Универзитет у Београду

Електротехнички факултет

Одабрана поглавља нумеричке анализе

A picture containing text, watch

Description automatically generated

Пројектни задатак 2

Штурмова теорема

Студент:

Александра Богићевић 0390/17

Београд, школска година 2021/2022

**Задатак:**

Нека је дат реални полином *P(x)*над реалним сегментом [a, b].

1. А) Одредити Еуклидовим алгоритмом највећи заједнички делилац

*Q(x) = GCD(P(x), P’(x))*

Б) За полином *P(x)= P(x)/ Q(x)*, употребом Штурмове теореме, одредити број нула на [a, b].

1. Примена Штурмове теореме.

Нека је *k* број децимала на који се заокружује неки реалан број. За полином

са реалним коефицијентима

одредити низ наниже заокружених рационалних коефицијената

осређених по следећим правилима:

* Ако је тада ;
* Ако је тада где је навише заокружена цифра (уз евентуално последично заокруживање и претходних цифара за један број навише).

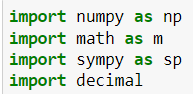
Одредити процедуру за формирање рационалног полинома

P

Нека је [a, b] сегмент са рационалним рубним тачкама. Наћи такав полином P(x) са реалним коефицијентима и број k, да на основу позитивности полинома P(x) над сегментом [a, b] имамо доказ о позитизивности полинома P(x) над сегментом [a, b].

**Напомена**:

За рачунање решења коришћен је програмски језик *Python* и његове библиотеке *numpy,math, sympy* и *decimal*. Радно окружење је *Jupуter notebook*. На дну документа се налази прилог са целокупним кодом.



**Решење (1):**

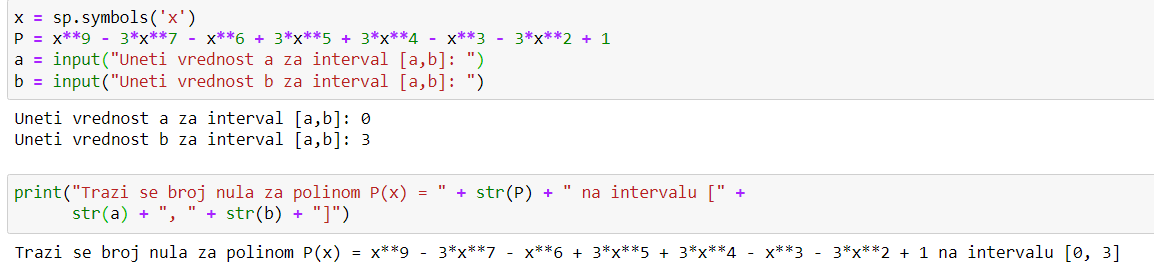
Пре свега треба дефинисати полином P(x), као и интервал [a, b]. Полином је кодован у програму, док се интервал уноси са стандардног улаза. Посматраћемо пар примера.

**Пример 1**

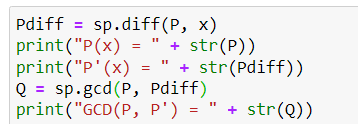
За вредности интервала уносимо вредности:

a = 0

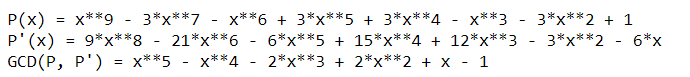
b = 3



Ставку (А) решавамо позивањем уграђене функције за тражење највећег заједничког делиоца (НЗД у наставку) за полином Р(х) и Р’(x). Добијена вредност се смешта у променљиву Q.



Излаз програма:



Добивши НЗД полинома Р(х) и Р’(x) можемо да почнемо решавање ставке (Б).

Први корак Штурмове теореме је формирање низа полинома

*Р0(х), Р1(х), Р2(х),...,Рi(x)*

као

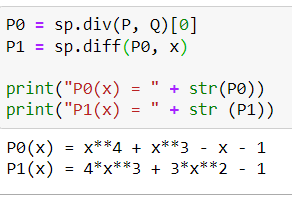
*P0(x) = P(x)/Q(x),*

*P1(x) = P’(x),*

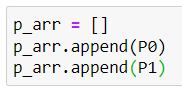
*Pi+1(x) = −REM(Pi(x), Pi−1(x))* редом за *i = 1, 2, . . . , r−1* и *Pr(x) = C– Const.*

Почетни полином Р0(х) израчунавамо као количник полинома Р(х) и Q(х).

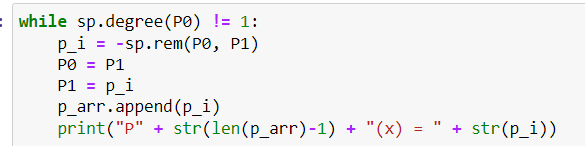
Полином Р1(х) израчунавамо као први извод полинома Р0(х).



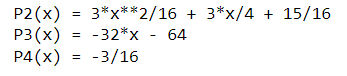
Добијене полиноме треба памтити како бисмо их касније користили, тако да уводимо низ полинома у којем чувамо претходно добијене полиноме Р0 и Р1.



Наставак израчунавања полинома може се представити једном петљом.



Након извршене петље добијамо следеће:



Овиме смо добили цео низ полинома и можемо прећи на наредни корак.

Следећи корак Штурмове теореме гласи:

Нека је *ξ ∈* [*a, b*] означимо *V* (*ξ*) број промена знакова у низу *P*0(*ξ*)*, P*1(*ξ*)*, . . . , Pr*(*ξ*), игноришући евентуално јављање корена полинома у том низу. Тада разлика

*N* = *V* (*a*) *− V* (*b*)

oдређује број нула полинома Р(х) на сегменту.

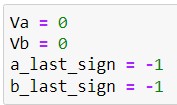
Другим речима:

За сваки полином у запамћеном низу полинома Р0(х), Р1(х), Р2(х),...,Рi(x) израчунаваћемо вредност за х=а и израчунати број промена знакова. Резултат ћемо сместити у променљиву Va. Исто то ћемо урадити и за х=b и резултат ћемо сместити у променљиву Vb.

Да бисмо добили број нула почетног полинома израчунаћемо разлику Va и Vb, и то сместити у променљиву N и тиме добити коначно решење.

У коду је то одрађено на следећи начин:

Уводимо нове променљиве

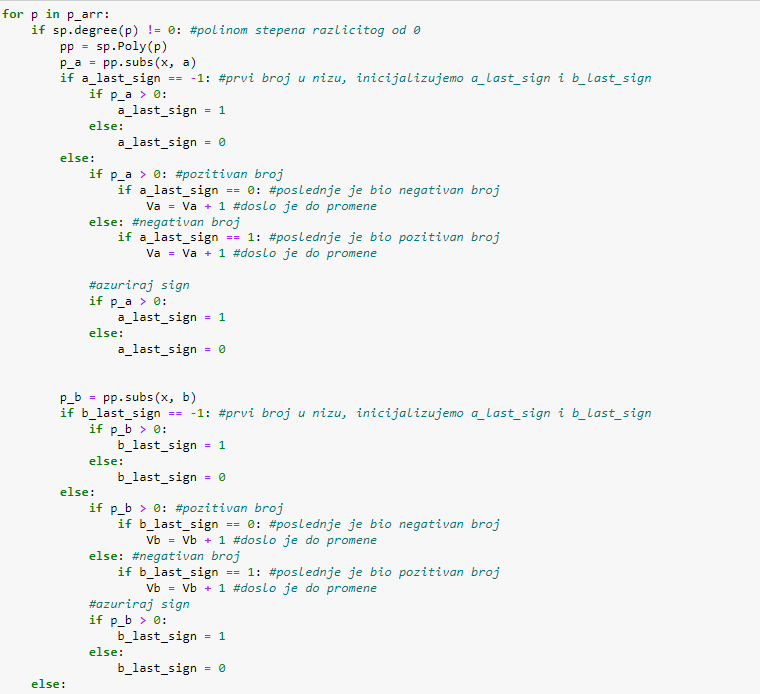


при чему Va и Vb представљају број промена знакова, и иницијално су постављени на нулу.

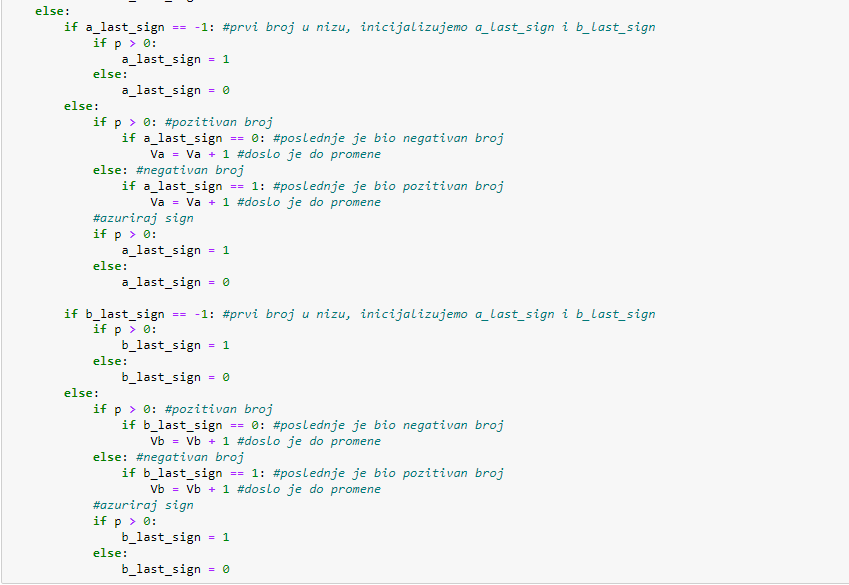
Променљиве a\_last\_sign и b\_last\_sign су помоћне променљиве за гледање да ли је дошло до промене знака или није. Оне означавају ког знака је последње израчунати полином. Вредност -1 означава иницијалну вредност, тј. да још увек ни један полином није израчунат. Уколико се добије негативна вредност полинома за х=а, а\_last\_sign се поставља на 0, а уколико се добије позитивна вредност, а\_last\_sign се поставља на 1. Исто важи и за х=b и b\_last\_sign.

При израчунавању сваког полинома пореди се његов знак са знаком претходно израчунатог полинома. Уколико је дошло до промене знака, увећава се променљива Va (или Vb) и ажурира се a\_last\_sign (или b\_last\_sign). Уколико није дошло до промене знака променљиве Va и Vb остају непромењене.

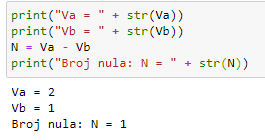
Да би замена х=а или х=b уопште била могућа, потребно је да степен полинома буде различит од нуле.



Уколико је степен полинома једнак нули, не врши се замена већ се само посматра знак и ажурирају вредности уколико је потребно.

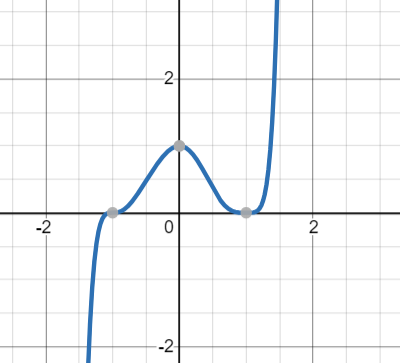


Када смо израчунали вредности Va и Vb можемо да израчунамо и коначни резултат N.



***Провера решења***:

Нуле полинома можемо уочити на следећем графику



Као што се може приметити, на интервалу [0, 3] заиста постоји једна нула.

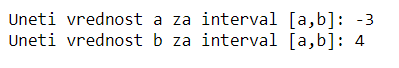
**Пример 2**

За вредности интервала уносимо вредности:

a = -3

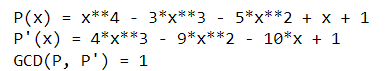
b = 4

***Излаз програма***:

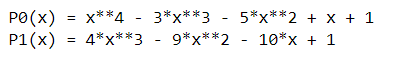


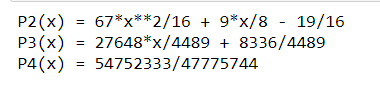


Тражење НЗД:

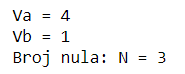


Формирање низа полинома:



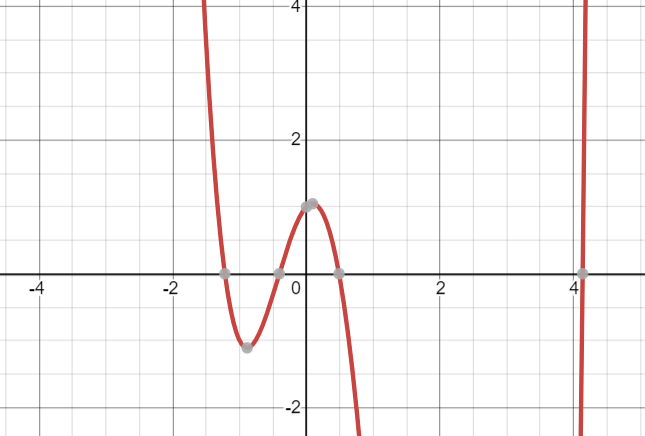


Број нула:



***Провера решења***:

Нуле полинома можемо уочити на следећем графику



Као што се може приметити, на интервалу [-3, 4] заиста постоје три нуле.

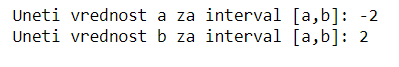
**Пример 3**

За вредности интервала уносимо вредности:

a = -2

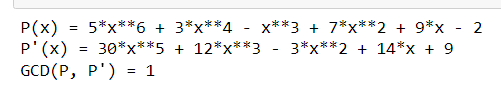
b = 2

***Излаз програма***:

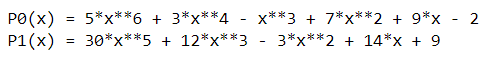


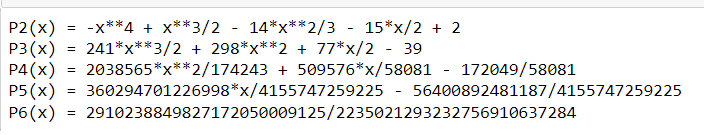


Тражење НЗД:

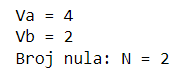


Формирање низа полинома:



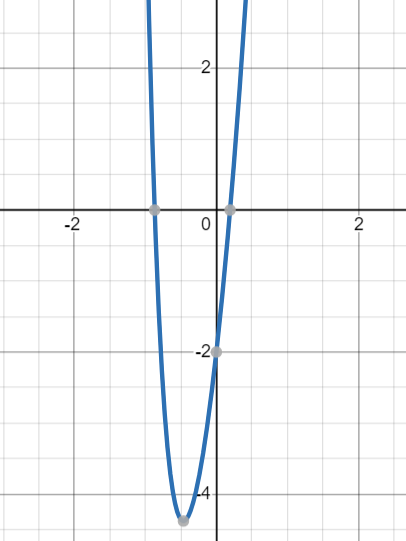


Број нула:



***Провера решења***:

Нуле полинома можемо уочити на следећем графику

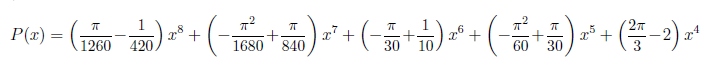


Као што се може приметити, на интервалу [-2, 2] заиста постоје две нуле.

**Решење (2):**

Посматрамо полиномску функцију са реалним коефицијентима.

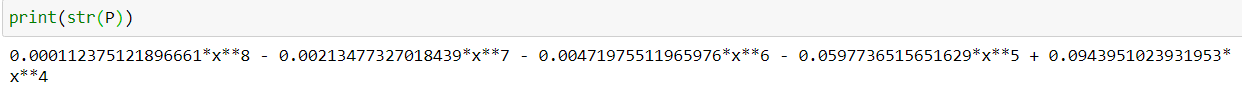
**Пример 1**



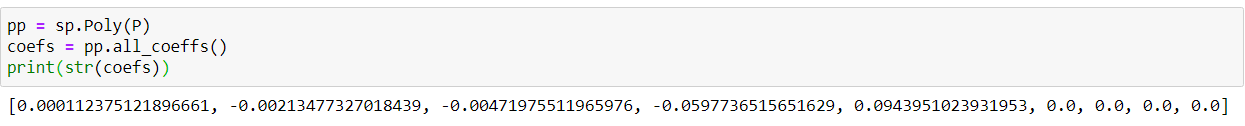
Показаћемо како се може уз нанижу апроксимацију коефицијената разломцима и уз употребу Штурмове теореме доказати Р(х)>0 над интервалом (0, 1.35).

Пре свега уносимо полином и вредност k која представља број децимала на који се заокружује неки реалан број.

Израчунавањем вредности Р(х) добијамо:

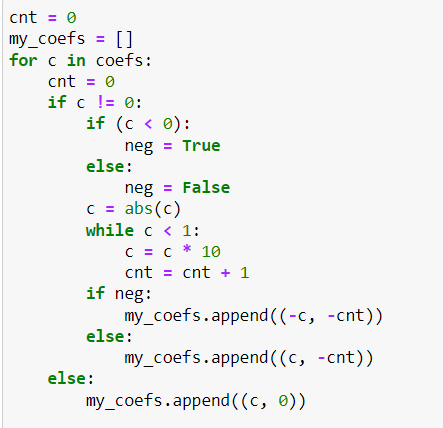


Из оваквог полинома издвајамо коефицијенте у један низ.



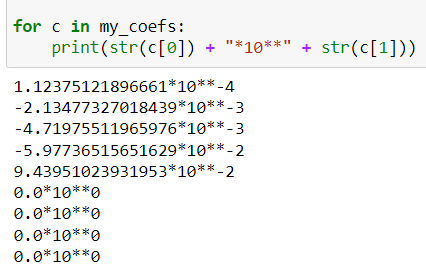
Када смо издвојили коефицијенте, можемо да их скраћујемо.

Пре свега желимо да радимо са бројевима чији је цели део већи од нуле, тј. да децимални број представимо умношком броја 10. Таква трансформација је извршена у следећем делу кода:

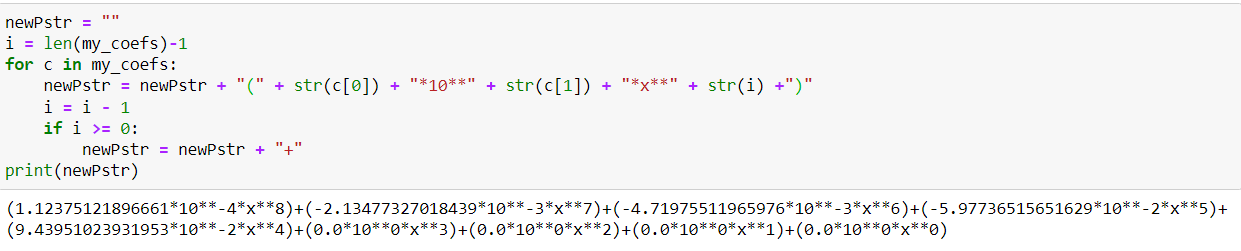


За сваки коефицијент се памти децимални део као и степен умношка.

Извршавањем ове петље добијамо коефицијенте који изгледају овако:

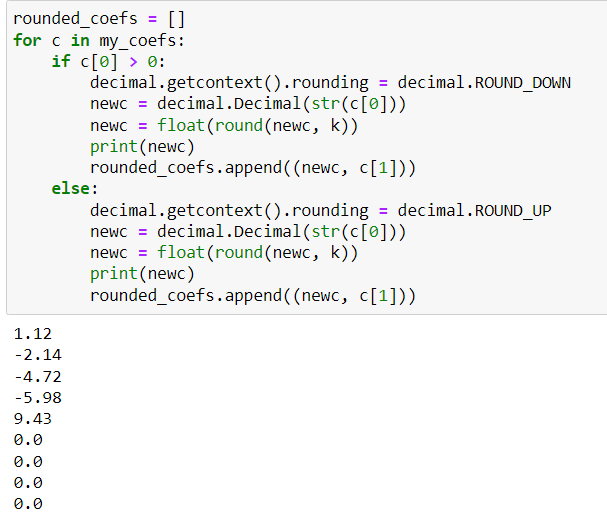


Тако да наш полином сада изгледа овако:

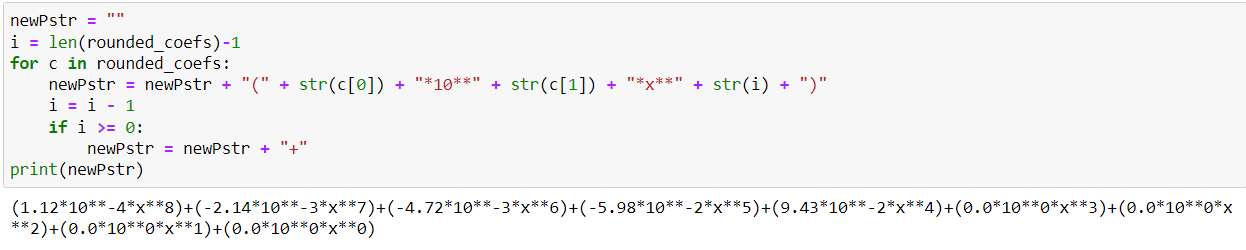


Када имамо издвојене коефицијенте, можемо да извршимо скраћивање на k децимала следећим правилом:

* Ако је тада ;
* Ако је тада где је навише заокружена цифра (уз евентуално последично заокруживање и претходних цифара за један број навише).

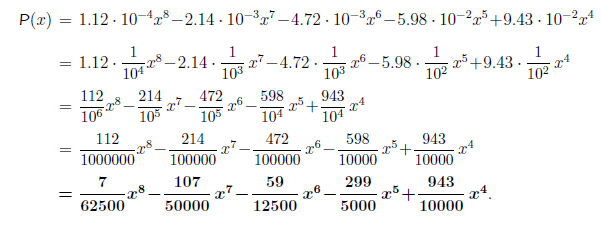


Након заокруживања наш полином изгледа овако:



Овако написан полином морамо средити ручно због недотатка програмског језика за представљање разломака.

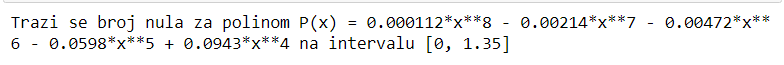
Коефицијенте претварамо у разломке:

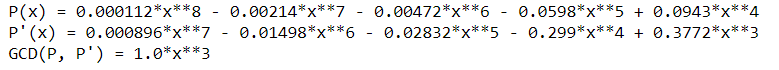


Сређен полином уносимо у програм који смо направили за ставку 1, како бисмо добили број нула полинома на интервалу [0, 1.35]

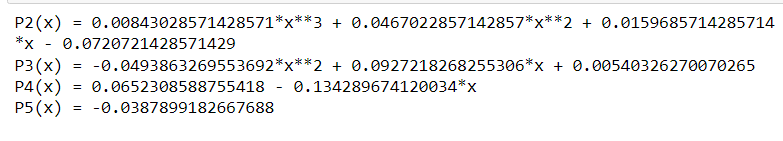
Излаз програма:

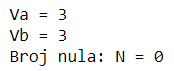












Добијамо да овај полином нема нуле на том интервалу. На осову тога следи:

P(1,35)=0,0002478.. > 0

Следи закључак



На основу поретка

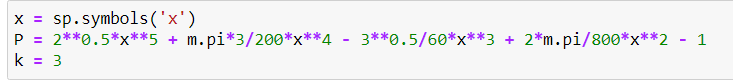


Уједно је доказана и позитвност полинома P(x).

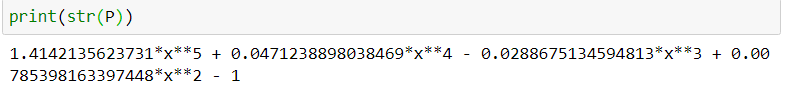
**Пример 2:**

Почетни полином:

k = 3



Израчунавањем добијамо:



Сређивање коефицијената:



Заокруживање коефицијената на три децимале:

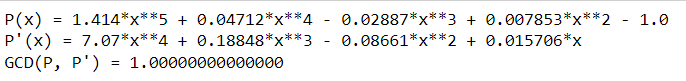


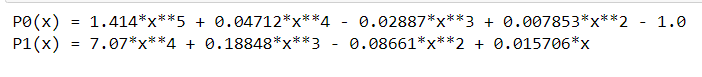
Претварање децималних бројева у разломке:

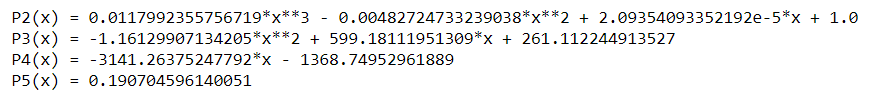
Израчунавање нула:

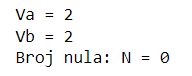












Добијамо:

*P(x)>0*

Из тога P(x) > *P*(x) следи:

**Прилог:**

У наставку следе кодови у целости:

**Код 1:**

#!/usr/bin/env python

# coding: utf-8

# In[1]:

**import** numpy **as** np

**import** math **as** m

**import** sympy **as** sp

# P(x) -> pocetni polinom

#

# P = x\*\*9 - 3\*x\*\*7 - x\*\*6 + 3\*x\*\*5 + 3\*x\*\*4 - x\*\*3 - 3\*x\*\*2 + 1

#

# P = x\*\*4 - 3\*x\*\*3 - 5\*x\*\*2 + x + 1

#

# P = 5\*x\*\*6 + 3\*x\*\*4 - x\*\*3 + 7\*x\*\*2 + 9\*x - 2

#

# P = (7/62500\*x\*\*8)+(-107/50000\*x\*\*7)+(-59/12500\*x\*\*6)+(-299/5000\*x\*\*5)+(943/10000\*x\*\*4)

#

# P = (1.414\*10\*\*0\*x\*\*5)+(4.712\*10\*\*-2\*x\*\*4)+(-2.887\*10\*\*-2\*x\*\*3)+(7.853\*10\*\*-3\*x\*\*2)-1

#

# P = (707/500\*x\*\*5)+(589/12500\*x\*\*4)+(-2887/100000\*x\*\*3)+(7853/1000000\*x\*\*2)-1

# In[2]:

x **=** sp**.**symbols**(**'x'**)**

P **=** **(**707**/**500**\***x**\*\***5**)+(**589**/**12500**\***x**\*\***4**)+(-**2887**/**100000**\***x**\*\***3**)+(**7853**/**1000000**\***x**\*\***2**)-**1

a **=** input**(**"Uneti vrednost a za interval [a,b]: "**)**

b **=** input**(**"Uneti vrednost b za interval [a,b]: "**)**

# In[3]:

**print(**"Trazi se broj nula za polinom P(x) = " **+** str**(**P**)** **+** " na intervalu [" **+**

str**(**a**)** **+** ", " **+** str**(**b**)** **+** "]"**)**

# Uz pomoc Euklidovog algoritma trazimo GCD za P i P'

# In[4]:

Pdiff **=** sp**.**diff**(**P**,** x**)**

**print(**"P(x) = " **+** str**(**P**))**

**print(**"P'(x) = " **+** str**(**Pdiff**))**

Q **=** sp**.**gcd**(**P**,** Pdiff**)**

**print(**"GCD(P, P') = " **+** str**(**Q**))**

# P0 = P/ Q

# In[5]:

P0 **=** sp**.**div**(**P**,** Q**)[**0**]**

P1 **=** sp**.**diff**(**P0**,** x**)**

**print(**"P0(x) = " **+** str**(**P0**))**

**print(**"P1(x) = " **+** str **(**P1**))**

# moramo imati niz u kom cemo cuvati sve Pi vrednosti

# In[6]:

p\_arr **=** **[]**

p\_arr**.**append**(**P0**)**

p\_arr**.**append**(**P1**)**

# In[7]:

**while** sp**.**degree**(**P0**)** **!=** 1**:**

p\_i **=** **-**sp**.**rem**(**P0**,** P1**)**

P0 **=** P1

P1 **=** p\_i

p\_arr**.**append**(**p\_i**)**

**print(**"P" **+** str**(**len**(**p\_arr**)-**1**)** **+** "(x) = " **+** str**(**p\_i**))**

# Kada imamo niz, mozemo da odredimo broj nula na intervalu [a,b] tako sto cemo odrediti sve Pi(a) i Pi(b) i sracunati razliku broja promene znakova.

# a\_last\_sign i b\_last\_sign predstavljaju poslednji znak, i s njim uporedjujemo sledeci. 0 = negativno, 1 = pozitivno

# In[8]:

Va **=** 0

Vb **=** 0

a\_last\_sign **=** **-**1

b\_last\_sign **=** **-**1

# In[9]:

**for** p **in** p\_arr**:**

**if** sp**.**degree**(**p**)** **!=** 0**:** #polinom stepena razlicitog od 0

pp **=** sp**.**Poly**(**p**)**

p\_a **=** pp**.**subs**(**x**,** a**)**

**if** a\_last\_sign **==** **-**1**:** #prvi broj u nizu, inicijalizujemo a\_last\_sign i b\_last\_sign

**if** p\_a **>** 0**:**

a\_last\_sign **=** 1

**elif** p\_a **<** 0**:**

a\_last\_sign **=** 0

**else:**

**if** p\_a **>** 0**:** #pozitivan broj

**if** a\_last\_sign **==** 0**:** #poslednje je bio negativan broj

Va **=** Va **+** 1 #doslo je do promene

**else:** #negativan broj

**if** a\_last\_sign **==** 1**:** #poslednje je bio pozitivan broj

Va **=** Va **+** 1 #doslo je do promene

#azuriraj sign

**if** p\_a **>** 0**:**

a\_last\_sign **=** 1

**elif** p\_a **<** 0**:**

a\_last\_sign **=** 0

p\_b **=** pp**.**subs**(**x**,** b**)**

**if** b\_last\_sign **==** **-**1**:** #prvi broj u nizu, inicijalizujemo a\_last\_sign i b\_last\_sign

**if** p\_b **>** 0**:**

b\_last\_sign **=** 1

**elif** p\_b **<** 0**:**

b\_last\_sign **=** 0

**else:**

**if** p\_b **>** 0**:** #pozitivan broj

**if** b\_last\_sign **==** 0**:** #poslednje je bio negativan broj

Vb **=** Vb **+** 1 #doslo je do promene

**elif** p\_b **<** 0**:** #negativan broj

**if** b\_last\_sign **==** 1**:** #poslednje je bio pozitivan broj

Vb **=** Vb **+** 1 #doslo je do promene

#azuriraj sign

**if** p\_b **>** 0**:**

b\_last\_sign **=** 1

**else:**

b\_last\_sign **=** 0

**else:**

**if** a\_last\_sign **==** **-**1**:** #prvi broj u nizu, inicijalizujemo a\_last\_sign i b\_last\_sign

**if** p **>** 0**:**

a\_last\_sign **=** 1

**elif** p **<** 0**:**

a\_last\_sign **=** 0

**else:**

**if** p **>** 0**:** #pozitivan broj

**if** a\_last\_sign **==** 0**:** #poslednje je bio negativan broj

Va **=** Va **+** 1 #doslo je do promene

**elif** p **<** 0**:** #negativan broj

**if** a\_last\_sign **==** 1**:** #poslednje je bio pozitivan broj

Va **=** Va **+** 1 #doslo je do promene

#azuriraj sign

**if** p **>** 0**:**

a\_last\_sign **=** 1

**elif** p **<** 0**:**

a\_last\_sign **=** 0

**if** b\_last\_sign **==** **-**1**:** #prvi broj u nizu, inicijalizujemo a\_last\_sign i b\_last\_sign

**if** p **>** 0**:**

b\_last\_sign **=** 1

**elif** p **<** 0**:**

b\_last\_sign **=** 0

**else:**

**if** p **>** 0**:** #pozitivan broj

**if** b\_last\_sign **==** 0**:** #poslednje je bio negativan broj

Vb **=** Vb **+** 1 #doslo je do promene

**elif** p **<** 0**:** #negativan broj

**if** b\_last\_sign **==** 1**:** #poslednje je bio pozitivan broj

Vb **=** Vb **+** 1 #doslo je do promene

#azuriraj sign

**if** p **>** 0**:**

b\_last\_sign **=** 1

**elif** p **<** 0**:**

b\_last\_sign **=** 0

# In[10]:

**print(**"Va = " **+** str**(**Va**))**

**print(**"Vb = " **+** str**(**Vb**))**

N **=** Va **-** Vb

**print(**"Broj nula: N = " **+** str**(**N**))**

**Код 2:**

#!/usr/bin/env python

# coding: utf-8

# In[1]:

**import** numpy **as** np

**import** math **as** m

**import** sympy **as** sp

**import** decimal

# P = (m.pi/1260 - 1/420)\*x\*\*8 + (-(m.pi\*\*2/1680)+m.pi/840)\*x\*\*7 + \((m.pi/30)+1/10)\*x\*\*6 + (-(m.pi\*\*2/60)+m.pi/30)\*x\*\*5 +(2\*m.pi/3-2)\*x\*\*4

#

# P = 2\*\*0.5\*x\*\*5 + m.pi\*3/200\*x\*\*4 - 3\*\*0.5/60\*x\*\*3 + 2\*m.pi/800\*x\*\*2 - 1

# In[2]:

x **=** sp**.**symbols**(**'x'**)**

P **=** 2**\*\***0.5**\***x**\*\***5 **+** m**.**pi**\***3**/**200**\***x**\*\***4 **-** 3**\*\***0.5**/**60**\***x**\*\***3 **+** 2**\***m**.**pi**/**800**\***x**\*\***2 **-** 1

k **=** 3

# In[3]:

**print(**str**(**P**))**

# In[4]:

pp **=** sp**.**Poly**(**P**)**

coefs **=** pp**.**all\_coeffs**()**

**print(**str**(**coefs**))**

# Sredjivanje koeficijenata na oblik 1,...\*10^..

#

# my\_coefs predstavlja novi niz koeficijenata, koji se sastoji iz parova vrednosti

#

# prva vrednost predstavlja koeficijent, a druga umnozak broja 10

# In[5]:

cnt **=** 0

my\_coefs **=** **[]**

**for** c **in** coefs**:**

cnt **=** 0

**if** c **!=** 0**:**

**if** **(**c **<** 0**):**

neg **=** **True**

**else:**

neg **=** **False**

c **=** abs**(**c**)**

**while** c **<** 1**:**

c **=** c **\*** 10

cnt **=** cnt **+** 1

**if** neg**:**

my\_coefs**.**append**((-**c**,** **-**cnt**))**

**else:**

my\_coefs**.**append**((**c**,** **-**cnt**))**

**else:**

my\_coefs**.**append**((**c**,** 0**))**

**for** c **in** my\_coefs**:**

**print(**str**(**c**[**0**])** **+** "\*10\*\*" **+** str**(**c**[**1**]))**

# In[6]:

newPstr **=** ""

i **=** len**(**my\_coefs**)-**1

**for** c **in** my\_coefs**:**

newPstr **=** newPstr **+** "(" **+** str**(**c**[**0**])** **+** "\*10\*\*" **+** str**(**c**[**1**])** **+** "\*x\*\*" **+** str**(**i**)** **+**")"

i **=** i **-** 1

**if** i **>=** 0**:**

newPstr **=** newPstr **+** "+"

**print(**newPstr**)**

# Pravilo za zaokruzivanje:

#

# ako je koeficijent pozitivan, zaokruzuje se na donju brojku

#

# ako je negativan, zaokruzuje se na gornju

# In[7]:

rounded\_coefs **=** **[]**

**for** c **in** my\_coefs**:**

**if** c**[**0**]** **>** 0**:**

decimal**.**getcontext**().**rounding **=** decimal**.**ROUND\_DOWN

newc **=** decimal**.**Decimal**(**str**(**c**[**0**]))**

newc **=** float**(**round**(**newc**,** k**))**

**print(**newc**)**

rounded\_coefs**.**append**((**newc**,** c**[**1**]))**

**else:**

decimal**.**getcontext**().**rounding **=** decimal**.**ROUND\_UP

newc **=** decimal**.**Decimal**(**str**(**c**[**0**]))**

newc **=** float**(**round**(**newc**,** k**))**

**print(**newc**)**

rounded\_coefs**.**append**((**newc**,** c**[**1**]))**

# Ispis zaokruzenog polinoma

# In[8]:

newPstr **=** ""

i **=** len**(**rounded\_coefs**)-**1

**for** c **in** rounded\_coefs**:**

newPstr **=** newPstr **+** "(" **+** str**(**c**[**0**])** **+** "\*10\*\*" **+** str**(**c**[**1**])** **+** "\*x\*\*" **+** str**(**i**)** **+** ")"

i **=** i **-** 1

**if** i **>=** 0**:**

newPstr **=** newPstr **+** "+"

**print(**newPstr**)**