Универзитет у Београду

Електротехнички факултет

Одабрана поглавља нумеричке анализе

A picture containing text, watch

Description automatically generated

Пројектни задатак 3

Миксовано тригонометријско полиномске неједнакости

Студент:

Александра Богићевић 0390/17

Београд, школска година 2021/2022

**Миксовано тригонометријско полиномске неједнакости – МТП неједнакости**

МТП функција представља функцију

за и за вредности аргумента , при стандардној вредности . За МТП функцију *f* основни проблем наниже апроксимације је да се одреди полином P такав да

*f(x) > P(x)*

за . Уколико за полином *P* важи полиномска неједнакост

*P(x) > 0*

за , тада за МТП функцију *f* важи МТП неједнакост

*f(x) > 0*

за .

Користићемо Маклоренове полиноме тригонометријских функција cos и sin у циљу одређивања навиших и нанижих полиномских апроксимација МТП функција. Користимо ознаку за Тејлоров полином степена *k* функције у тачки *a* и специјално за *a* = 0 такав полином уобичајено називамо Маклоренов полином.

Schematic

Description automatically generated

за и .

**Теорема 1:**

За Маклоренове полиноме одговарајућег степена *cos* и *sin* функција важе неједнакости:

Diagram, schematic

Description automatically generated with medium confidence

за реалне вредности аргумента х.

**Теорема 2:**

За ма коју МТП функцију

постоји полином Р као нанижа полиномска апроксимација МТП функције *f* таква да важи

*f(x) > P(x)*

за вредности аргумента .

**Задатак:**

За погодно изабрану МТП функцију доказати МТП неједнакости над , одређујући позитивну нанижу полиномску апроксимацију над .

**Напомена**:

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

**Text

Description automatically generated**

**Решење**:

За одређивање наниже полиномске апроксимације *Р(х)* користићемо **метод директног поређења**.

Метод директног поређења се користи за класу простих МТП функција облика

за и за вредности аргумента , при стандардној вредности . У циљу одређивања полинома *Р(х)* могуће је користити процене:

Text, letter

Description automatically generated

за .

*Напомена 1 - Ограничење методе директног поређења:*

Приметимо да

и при том има јединствен корен cs ∈ (0, π/2). Стога за сабирке са позитивним коефицијентима уз парне степене важи

*Напомена 2 - Превазилажење ограничења методе директног поређења:*

Остварује се трансформацијом просте MTП функције тако што се за сабирке са позитивним коефицијентима уз парне степене примени трансформација

A picture containing text, watch

Description automatically generated

Да бисмо израчунали Маклоренов полином степена *n* користимо функцију *maclaurin*, којој проследђујемо врсту функцију *f* и степен *n*.

Graphical user interface, text, application

Description automatically generated

Користећи функцију која је детаљно објашњена у другом пројектном задатку, сада само оптималније написана, на основу Штурмове теореме одређујемо да ли полиномска функција има нуле на задатом сегменту. Уколико на датом сегменту постоје нуле, повратна вредност функције је *true*, уколико нема, повратна вредност је *false.*

Text

Description automatically generated

Text

Description automatically generated with low confidence

Text

Description automatically generated

Text

Description automatically generated

Имајући ове две функције, можемо да напишемо функцију која представља метод директног поређења.

Пре свега треба пронаћи вредности α и β, тј. одредити знак синуса и косинуса. То радимо уз помоћ метода *find\_alfa, find\_beta* које као аргумент примају функцију, а повратна вредност им је вредност α и β. Те вредности уноси корисник због немогућности окружења да то самостално одреди. Након тога обављамо поређење и извршавамо одговарајуће замене у циљу одређивања одговарајућихј нанижних апроксимација функције.

Позивањем функције *ima\_nule* одређујемо да ли је прослеђени пар *k1,k2* одговарајући.

* Уколико функција има нуле, значи да није позитивна на целом сегменту и да није одговарајућа апроксимација.
* Уколико нема нуле, посматрамо вредност функције
  + Уколико је негативна – апроксимација није одоварајућа
  + Уколико је позитивна – апроксимација је одговарајућа

Text

Description automatically generated

У главном делу програма уносимо у поље *f* вредност функције чију апроксимацију желимо да нађемо и позивамо функцију *ispitaj.* На крају добијене вредности представљамо на графику.

Graphical user interface, text, application

Description automatically generated

**Пример 1**

У првом примеру узимамо следеће вредности:

Text, letter

Description automatically generated with medium confidence

Унос у програм:

Graphical user interface, text, application

Description automatically generated

Излаз програма:

Diagram

Description automatically generated

**Пример 2**

У другом примеру узимамо следеће вредности:

Text

Description automatically generated with medium confidence

Унос у програм:

Graphical user interface, text, application

Description automatically generatedИзлаз програма:Chart

Description automatically generated

**Пример 3**

У трећем примеру узимамо следеће вредности:

Text

Description automatically generated

Унос у програм:

Graphical user interface, text, application

Description automatically generated

Излаз програма:

A picture containing chart

Description automatically generated

**Пример 4**

У четвртом примеру узимамо следеће вредности:

Text, letter

Description automatically generated

Унос у програм:

Graphical user interface, text, application, chat or text message

Description automatically generated

Излаз програма:

Chart

Description automatically generated with medium confidence

**Прилог**

#!/usr/bin/env python

# coding: utf-8

# In[1]:

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

**import** math

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

**import** matplotlib**.**pyplot **as** plt

**from** colorama **import** Fore

# In[2]:

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

# In[3]:

**def** maclaurin**(**f**,** n**):**

sign **=** **-**1

**if** f **==** 'cos'**:**

n **=** **(int)(**n **/** 2**)**

T **=** sp**.**Poly**(**1**,** x**)** # inicijalizujemo na T = 1

**for** k **in** **range(**1**,** n**+**1**):**

T **=** T**.**add**(**sp**.**Poly**(**sign**\*\***k **\*** x**\*\*(**2**\***k**)** **/** math**.**factorial**((**2**\***k**))))**

**elif** f **==** 'sin'**:**

n **=** **(int)((**n **-** 1**)** **/** 2**)**

T **=** sp**.**Poly**(**x**,** x**)** # inicijalizujemo na T = x

**for** k **in** **range(**1**,** n**+**1**):**

T **=** T**.**add**(**sp**.**Poly**(**sign**\*\***k **\*** x**\*\*(**2**\***k**+**1**)** **/** math**.**factorial**((**2**\***k**+**1**))))**

**return** T

# In[4]:

**def** ima\_nule**(**P**):**

# pravljenje sturmovog niza

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

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

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

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

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

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

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

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

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

P0 **=** P1

P1 **=** p\_i

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

# odredjujemo broj nula na intervalu [a, b]

a **=** 0.001

b **=** np**.**pi **/** 2

Va **=** 0

Vb **=** 0

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

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

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

**if** sp**.**degree**(**p**)** **!=** 0**:**

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

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

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

**else:**

p\_a **=** p**.**as\_expr**()**

p\_b **=** p**.**as\_expr**()**

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

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

a\_last\_sign **=** 1

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

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

a\_last\_sign **=** 1 #azuriramo znak

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

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

a\_last\_sign **=** 0

**elif** a\_last\_sign **==** 1**:** #poslednje je bio negativan broj

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

a\_last\_sign **=** 0 #azuriramo znak

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

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

b\_last\_sign **=** 1

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

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

b\_last\_sign **=** 1 #azuriramo znak

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

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

b\_last\_sign **=** 0

**elif** b\_last\_sign **==** 1**:** #poslednje je bio negativan broj

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

b\_last\_sign **=** 0 #azuriramo znak

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

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

**if** N **>** 0**:**

**return** **True**

**else:**

**return** **False**

# alfa uz cos, beta uz sin

# In[5]:

**def** ispitaj**(**f**,** k1**,** k2**):**

alfa **=** find\_alfa**(**f**)**

beta **=** find\_beta**(**f**)**

**if** alfa **>** 0**:** # pozitivno je -> gledamo od cega je cos veci

cos\_index **=** 4 **\*** k1 **+** 2

**elif** alfa **<** 0**:** # negativno je -> gledamo od cega je cos manji

cos\_index **=** 4 **\*** k1

**if** beta **>** 0**:** # pozitivno je -> gledamo od cega je sin veci

sin\_index **=** 4 **\*** k2 **+** 3

**elif** beta **<** 0**:** # negativno je -> gledamo od cega je sin manji

sin\_index **=** 4 **\*** k2 **+** 1

p **=** f**.**subs**(**sp**.**cos**(**x**),** maclaurin**(**"cos"**,** cos\_index**).**as\_expr**())**

p **=** p**.**subs**(**sp**.**sin**(**x**),** maclaurin**(**"sin"**,** sin\_index**).**as\_expr**())**

**if** ima\_nule**(**p**):**

# AKO FUNKCIJA IMA NULE NA DATOM SEGMENTU - NE DOKAZUJE POZITIVNOST

**print(**"[NOT OK] P[" **+** **str(**k1**)** **+** "," **+** **str(**k2**)**

**+** "](x) ne dokazuje pozitivnost MTP funkcije nad (0, Pi/2) jer ima nule na tom segmentu"**)**

**else:**

# AKO FUNKCIJA NEMA NULE - ODREDJUJEMO ZNAK FUNKCIJE

fSign**=** sp**.**Poly**(**p**).**subs**(**x**,** 1**)**

**if** fSign **<** 0**:**# AKO JE NEGATIVNA - NE DOKAZUJE POZITIVNOST

**print(**"[NOT OK] P[" **+** **str(**k1**)** **+** "," **+** **str(**k2**)**

**+** "](x) ne dokazuje pozitivnost MTP funkcije nad (0, Pi/2) jer je negativna na zadatom domenu"**)**

**else:** # AKO JE POZITIVNA - DOKAZUJE POZITIVNOST

**print(**"[OK] P[" **+** **str(**k1**)** **+** "," **+** **str(**k2**)**

**+** "](x) dokazuje pozitivnost MTP funkcije nad (0, Pi/2)."**)**

**return** p

# ## Primer 1

# In[ ]:

**def** find\_alfa**(**f**):** #uz cos

**return** **-**1

**def** find\_beta**(**f**):** # uz sin

**return** 1

f **=** x**\*\***3 **\*** sp**.**sin**(**x**)** **-** x **\*** sp**.**cos**(**x**)\*\***3 **+** x **-** 3**/**2 **\*** x**\*\***3 **+** 3**/**32 **\*** x**\*\***4

# ## Primer 2

# In[20]:

**def** find\_alfa**(**f**):** # uz cos

**return** **-**1

**def** find\_beta**(**f**):** # uz sin

**return** **-**1

f **=** **-**4**/**6**\***x**\*\***3**\***sp**.**sin**(**x**)** **-** 3**\***x**\*\***2 **\*** sp**.**cos**(**x**)/**20 **+** x**\*\***5 **+** 8**/**98**\***x**\*\***2 **+** 1

# ## Primer 3

# In[30]:

**def** find\_alfa**(**f**):** # uz cos

**return** **-**1

**def** find\_beta**(**f**):** # uz sin

**return** 1

f **=** x**\*\***3 **\*** sp**.**sin**(**x**)** **-** x**/**5 **\*** sp**.**cos**(**x**)\*\***3 **+**x**\***6**/**9 **-** 6**/**5**\***x**\*\***3 **+** 9**/**59**\***x**\*\***2

# ## Primer 4

# In[33]:

**def** find\_alfa**(**f**):** #uz cos

**return** **-**1

**def** find\_beta**(**f**):** # uz sin

**return** 1

f **=** x**\*\***6 **\*** sp**.**sin**(**x**)** **-** x**/**5 **\*** sp**.**cos**(**x**)\*\***3 **-** 6**/**5**\*** x**\*\***3 **+** 5**/**9 **-** x**\*\***4 **+** 1

# # GLAVNI PROGRAM

# In[34]:

p1 **=** ispitaj**(**f**,** 0**,** 0**)**

p2 **=** ispitaj**(**f**,** 0**,** 1**)**

p3 **=** ispitaj**(**f**,** 1**,** 0**)**

p4 **=** ispitaj**(**f**,** 1**,** 1**)**

graf0 **=** sp**.**plot**(**f**,** xlim**=[-**1**,** 2**],** ylim**=[-**1.25**,** 5**],** show**=False,** label**=**"f(x)"**)**

graf1 **=** sp**.**plot**(**p1**.**as\_expr**(),** xlim**=[-**1**,** 2**],** ylim**=[-**1.25**,** 5**],** show**=False,** label**=**"$P\_{0,0}$"**)**

graf2 **=** sp**.**plot**(**p2**.**as\_expr**(),** xlim**=[-**1**,** 2**],** ylim**=[-**1.25**,** 5**],** show**=False,** label**=**"$P\_{0,1}$"**)**

graf3 **=** sp**.**plot**(**p3**.**as\_expr**(),** xlim**=[-**1**,** 2**],** ylim**=[-**1.25**,** 5**],** show**=False,** label**=**"$P\_{1,0}$"**)**

graf4 **=** sp**.**plot**(**p4**.**as\_expr**(),** xlim**=[-**1**,** 2**],** ylim**=[-**1.25**,** 5**],** show**=False,** label**=**"$P\_{1,1}$"**)**

graf0**.**append**(**graf1**[**0**])**

graf0**.**append**(**graf2**[**0**])**

graf0**.**append**(**graf3**[**0**])**

graf0**.**append**(**graf4**[**0**])**

graf0**.**legend **=** **True**

graf0**.**show**()**