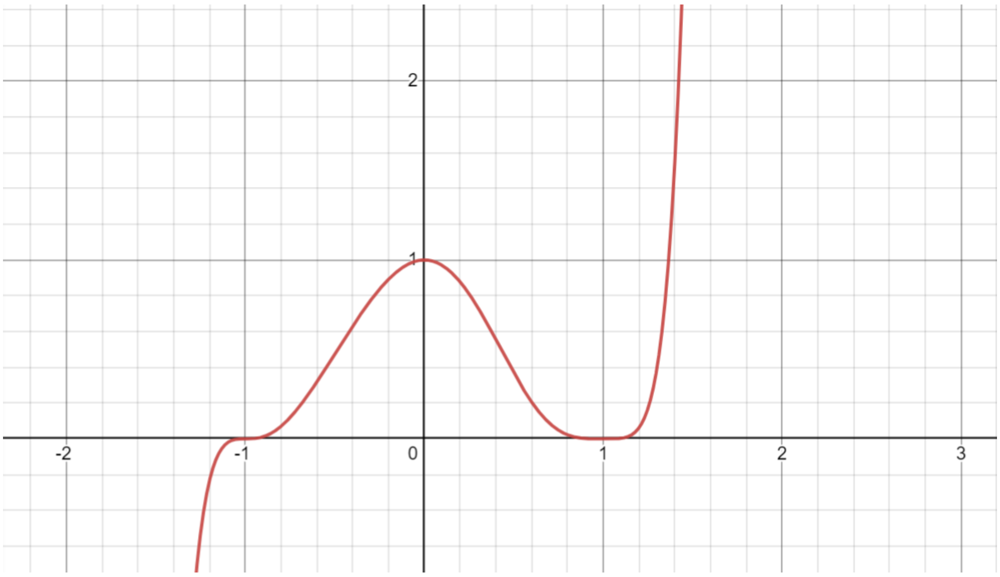
2.2. Testiranje rešenja prvog dela projektog zadatka

U nastavku će biti izloženo testiranje rešenja prvog dela projektnog zadatka sa tri različita seta ulaznih podataka.

1. Test primer 1(primer sa predavanja):

```
C:\Users\Korisnik\Desktop\OPNA-DZ2\venv\Scripts\python.exe C:/Users/Korisnik/Desktop/OPNA-DZ2/projekat2_1.py
P0(x) = x^9 - 3*x^7 - x^6 + 3*x^5 + 3*x^4 - x^3 - 3*x^2 + 1
P1(x) = 9*x^8 - 21*x^6 - 6*x^5 + 15*x^4 + 12*x^3 - 3*x^2 - 6*x
GCD(P0(x), P1(x)) = x^5 - x^4 - 2*x^3 + 2*x^2 + x - 1
*******************************************************************
Šturmov niz polinoma:
*******************************************************************
P0(x) = x^4 + x^3 - x - 1
P1(x) = 4*x^3 + 3*x^2 - 1
P2(x) = 3*x^2/16 + 3*x/4 + 15/16
P3(x) = -32*x - 64
P4(x) = -3/16
*******************************************************************
Vrednosti Šturmovih polinoma:
*******************************************************************
P0(0) = -1  P0(3) = 104
P1(0) = -1  P1(3) = 134
P2(0) = 15/16  P2(3) = 39/8
P3(0) = -64  P3(3) = -160
P4(0) = -3/16  P4(3) = -3/16
*******************************************************************
Broj promena znaka za granicu a: V(0) = 2
Broj promena znaka za granicu b: V(3) = 1
*******************************************************************
Broj nula polinoma P(x) = x^9 - 3*x^7 - x^6 + 3*x^5 + 3*x^4 - x^3 - 3*x^2 + 1, na intervalu [a, b] = [0, 3] je: 1

Process finished with exit code 0
```

Test primer 1 – ispis rada programa



Test primer 1 – grafik polinomske funkcije

Na osnovu algoritamske analize, utvrđeno je da polinom iz našeg prvog test primera ima tačno jedan koren na zadatom intervalu [0, 3]. Ovaj nalaz je potvrđen i vizuelnom analizom koristeći Desmos grafikon, gde je jasno prikazana tačno jedna nula polinoma na istom intervalu.

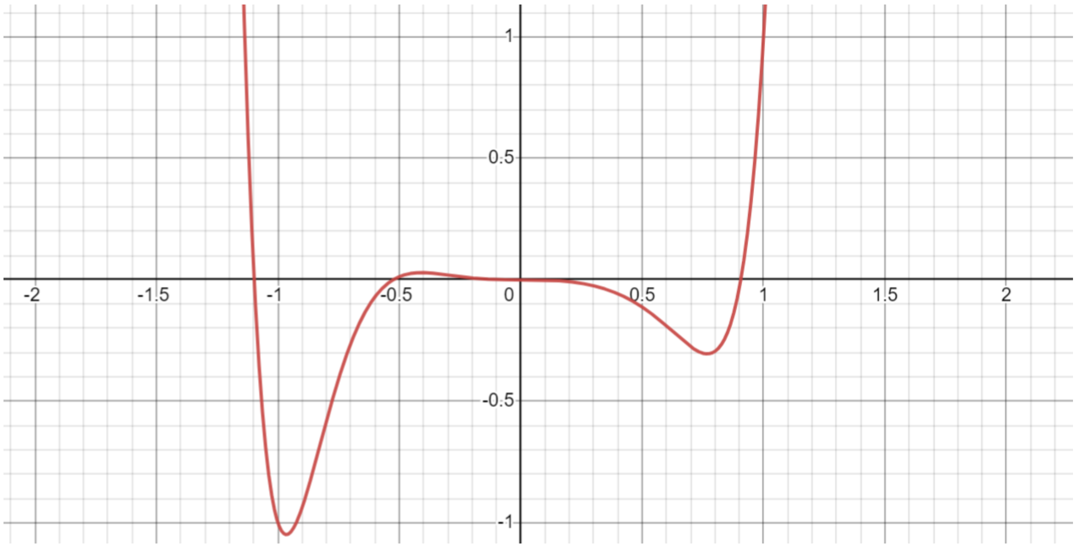
2. Test primer 2:

```
# polinom za testiranje
polinom = x**10 + 3*x**8 - 4*x**6 + 2*x**5 - x**3

# granice intervala [a, b]
a = -2 ; b = 2;
```

Test primer 2 – ulazni podaci

```
C:\Users\Korisnik\Desktop\OPNA-DZ2\venv\Scripts\python.exe C:/Users/Korisnik/Desktop/OPNA-DZ2/projekat2_1.py
P0(x) = x^10 + 3*x^8 - 4*x^6 + 2*x^5 - x^3
P1(x) = 10*x^9 + 24*x^7 - 24*x^5 + 10*x^4 - 3*x^2
GCD(P0(x), P1(x)) = x^2
*********************************************************************
Šturmov niz polinoma:
*********************************************************************
P0(x) = x^8 + 3*x^6 - 4*x^4 + 2*x^3 - x
P1(x) = 8*x^7 + 18*x^5 - 16*x^3 + 6*x^2 - 1
P2(x) = -3*x^6/4 + 2*x^4 - 5*x^3/4 + 7*x/8
P3(x) = -118*x^5/3 + 40*x^4/3 + 16*x^3 - 46*x^2/3 + 1
P4(x) = -5600*x^4/3481 + 7387*x^3/6962 - 345*x^2/3481 - 101*x/118 + 45/6962
P5(x) = -633900543*x^3/62720000 - 39958399*x^2/6272000 - 410858949*x/62720000 - 11908501/12544000
P6(x) = 42074708480000*x^2/115435190581929 + 26291032320000*x/12826132286881 + 21754149760000/115435190581929
P7(x) = 4024132732383387435873*x/14112572494236160000 + 1526585722027255818477/5645028997694464000
P8(x) = 37459793327785856581456000/140283428010162175893585843801
*********************************************************************
Vrednosti Šturmovih polinoma:
*********************************************************************
P0(-2) = 370  P0(2) = 398
P1(-2) = -1449  P1(2) = 1495
P2(-2) = -31/4  P2(2) = -97/4
P3(-2) = 3851/3  P3(2) = -2933/3
P4(-2) = -229093/6962  P4(2) = -134737/6962
P5(-2) = 4235043777/62720000  P5(2) = -7550800707/62720000
P6(-2) = -94395199360000/38478396860643  P6(2) = 221097188480000/38478396860643
P7(-2) = -3066647613703984366850 7/5645028997694464000  P7(2) = 3371964758109435530546 1/5645028997694464000
P8(-2) = 37459793327785856581456000/140283428010162175893585843801  P8(2) = 37459793327785856581456000/140283428010162175893585843801
*********************************************************************
Broj promena znaka za granicu a: V(-2) = 6
Broj promena znaka za granicu b: V(2) = 2
*********************************************************************
Broj nula polinoma P(x) = x^10 + 3*x^8 - 4*x^6 + 2*x^5 - x^3, na intervalu [a, b] = [-2, 2] je: 4

Process finished with exit code 0
```

Test primer 2 – ispis rada programa

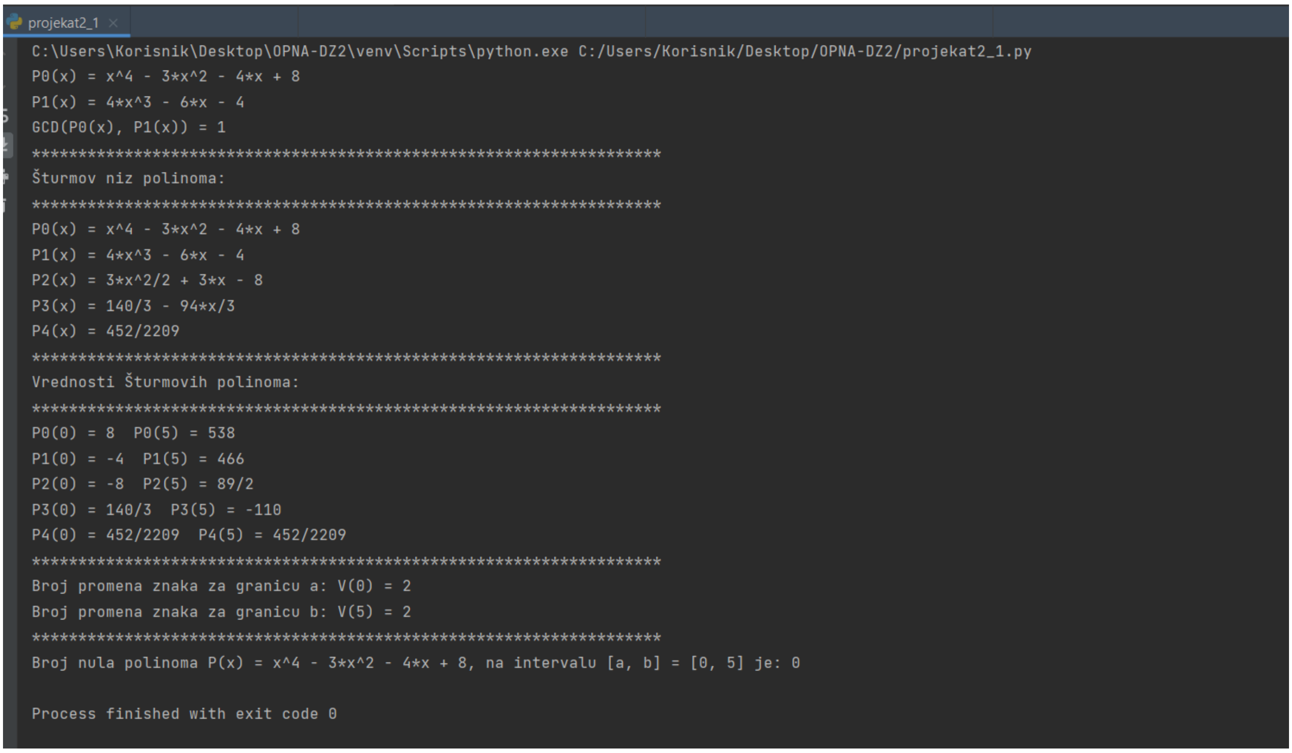


Test primer 2 – grafik polinomske funkcije

3. Test primer 3:

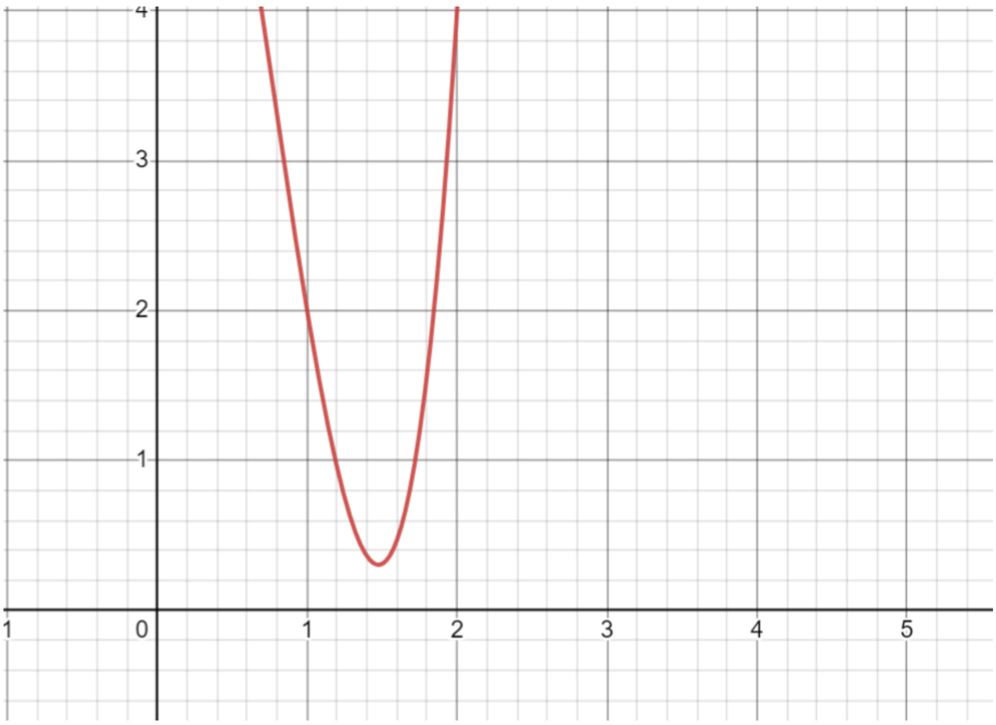
```python
# polinom za testiranje
polinom = x**4 - 3*x**2 - 4*x + 8

# granice intervala [a, b]
a = 0 ; b = 5;
```

Test primer 3 – ulazni podaci



```
projekat2_1
C:\Users\Korisnik\Desktop\OPNA-DZ2\venv\Scripts\python.exe C:/Users/Korisnik/Desktop/OPNA-DZ2/projekat2_1.py
P0(x) = x^4 - 3*x^2 - 4*x + 8
P1(x) = 4*x^3 - 6*x - 4
GCD(P0(x), P1(x)) = 1
**********************************************************************
Šturmov niz polinoma:
**********************************************************************
P0(x) = x^4 - 3*x^2 - 4*x + 8
P1(x) = 4*x^3 - 6*x - 4
P2(x) = 3*x^2/2 + 3*x - 8
P3(x) = 140/3 - 94*x/3
P4(x) = 452/2209
**********************************************************************
Vrednosti Šturmovih polinoma:
**********************************************************************
P0(0) = 8  P0(5) = 538
P1(0) = -4  P1(5) = 466
P2(0) = -8  P2(5) = 89/2
P3(0) = 140/3  P3(5) = -110
P4(0) = 452/2209  P4(5) = 452/2209
**********************************************************************
Broj promena znaka za granicu a: V(0) = 2
Broj promena znaka za granicu b: V(5) = 2
**********************************************************************
Broj nula polinoma P(x) = x^4 - 3*x^2 - 4*x + 8, na intervalu [a, b] = [0, 5] je: 0

Process finished with exit code 0
```

Test primer 3 – ispis rada programa



Test primer 3 – grafik polinomske funkcije

2.3. Drugi deo projektnog zadatka

Koeficijenti polinoma su dati kao niz, a postupak obrade uključuje formiranje dva nova niza: jedan sa skraćenim koeficijentima - prema specifičnim pravilima zaokruživanja na određeni broj decimala - i drugi sa stepenima broja 10, koji odgovaraju redu veličine svakog koeficijenta. Rezultat ovog procesa je polinom sa racionalnim koeficijentima, koji se zatim može koristiti u prvom delu projekta za određivanje broja nula polinoma na određenom intervalu, kao i za proveru njegove pozitivnosti na tom intervalu.

projekat2_2.py

```python
import math as mat

# svodi koeficijente na oblik a0.a1a2a3... * 10^stepen
def urediKoeficijenteStepene(polinom):
    koeficijenti = []
    stepeni = []
    for i in range(len(polinom)):
        stepen = 0
        if polinom[i] == 0:
            koeficijenti.append(polinom[i])
            stepeni.append(0)
        else:
            koef = abs(polinom[i])
            while koef < 1:
                stepen -= 1
                koef *= 10
            if polinom[i] < 0:
                koeficijenti.append(-koef)
            else:
                koeficijenti.append(koef)
            stepeni.append(stepen)
    return koeficijenti, stepeni

# svodi koeficijente na oblik a0.a1a2 * 10^stepen
# vrseci skracivanje na k decimala prema pravilima
def skratiKoeficijente(koeficijenti):
    skraceni_koeficijenti = []
    for koeficijent in koeficijenti:
        koef_str = str(koeficijent)
        if koeficijent < 0:
            koef_str_skracen = koef_str[:5]
```

projekat2_2.py – Deo 1

```python
# svodi koeficijente na oblik a0.a1a2 * 10^stepen
# vrseci skracivanje na k decimala prema pravilima
def skratiKoeficijente(koeficijenti):
    skraceni_koeficijenti = []
    for koeficijent in koeficijenti:
        koef_str = str(koeficijent)
        if koeficijent < 0:
            koef_str_skracen = koef_str[:5]
            koef_abs = abs(float(koef_str_skracen))
            koef_abs += 0.01
            skraceni_koeficijenti.append(round(float(koef_abs * -1), k))
        else:
            koef_str_skracen = koef_str[:4]
            skraceni_koeficijenti.append(float(koef_str_skracen))
    return skraceni_koeficijenti

# ispisuje polinom u obliku koef[i]*10^stepeni[i]*x^stepen + ...
def ispisiPolinom(koeficijenti, skraceni_koeficijenti, stepeni):
    ispis = ""
    brojStepeni = len(koeficijenti) - 1
    for i in range(len(koeficijenti)):
        if skraceni_koeficijenti[i] == 0:
            continue  # preskače ispisivanje ako je koeficijent 0
        stepen = brojStepeni - i
        if ispis != "" and skraceni_koeficijenti[i] > 0:
            ispis += "+ "  # dodaje plus znak pre pozitivnih koeficijenata
        # ispisuje koeficijent
        ispis += str(skraceni_koeficijenti[i])
        # dodaje faktor 10^stepen ako stepen nije 0
        if stepeni[i] != 0:
```

projekat2_2.py – Deo 2

```python
# ispisuje polinom u obliku koef[i]*10^stepeni[i]*x^stepen + ...
def ispisiPolinom(koeficijenti, skraceni_koeficijenti, stepeni):
    ispis = ""
    brojStepeni = len(koeficijenti) - 1
    for i in range(len(koeficijenti)):
        if skraceni_koeficijenti[i] == 0:
            continue  # preskače ispisivanje ako je koeficijent 0
        stepen = brojStepeni - i
        if ispis != "" and skraceni_koeficijenti[i] > 0:
            ispis += "+ "  # dodaje plus znak pre pozitivnih koeficijenata
        # ispisuje koeficijent
        ispis += str(skraceni_koeficijenti[i])
        # dodaje faktor 10^stepen ako stepen nije 0
        if stepeni[i] != 0:
            ispis += "*10^" + str(stepeni[i]) + " "
        # dodaje faktor x^stepen ako stepen nije 0
        if stepen > 0:
            ispis += "*x^" + str(stepen) + " "
    ispis = ispis.strip()
    ispis = ispis.replace('+-', '-')
    return ispis

# za zaokruzivanje koeficijenta na k decimala prema pravilima
k = 2

#polinom sa koeficijentima kao clanovima niza
polinom = [mat.pi/1260 - 1/420, -mat.pi*mat.pi/1680 + mat.pi/840, -mat.pi/30 + 1/10, -mat.pi*mat.pi/60+mat.pi/30, mat.pi*2/3-2, 0, 0, 0, 0]
print(polinom)

# nizovi koeficijenta u prilagodjenom obliku
```

projekat2_2.py – Deo 3

```python
projekat2_2.py
            ispis += "*10^" + str(stepeni[i]) + " "
        # dodaje faktor x^stepen ako stepen nije 0
        if stepen > 0:
            ispis += "*x^" + str(stepen) + " "
    ispis = ispis.strip()
    ispis = ispis.replace('+-', '-')
    return ispis

# za zaokruzivanje koeficijenta na k decimala prema pravilima
k = 2

#polinom sa koeficijentima kao clanovima niza
polinom = [mat.pi/1260 - 1/420, -mat.pi*mat.pi/1680 + mat.pi/840, -mat.pi/30 + 1/10, -mat.pi*mat.pi/60+mat.pi/30, mat.pi*2/3-2, 0, 0, 0, 0]
print(polinom)

# nizovi koeficijenta u prilagodjenom obliku
koeficijenti = []
# nizovi stepeni broja 10 koji ce se mnoziti sa koeficijentima
stepeni = []

koeficijenti, stepeni = urediKoeficijenteStepene(polinom)
koeficijenti_skraceni = skratiKoeficijente(koeficijenti)
print("Koeficijenti nakon skracivanja su: " + str(koeficijenti_skraceni))
print("Stepeni nakon skracivanja su: " + str(stepeni))
print(ispisiPolinom(koeficijenti, koeficijenti_skraceni, stepeni))
```

projekat2_2.py – Deo 4

2.4. Testiranje rešenja drugog dela projektog zadatka

U nastavku će biti izloženo testiranje rešenja drugog dela projektnog zadatka sa još dva različita seta ulaznih podataka.

1. Test primer 1(primer sa predavanja):

```
projekat2_2
C:\Users\Korisnik\Desktop\OPNA-DZ2\venv\Scripts\python.exe C:\Users\Korisnik\Desktop\OPNA-DZ2\projekat2_2.py
[0.0001123751218966608, -0.002134773270184388, -0.00471975511965976, -0.05977365156516286, 0.09439510239319526, 0, 0, 0, 0]
Koeficijenti nakon skracivanja su: [1.12, -2.14, -4.72, -5.98, 9.43, 0.0, 0.0, 0.0, 0.0]
Stepeni nakon skracivanja su: [-4, -3, -3, -2, -2, 0, 0, 0, 0]
1.12*10^-4 *x^8 -2.14*10^-3 *x^7 -4.72*10^-3 *x^6 -5.98*10^-2 *x^5 + 9.43*10^-2 *x^4

Process finished with exit code 0
```

Test primer 1 - dobijanje polinoma sa racionalnim koeficijentima

```python
# polinom za testiranje
polinom = 1.12 * 10**-4 * x**8 - 2.14 * 10**-3 * x**7 - 4.72 * 10**-3 * x**6 - 5.98 * 10**-2 * x**5 + 9.43 * 10**-2 * x**4

# granice intervala [a, b]
a = 0 ; b = 1.35;
```

Test primer 1 – uvrštavanje polinoma u prvi deo projekta

```
projekat2_1
C:\Users\Korisnik\Desktop\OPNA-DZ2\venv\Scripts\python.exe C:/Users/Korisnik/Desktop/OPNA-DZ2/projekat2_1.py
P0(x) = 0.000112*x^8 - 0.00214*x^7 - 0.00472*x^6 - 0.0598*x^5 + 0.0943*x^4
P1(x) = 0.000896*x^7 - 0.01498*x^6 - 0.02832*x^5 - 0.299*x^4 + 0.3772*x^3
GCD(P0(x), P1(x)) = 1.0*x^3
********************************************************************
Šturmov niz polinoma:
********************************************************************
P0(x) = 0.000112*x^5 - 0.00214*x^4 - 0.00472*x^3 - 0.0598*x^2 + 0.0943*x
P1(x) = 0.00056*x^4 - 0.00856*x^3 - 0.01416*x^2 - 0.1196*x + 0.0943
P2(x) = 0.00843028571428571*x^3 + 0.0467022857142857*x^2 + 0.0159685714285714*x - 0.0720721428571428
P3(x) = -0.0493863269553692*x^2 + 0.0927218268255306*x + 0.00540326270070267
P4(x) = 0.0652308588755418 - 0.134289674120034*x
P5(x) = -0.0387899182667688
********************************************************************
Vrednosti Šturmovih polinoma:
********************************************************************
P0(0) = 0  P0(1.35) = 0.000100718369999981
P1(0) = 0.0943000000000000  P1(1.35) = -0.112167366500000
P2(0) = -0.0720721428571428  P2(1.35) = 0.0553420085000000
P3(0) = 0.00540326270070267  P3(1.35) = 0.0405711480390086
P4(0) = 0.0652308588755418  P4(1.35) = -0.116060201186504
P5(0) = -0.0387899182667688  P5(1.35) = -0.0387899182667688
********************************************************************
Broj promena znaka za granicu a: V(0) = 3
Broj promena znaka za granicu b: V(1.35) = 3
********************************************************************
Broj nula polinoma P(x) = 0.000112*x^8 - 0.00214*x^7 - 0.00472*x^6 - 0.0598*x^5 + 0.0943*x^4, na intervalu [a, b] = [0, 1.35] je: 0

Process finished with exit code 0
```

Test primer 1 – ispis rada prvog dela projekta

U okviru analize datog intervala, utvrđeno je da posmatrani polinom nema nule u tom opsegu. To se jasno vidi iz činjenice da je vrednost polinoma na donjoj granici intervala, na početnoj tački, jednaka nuli. Dalje, na gornjoj granici intervala, vrednost polinoma ostaje pozitivna. Ovi nalazi nas dovode do zaključka da je polinom u svim tačkama unutar odabranog intervala pozitivan. Drugim rečima, polinom zadržava pozitivne vrednosti bez obzira na tačku koju odaberemo unutar datog raspona.

2. Test primer 2:

```python
# polinom sa koeficijentima kao clanovima niza
polinom = [mat.pi, 0, 0, 0, 0, 2 * mat.sqrt(3), 3, -mat.sqrt(2) / mat.pi, 2, 0, mat.sqrt(5) - 1]
print(polinom)
```

Test primer 2 – koeficijenti ulaznog polinoma

```
projekat2_2
C:\Users\Korisnik\Desktop\OPNA-DZ2\venv\Scripts\python.exe C:\Users\Korisnik\Desktop\OPNA-DZ2\projekat2_2.py
[3.141592653589793, 0, 0, 0, 0, 3.4641016151377544, 3, -0.4501581580785531, 2, 0, 1.2360679774997898]
Koeficijenti nakon skracivanja su: [3.14, 0.0, 0.0, 0.0, 0.0, 3.46, 3.0, -4.51, 2.0, 0.0, 1.23]
Stepeni nakon skracivanja su: [0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0]
3.14*x^10 + 3.46*x^5 + 3.0*x^4 -4.51*10^-1 *x^3 + 2.0*x^2 + 1.23

Process finished with exit code 0
```

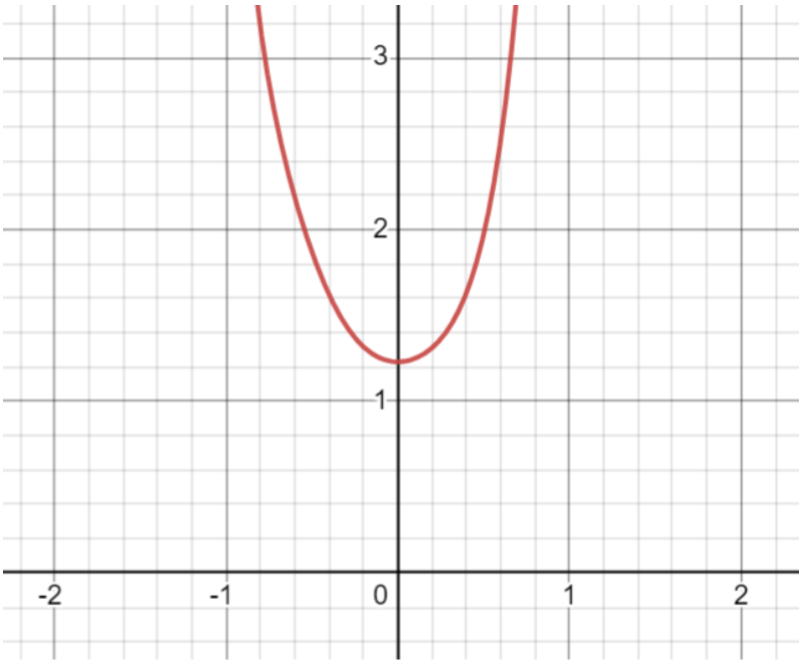
Test primer 2 – dobijanje polinoma sa racionalnim koeficijentima

```python
# polinom za testiranje
polinom = 3.14 * x**10 + 3.46 * x**5 + 3.0 * x**4 - 4.51 * 10**-1 * x**3 + 2.0 * x**2 + 1.23

# granice intervala [a, b]
a = -1 ; b = 1;
```

Test primer 2 – uvrštavanje polinoma u prvi deo projekta

```
C:\Users\Korisnik\Desktop\OPNA-DZ2\venv\Scripts\python.exe C:/Users/Korisnik/Desktop/OPNA-DZ2/projekat2_1.py
P0(x) = 3.14*x^10 + 3.46*x^5 + 3.0*x^4 - 0.451*x^3 + 2.0*x^2 + 1.23
P1(x) = 31.4*x^9 + 17.3*x^4 + 12.0*x^3 - 1.353*x^2 + 4.0*x
GCD(P0(x), P1(x)) = 1.00000000000000
********************************************************************
Šturmov niz polinoma:
********************************************************************
P0(x) = 3.14*x^10 + 3.46*x^5 + 3.0*x^4 - 0.451*x^3 + 2.0*x^2 + 1.23
P1(x) = 31.4*x^9 + 17.3*x^4 + 12.0*x^3 - 1.353*x^2 + 4.0*x
P2(x) = -1.73*x^5 - 1.8*x^4 + 0.3157*x^3 - 1.6*x^2 - 1.23
P3(x) = 177.306381780059*x^4 - 127.154180372538*x^3 + 137.696737659477*x^2 - 58.2708163975423*x + 83.1032068721897
P4(x) = 0.521365910759976*x^3 - 0.192830515606516*x^2 + 0.188448523167821*x - 0.195151757404259
P5(x) = -50.834687507689*x^2 - 30.3533347857678*x - 60.0546736187924
P6(x) = 0.126458592807727*x - 0.400422158933252
P7(x) = 665.848797612665
********************************************************************
Vrednosti Šturmovih polinoma:
********************************************************************
P0(-1) = 6.36100000000000  P0(1) = 12.3790000000000
P1(-1) = -31.4530000000000  P1(1) = 63.3470000000000
P2(-1) = -3.21570000000000  P2(1) = -6.04430000000000
P3(-1) = 583.531323081805  P3(1) = 212.681329541645
P4(-1) = -1.09779670693857  P4(1) = 0.321832160917021
P5(-1) = -80.5360263407136  P5(1) = -141.242695912249
P6(-1) = -0.526880751740979  P6(1) = -0.273963566125525
P7(-1) = 665.848797612665  P7(1) = 665.848797612665
********************************************************************
Broj promena znaka za granicu a: V(-1) = 4
Broj promena znaka za granicu b: V(1) = 4
********************************************************************
Broj nula polinoma P(x) = 3.14*x^10 + 3.46*x^5 + 3.0*x^4 - 0.451*x^3 + 2.0*x^2 + 1.23, na intervalu [a, b] = [-1, 1] je: 0

Process finished with exit code 0
```

Test primer 2 – ispis rada prvog dela projekta



Test primer 2 – grafik polinomske funkcije

Naša analiza polinoma pomoću Šturmove teoreme pokazuje da unutar odabranog intervala nema nula. Kada se u obzir uzmu pozitivne vrednosti polinoma na krajevima intervala, jasno je da polinom ne prelazi x-osu u bilo kojoj tački unutar intervala. Ovo dalje implicira da je polinom konstantno pozitivan kroz čitav opseg, potvrđujući da njegova vrednost ostaje pozitivna bez obzira na odabrane tačke unutar intervala. Takvim pristupom smo uspešno dokazali pozitivnost polinoma na datom intervalu.

SPISAK SLIKA

LITERATURA

[1] Branko J. Malešević, Sturmov algoritam (2023).pdf