Kauno technologijos universitetas

Informatikos fakultetas

P170B115 Skaitiniai metodai ir algoritmai

1 projektinė užduotis. Netiesinių lygčių sprendimas.

Martynas Kuliešius IFF-1/9

Studentas

doc. Čalnerytė Dalia

Dėstytoja

KAUNAS, 2023

Turinys

Įvadas

Darbo tikslas - rasti vienos netiesinės lygties šaknis naudojant šaknų atskyrimo (grafinį, skenavimo) ir jų tikslinimo (Niutono, kirstinių, paprastųjų iteracijų, stygų, pusiaukirtos) metodus. Programuojant skaitinius šaknų nustatymo metodus išmokstama parinkti pradinius sprendinių artinius (intervalus), skaičiavimo pabaigos sąlygas, vizualizuoti algoritmų vykdymo eigą. Gautų sprendinių patikrinimui naudojamos išorinių išteklių funkcijos.

1 Dalis: Netiesinių lygčių sprendimas

Netiesinių lygčių sprendimas

1 dalis (5 balai). Išspręskite netiesines lygtis (1 ir 2 lentelės), kai lygties funkcija yra daugianaris $f(x) = 0$ ir transcendentinė funkcija $g(x) = 0$.

Variantas: 9

Lygtys:

Varianto Nr.	Daugianariai $f(x)$	Funkcijos g(x)	Metodai[1]
9	$0.48x^5 + 1.71x^4 - 0.67x^3 - 4.86x^2 - 1.33x + 1.50$	$e^{-x}\sin(x^2) + 0{,}001; 5 \leq x \leq 10$	2, 3

Pirma dalis

1. Nustatykite daugianario $f(x)$ šaknų intervalą, taikydami „grubų“ ir „tikslesnį“ įverčius. Grafiškai pavaizduokite daugianarį tokiame intervale, kad matytųsi abu įverčiai. Funkciją $g(x)$ grafiškai pavaizduokite užduotyje nurodytame intervale. Esant poreikiui, grafikų ašis pakeiskite taip, kad būtų aiškiai matomos funkcijų šaknys;

$f(x) = 0,48x^5 + 1,71x^4 - 0,67x^3 - 4,86x^2 - 1,33x + 1,50$

$a_5 = 0,48$ $\quad |x| < 1 + \frac{\max|a_i|}{a_n} = R$

$a_4 = 1,71$

$a_3 = -0,67$ $\quad x = 1 + \frac{4,87}{0,48} = 20,48 \approx 20,5$

$a_2 = -4,87$

$a_1 = -1,33$ $\quad$ grubus įvertis $(-20,5 ; 20,5)$

$a_0 = 1,50$

Tikslus įvertis

$0,48x^5 + 1,71x^4 - 0,67x^3 - 4,86x^2 - 1,33x + 1,50$

$R_{teig} = 1 + \sqrt[k]{\frac{B}{a_n}}$ $\quad k = n - \max(i, a_i < 0)$

$B = \max|a_i|$ $\quad k = 5 - 3 = 2$

$B = 4,87$

$R_{teig} = 1 + \sqrt[3]{\frac{4,87}{0,48}} \approx 4,19$

R_{neig}, reik $x \to -x$

$-0,48x^5 - 1,71x^4 + 0,67x^3 + 4,86x^2 + 1,33x + 1,50 \quad (-1)$

$+0,48x^5 + 1,71x^4 - 0,67x^3 - 4,86x^2 - 1,33x - 1,50$

$a_5 = 0,48$ $\quad k = 5 - 3 = 2$

$a_4 = 1,71$

$a_3 = -0,67$ $\quad R_{neig} = 1 + \frac{4,87}{0,48} \approx -11,14$

$a_2 = -4,86$

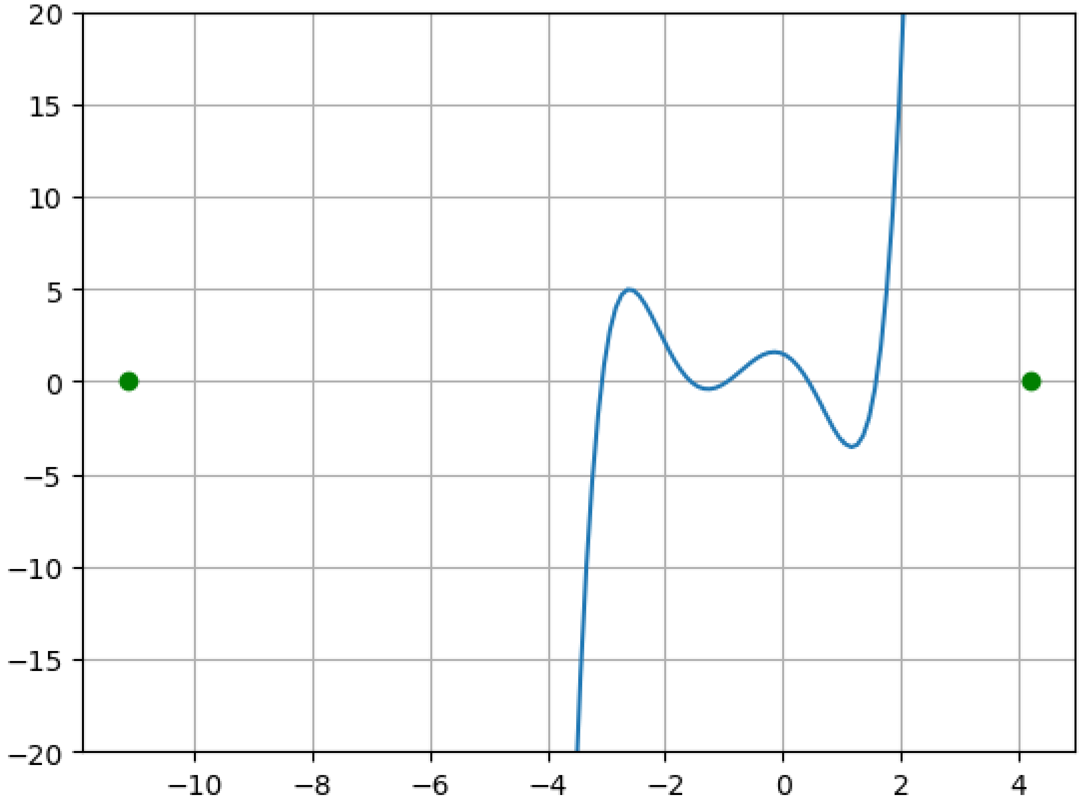
$a_1 = -1,33$ $\quad$ Tikslaus įverčio rezultatas $(-11,14 ; 4,19)$

$a_0 = -1,50$

f(x) Braižymo kodas:

```
import numpy as np
import math
import sympy
from matplotlib import pyplot as plt

def f(x):
    return 0.48*x**5+1.71*x**4-0.67*x**3-4.86*x**2-1.33*x+1.50
def df(x):
    return 5*0.48*x**4+1.71*4*x**3-0.67*3*x**2-4.86*2*x**1-1.33
def g(x):
    return np.exp(-x)*np.sin(x**2)+0.001



xmin = -11.14
xmax = 4.19
x = np.arange(xmin, xmax, 0.1)
plt.plot(x, f(x))
plt.plot(xmin, 0, 'go')
plt.plot(xmax, 0, 'go')
plt.grid()
plt.ylim([-20, 20])
```

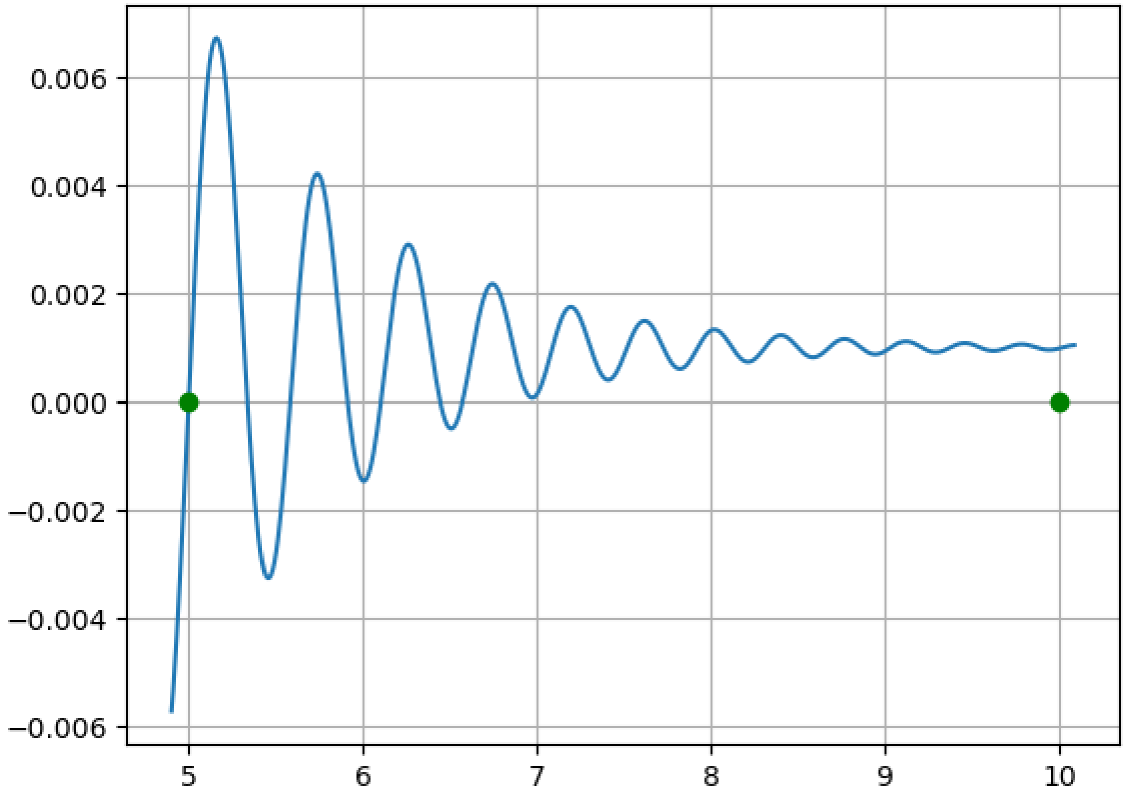
f(x) braižytas rezultatas:



g(x) Braižymo kodas:

```
import numpy as np
import math
import sympy
from matplotlib import pyplot as plt

def f(x):
   return 0.48*x**5+1.71*x**4-0.67*x**3-4.86*x**2-1.33*x+1.50
def df(x):
   return 5*0.48*x**4+1.71*4*x**3-0.67*3*x**2-4.86*2*x**1-1.33
def g(x):
   return np.exp(-x)*np.sin(x**2)+0.001



xmin = 4.9
xmax = 10.1
x = np.arange(4.9, 10.1, 0.01)
plt.plot(x, g(x))
plt.plot(5, 0, 'go')
plt.plot(10, 0, 'go')
plt.grid()
```

g(x) braižytas rezultatas:



Antra dalis

2. Naudodami skenavimo algoritmą su nekintančiu skenavimo žingsniu raskite šaknų atskyrimo intervalus. Daugianariui skenavimo intervalas parenkamas pagal 1 užduoties punkte gautas įverčių reikšmes. Funkcija $g(x)$ skenuojama užduotyje nurodytame intervale.

f(x) Intervalo skenavimo kodas:

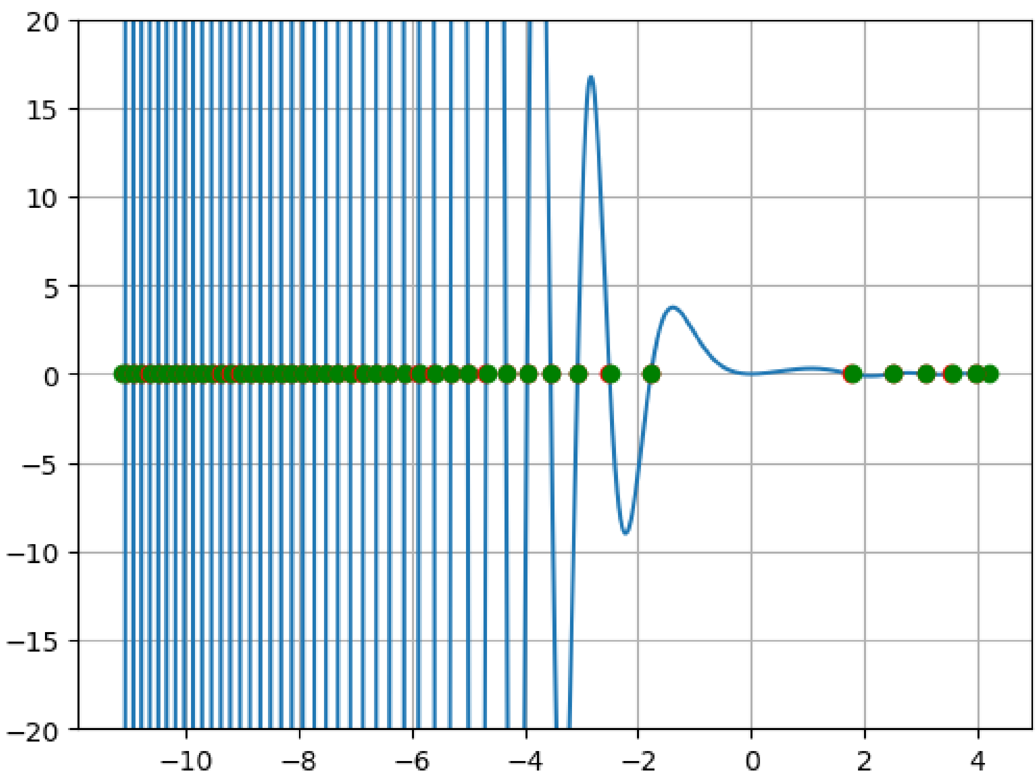
```
import numpy as np
import math
import sympy
from matplotlib import pyplot as plt

def f(x):
   return 0.48*x**5+1.71*x**4-0.67*x**3-4.86*x**2-1.33*x+1.50
def df(x):
   return 5*0.48*x**4+1.71*4*x**3-0.67*3*x**2-4.86*2*x**1-1.33
def g(x):
   return np.exp(-x)*np.sin(x**2)+0.001



xmin = -11.14
xmax = 4.19
x = np.arange(xmin, xmax, 0.1)
plt.plot(x, f(x))
plt.plot(xmin, 0, 'go')
plt.plot(xmax, 0, 'go')
plt.grid()
plt.ylim([-20, 20])

x = xmin
dx = 0.005
while x < (xmax + dx):
   if( np.sign(f(x)) != np.sign(f(x+dx))):
      print("intervalas:")
      print(x)
      print(x+dx)
      plt.plot(x, 0, 'ro')
      plt.plot(x+dx, 0, 'go')
   x += dx
```

Gautas rezultatas:

```
intervalas:
-11.06999999999999
-11.064999999999989
intervalas:
-10.929999999999968
-10.924999999999967
intervalas:
-10.784999999999945
-10.779999999999944
intervalas:
-10.634999999999922
-10.62999999999992
intervalas:
-10.489999999999899
-10.484999999999898
intervalas:
-10.339999999999876
-10.334999999999875
intervalas:
-10.184999999999851
-10.17999999999985
intervalas:
-10.029999999999827
-10.024999999999826
intervalas:
-9.869999999999802
-9.864999999999801
intervalas:
-9.709999999999777
-9.704999999999776
intervalas:
-9.544999999999751
-9.53999999999975
intervalas:
-9.379999999999725
-9.374999999999725
intervalas:
-9.209999999999699
-9.204999999999698
intervalas:
-9.039999999999672
-9.034999999999672
intervalas:
-8.864999999999645
-8.859999999999644
intervalas:
-8.684999999999617
-8.679999999999616
intervalas:
-8.504999999999589
-8.499999999999588
intervalas:
-8.314999999999559
-8.309999999999558
intervalas:
-8.12499999999953
-8.119999999999528
intervalas:
-7.929999999999512
```

```
-7.924999999999512
intervalas:
-7.729999999999516
-7.7249999999995165
intervalas:
-7.519999999999521
-7.514999999999521
intervalas:
-7.309999999999525
-7.304999999999525
intervalas:
-7.08999999999953
-7.08499999999953
intervalas:
-6.864999999999535
-6.859999999999535
intervalas:
-6.63499999999954
-6.62999999999954
intervalas:
-6.394999999999545
-6.389999999999545
intervalas:
-6.13999999999955
-6.13499999999955
intervalas:
-5.879999999999556
-5.874999999999556
intervalas:
-5.604999999999562
-5.599999999999562
intervalas:
-5.319999999999568
-5.314999999999568
intervalas:
-5.014999999999574
-5.009999999999574
intervalas:
-4.689999999999581
-4.684999999999581
intervalas:
-4.3449999999995885
-4.339999999999589
intervalas:
-3.9649999999995966
-3.9599999999995967
intervalas:
-3.5449999999996056
-3.5399999999996057
intervalas:
-3.0699999999996157
-3.064999999999616
intervalas:
-2.5099999999996276
-2.5049999999996277
intervalas:
-1.7749999999996433
-1.7699999999996434
intervalas:
1.7700000000003253
1.7750000000003252
intervalas:
```

```
2.50000000000031
2.50500000000031
intervalas:
3.070000000000298
3.0750000000002977
intervalas:
3.540000000000288
3.5450000000002877
intervalas:
3.9700000000002786
3.9750000000002785
```



<u>g(x) Intervalo skenavimo kodas:</u>

```
import numpy as np
import math
import sympy
from matplotlib import pyplot as plt

def f(x):
    return 0.48*x**5+1.71*x**4-0.67*x**3-4.86*x**2-1.33*x+1.50
def df(x):
    return 5*0.48*x**4+1.71*4*x**3-0.67*3*x**2-4.86*2*x**1-1.33
def g(x):
    return np.exp(-x)*np.sin(x**2)+0.001



xmin = -11.14
xmax = 4.19
```

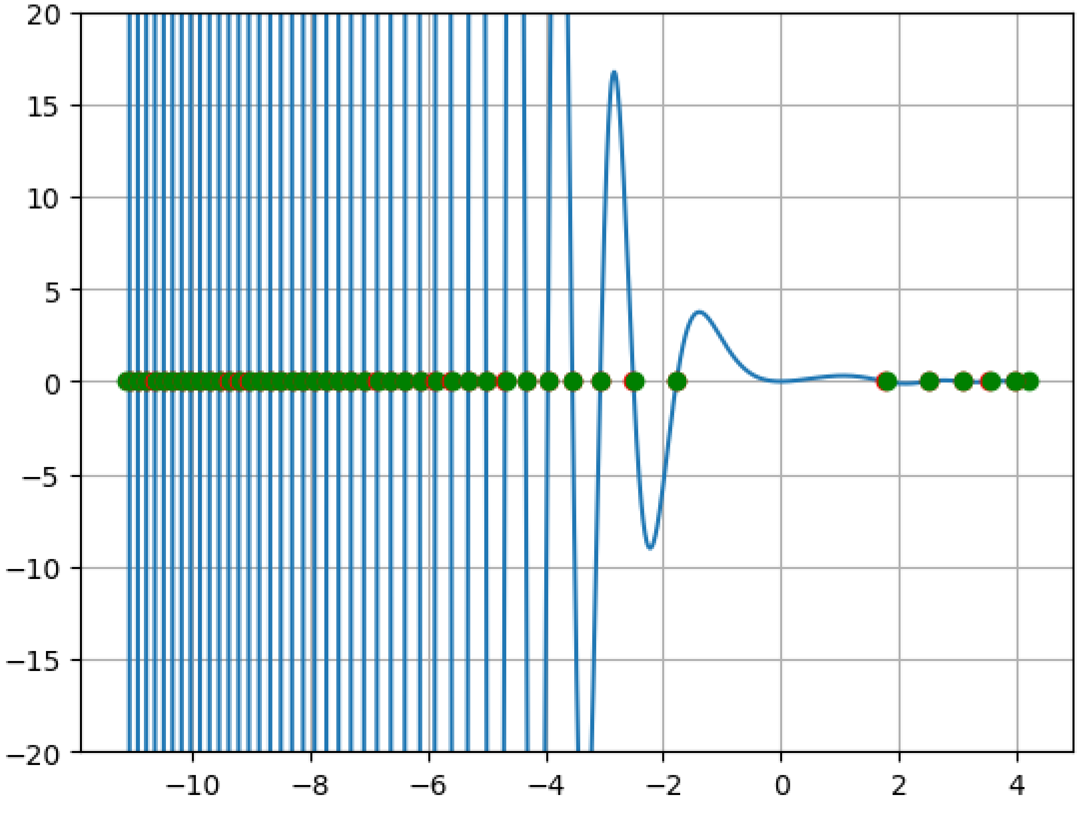
```python
x = np.arange(xmin, xmax, 0.1)
plt.plot(x, f(x))
plt.plot(xmin, 0, 'go')
plt.plot(xmax, 0, 'go')
plt.grid()
plt.ylim([-20, 20])

x = xmin
dx = 0.005
while x < (xmax + dx):
   if( np.sign(g(x)) != np.sign(g(x+dx))):
      print("intervalas:")
      print(x)
      print(x+dx)
      plt.plot(x, 0, 'ro')
      plt.plot(x+dx, 0, 'go')
   x += dx
```

Gautas rezultatas:

```
intervalas:
-11.06999999999999
-11.064999999999989
intervalas:
-10.929999999999968
-10.924999999999967
intervalas:
-10.784999999999945
-10.779999999999944
intervalas:
-10.634999999999922
-10.62999999999992
intervalas:
-10.489999999999899
-10.484999999999898
intervalas:
-10.339999999999876
-10.334999999999875
intervalas:
-10.184999999999851
-10.17999999999985
intervalas:
-10.029999999999827
-10.024999999999826
intervalas:
-9.869999999999802
-9.864999999999801
intervalas:
-9.709999999999777
-9.704999999999776
intervalas:
-9.544999999999751
-9.53999999999975
intervalas:
-9.379999999999725
-9.374999999999725
intervalas:
```

-9.209999999999699
-9.204999999999698
intervalas:
-9.039999999999672
-9.034999999999672
intervalas:
-8.864999999999645
-8.859999999999644
intervalas:
-8.684999999999617
-8.679999999999616
intervalas:
-8.504999999999589
-8.499999999999588
intervalas:
-8.314999999999559
-8.309999999999558
intervalas:
-8.12499999999953
-8.119999999999528
intervalas:
-7.929999999999512
-7.924999999999512
intervalas:
-7.729999999999516
-7.7249999999995165
intervalas:
-7.519999999999521
-7.514999999999521
intervalas:
-7.309999999999525
-7.304999999999525
intervalas:
-7.08999999999953
-7.08499999999953
intervalas:
-6.864999999999535
-6.859999999999535
intervalas:
-6.63499999999954
-6.62999999999954
intervalas:
-6.394999999999545
-6.389999999999545
intervalas:
-6.13999999999955
-6.13499999999955
intervalas:
-5.879999999999556
-5.874999999999556
intervalas:
-5.604999999999562
-5.599999999999562
intervalas:
-5.319999999999568
-5.314999999999568
intervalas:
-5.014999999999574
-5.009999999999574
intervalas:
-4.689999999999581
-4.684999999999581

intervalas:
-4.3449999999995885
-4.339999999999589
intervalas:
-3.9649999999995966
-3.9599999999995967
intervalas:
-3.5449999999996056
-3.5399999999996057
intervalas:
-3.0699999999996157
-3.064999999999616
intervalas:
-2.5099999999996276
-2.5049999999996277
intervalas:
-1.7749999999996433
-1.7699999999996434
intervalas:
1.7700000000003253
1.7750000000003252
intervalas:
2.50000000000031
2.50500000000031
intervalas:
3.070000000000298
3.0750000000002977
intervalas:
3.540000000000288
3.5450000000002877
intervalas:
3.9700000000002786
3.9750000000002785



1. Trečias Punktas

3. Skenavimo metodu atskirtas daugianario ir funkcijos šaknis tikslinkite užduotyje nurodytais metodais. Skaičiavimo scenarijuje turi būti panaudotos skaičiavimų pabaigos sąlygos. Skaičiavimų rezultatus pateikite lentelėje, kurioje nurodykite šaknies tikslinimui naudojamą metodą, pradinį artinį arba atskyrimo intervalą, gautą sprendinį (šaknį), funkcijos reikšmę ties šaknimi, tikslumą, iteracijų skaičių. Palyginkite, kuriuo metodu sprendiniui rasti panaudota mažiau iteracijų;

1 lentelė. Netiesinių lygčių sprendimas. Metodai.

Metodo Nr.	Metodo pavadinimas
1	Stygų
2	Pusiaukirtos
3	Niutono (liestinių)
4	Kvazi-Niutono (kirstinių)

9	$0.48x^5 + 1.71x^4 - 0.67x^3 - 4.86x^2 - 1.33x + 1.50$	$e^{-x}\sin(x^2) + 0{,}001; 5 \leq x \leq 10$	2, 3

Man paskirta 2 ir 3 metodas

Pasirinkau vieną intervalą iš antros dalies gautų intervalų ir pasitelkdamas jį, bandau skaičiuoti. Bandau 50 iteracijų.

f(x) funkcijos pusiaukirtos metodo kodas:

```
#f(x) pusiaukirtos metodas
##############################################################################################
##


import numpy as np
import math
import sympy
from matplotlib import pyplot as plt

def f(x):
    return 0.48*x**5+1.71*x**4-0.67*x**3-4.86*x**2-1.33*x+1.50
def df(x):
    return 5*0.48*x**4+1.71*4*x**3-0.67*3*x**2-4.86*2*x**1-1.33
def g(x):
    return np.exp(-x)*np.sin(x**2)+0.001




xmin = -11.14
xmax = 4.19
x = np.arange(xmin, xmax, 0.01)
```

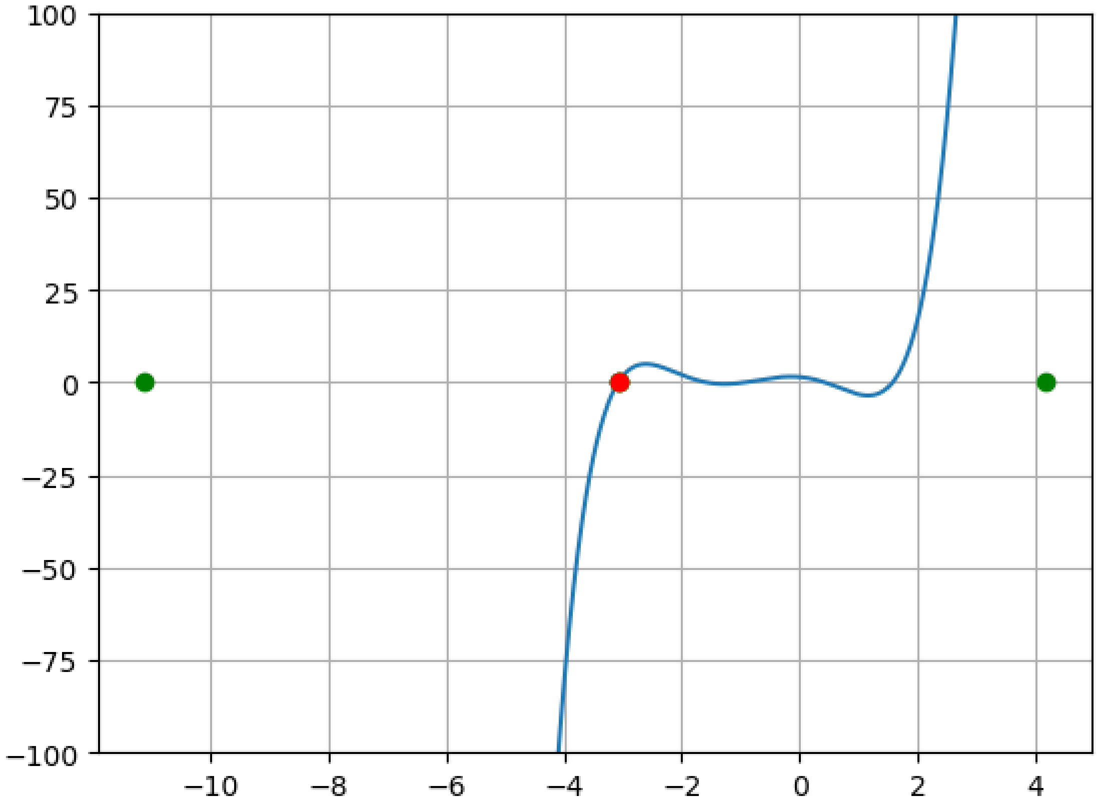
```
plt.plot(x, f(x))
plt.plot(xmin,0,'go')
plt.plot(xmax,0,'go')
plt.grid()
plt.ylim([-100, 100])


x1 = 1.5750000000003295 #čia rašome intervalo pradžią
x2 = 1.5800000000003294 #čia rašome intervalo pabaigą

plt.plot(x1,0,'go')
plt.plot(x2,0,'go')

x = (x1+x2)/2

print("{:>10s} {:>20s} {:>20s}".format("iteration", "root", "function value"))
i = 0
while np.abs(f(x))>1e-15:
    if( i >= 50):
        break
    if(np.sign(f(x1))==np.sign(f(x))):
        x1 = x
    else:
        x2 = x
    x = (x1+x2)/2
    i += 1
    print(" {:>10d} {:>20.10} {:>20.10}".format(i,x,f(x)))
plt.plot(x, f(x), 'ro')
```

f(x) funkcijos pusiaukirtos metodo rezultatas:

intervalas:
-3.0799999999996155
-3.0749999999996156

iteration	root	function value
1	-3.07625	0.006401143875
2	-3.076875	-0.009463404447
3	-3.0765625	-0.001527093112
4	-3.07640625	0.002438034404
5	-3.076484375	0.0004557229354
6	-3.076523437	-0.0005356220116
7	-3.076503906	-3.993376947*10-05
8	-3.076494141	0.000207898525
9	-3.076499023	8.398336329*10-05
10	-3.076501465	2.20250433*10-05
11	-3.076502686	-8.954301486*10-06
12	-3.076502075	6.535386269*10-06
13	-3.07650238	-1.209453722*10-06
14	-3.076502228	2.662967235*10-06
15	-3.076502304	7.267569835*10-07
16	-3.076502342	-2.413483475*10-07
17	-3.076502323	2.427043606*10-07
18	-3.076502333	6.780211947*10-10
19	-3.076502337	-1.203351525*10-07
20	-3.076502335	-5.982860163*10-08
21	-3.076502334	-2.957526934*10-08
22	-3.076502333	-1.444863518*10-08
23	-3.076502333	-6.88530033*10-09
24	-3.076502333	-3.103615143*10-09
25	-3.076502333	-1.212811185*10-09
26	-3.076502333	-2.673772315*10-10
27	-3.076502333	2.053077708*10-10
28	-3.076502333	-3.10818038*10-11

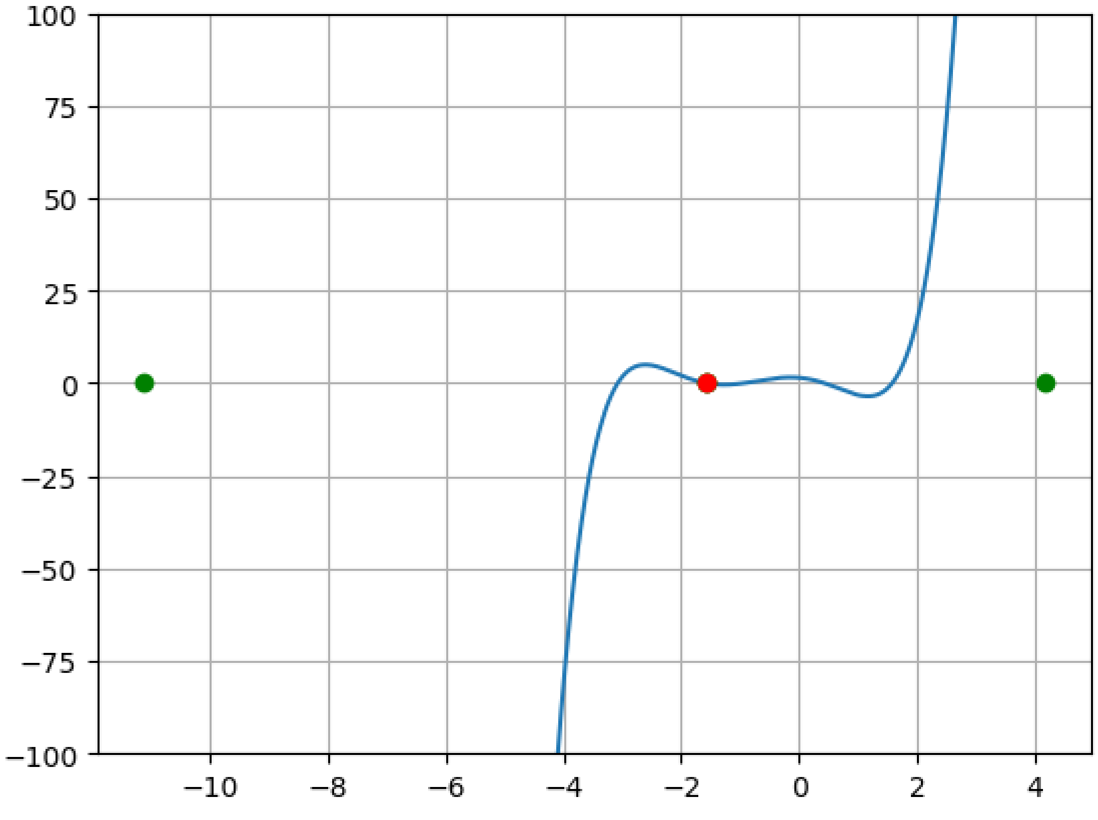
29	-3.076502333	8.714096111*10-11
30	-3.076502333	2.80442336*10-11
31	-3.076502333	-1.486810675*10-12
32	-3.076502333	1.328626098*10-11
33	-3.076502333	5.920597346*10-12
34	-3.076502333	2.177813485*10-12
35	-3.076502333	3.774758284*10-13
36	-3.076502333	-5.622169397*10-13
37	-3.076502333	-1.10134124*10-13
38	-3.076502333	1.474376177*10-13
39	-3.076502333	2.575717417*10-14
40	-3.076502333	-3.463895837*10-14
41	-3.076502333	-2.220446049*10-14
42	-3.076502333	5.329070518*10-15
43	-3.076502333	5.329070518*10-15



intervalas:
-1.5699999999996477
-1.5649999999996478

iteration	root	function value
1	-1.56625	0.001493529495
2	-1.565625	-0.0003012410786
3	-1.5659375	0.0005956544587
4	-1.56578125	0.0001470842482
5	-1.565703125	-7.710902621*10-05
6	-1.565742187	3.497995831*10-05
7	-1.565722656	-2.106644713*10-05
8	-1.565732422	6.956277295*10-06
9	-1.565727539	-7.055204488*10-06
10	-1.56572998	-4.949348797*10-08
11	-1.565731201	3.453384431*10-06
12	-1.565730591	1.701943602*10-06
13	-1.565730286	8.262245879*10-07
14	-1.565730133	3.883654345*10-07
15	-1.565730057	1.694359431*10-07

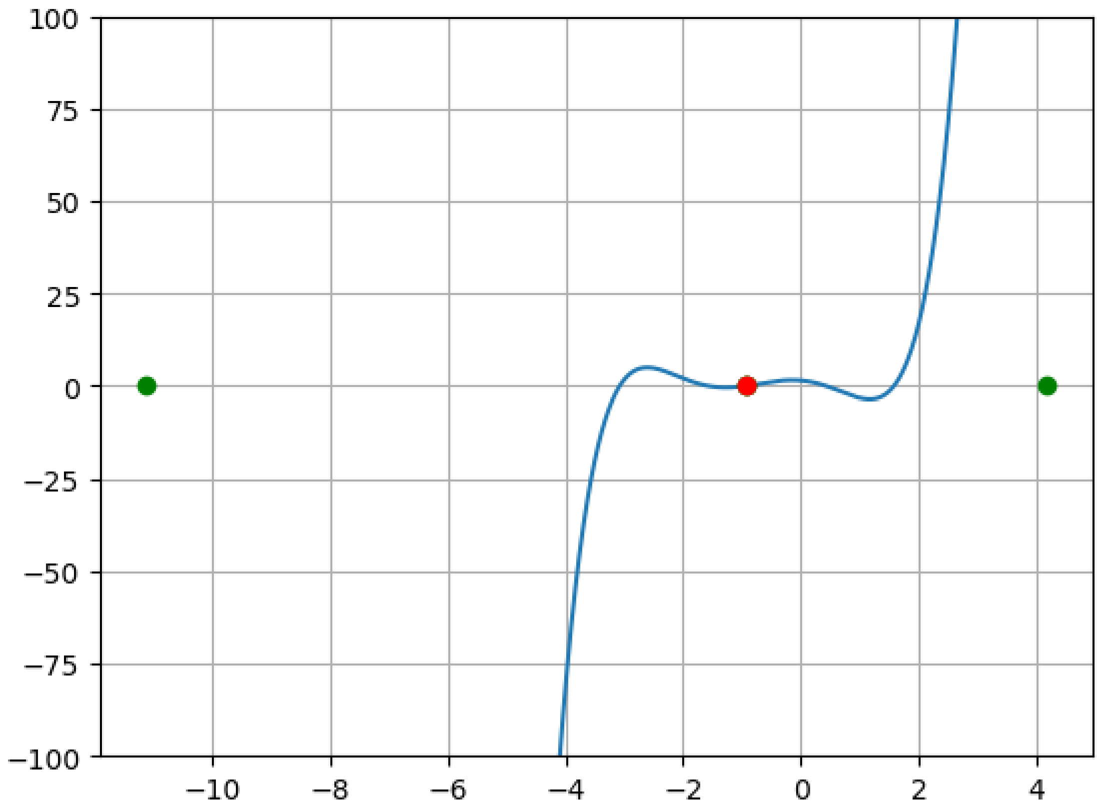
16	-1.565730019	5.997122043*10-08
17	-1.56573	5.238864453*10-09
18	-1.56572999	-2.21273142*10-08
19	-1.565729995	-8.444224875*10-09
20	-1.565729997	-1.602681099*10-09
21	-1.565729998	1.818090123*10-09
22	-1.565729998	1.077018474*10-10
23	-1.565729997	-7.47486073*10-10
24	-1.565729998	-3.198894483*10-10
25	-1.565729998	-1.060938004*10-10
26	-1.565729998	8.055778267*10-13
27	-1.565729998	-5.264322311*10-11
28	-1.565729998	-2.591615811*10-11
29	-1.565729998	-1.255529014*10-11
30	-1.565729998	-5.877520692*10-12
31	-1.565729998	-2.53486121*10-12
32	-1.565729998	-8.637535132*10-13
33	-1.565729998	-2.930988785*10-14
34	-1.565729998	3.881339694*10-13
35	-1.565729998	1.794120408*10-13
36	-1.565729998	7.416289804*10-14
37	-1.565729998	2.309263891*10-14
38	-1.565729998	-8.881784197*10-16



intervalas:
-0.9399999999996597
-0.9349999999996597

iteration	root	function value
1	-0.93875	-0.002015431785
2	-0.938125	-0.0006217228719
3	-0.9378125	7.574480861*10-05
4	-0.93796875	-0.0002730400791
5	-0.937890625	-9.866039298*10-05
6	-0.9378515625	-1.146098111*10-05
7	-0.9378320312	3.214111658*10-05

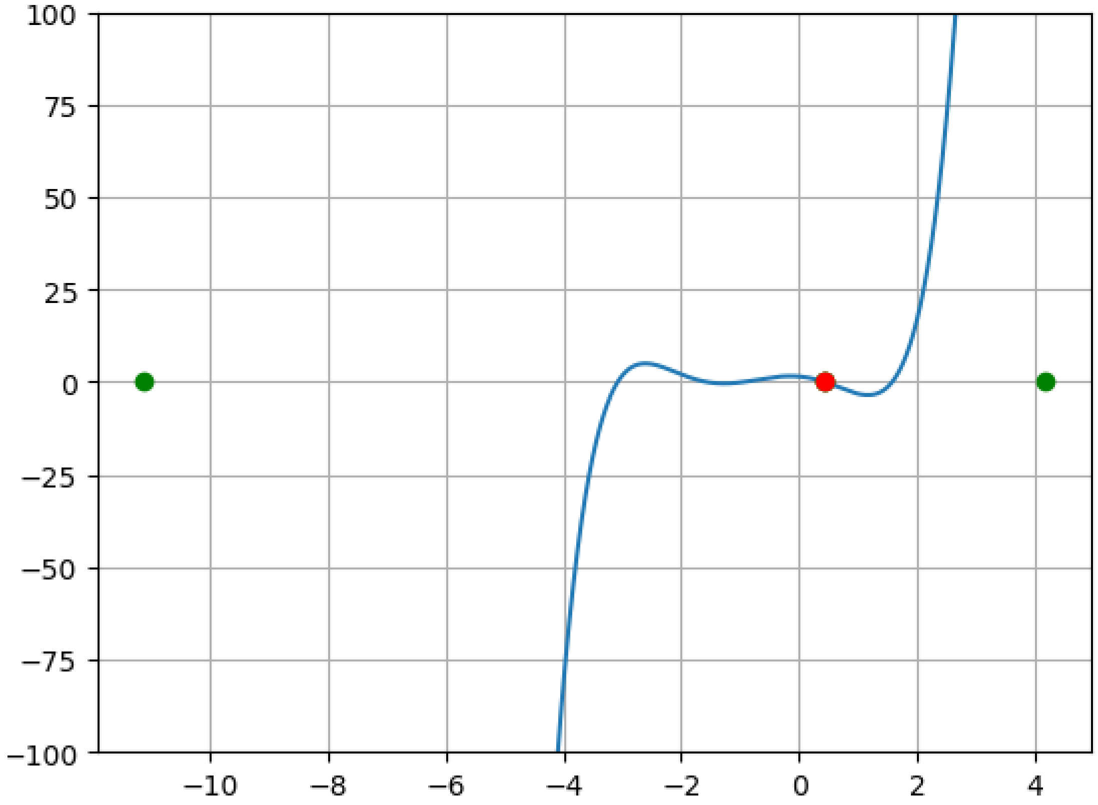
8	-0.9378417969	1.033986843*10-05
9	-0.9378466797	-5.606061659*10-07
10	-0.9378442383	4.889618677*10-06
11	-0.937845459	2.164503141*10-06
12	-0.9378460693	8.019477085*10-07
13	-0.9378463745	1.206705773*10-07
14	-0.9378465271	-2.199678428*10-07
15	-0.9378464508	-4.964864586*10-08
16	-0.9378464127	3.551096239*10-08
17	-0.9378464317	-7.068842622*10-09
18	-0.9378464222	1.422105989*10-08
19	-0.937846427	3.57610852*10-09
20	-0.9378464293	-1.746367495*10-09
21	-0.9378464282	9.148706237*10-10
22	-0.9378464288	-4.157483247*10-10
23	-0.9378464285	2.495612605*10-10
24	-0.9378464286	-8.309308797*10-11
25	-0.9378464285	8.323430833*10-11
26	-0.9378464286	6.972200595*10-14
27	-0.9378464286	-4.151168298*10-11
28	-0.9378464286	-2.072031435*10-11
29	-0.9378464286	-1.032551822*10-11
30	-0.9378464286	-5.127898106*10-12
31	-0.9378464286	-2.528866005*10-12
32	-0.9378464286	-1.229683022*10-12
33	-0.9378464286	-5.797584635*10-13
34	-0.9378464286	-2.555733403*10-13
35	-0.9378464286	-9.237055565*10-14
36	-0.9378464286	-1.065814104*10-14
37	-0.9378464286	2.997602166*10-14
38	-0.9378464286	9.103828802*10-15
39	-0.9378464286	-1.776356839*10-15
40	-0.9378464286	3.552713679*10-15
41	-0.9378464286	1.998401444*10-15
42	-0.9378464286	2.220446049*10-16



intervalas:
0.4350000000003414
0.4400000000003414

iteration	root	function value
1	0.43875	-0.004511945203
2	0.438125	-0.001191424747
3	0.4378125	0.000467849203
4	0.43796875	-0.0003617055203
5	0.437890625	5.309240889*10-05
6	0.4379296875	-0.0001543014144
7	0.4379101563	-5.060321736*10-05
8	0.4379003906	1.244917125*10-06
9	0.4379052734	-2.467906978*10-05
10	0.437902832	-1.171705624*10-05
11	0.4379016113	-5.236064538*10-06
12	0.437901001	-1.995572451*10-06
13	0.4379006958	-3.753273494*10-07
14	0.4379005432	4.34794966*10-07
15	0.4379006195	2.973382807*10-08
16	0.4379006577	-1.72796756*10-07
17	0.4379006386	-7.153146253*10-08
18	0.437900629	-2.08988169*10-08
19	0.4379006243	4.417505695*10-09
20	0.4379006267	-8.240655713*10-09
21	0.4379006255	-1.911574898*10-09
22	0.4379006249	1.252965287*10-09
23	0.4379006252	-3.293050277*10-10
24	0.437900625	4.618303517*10-10
25	0.4379006251	6.6262551*10-11
26	0.4379006251	-1.315207943*10-10
27	0.4379006251	-3.26290106*10-11
28	0.4379006251	1.68167702*10-11
29	0.4379006251	-7.906120203*10-12
30	0.4379006251	4.455324998*10-12
31	0.4379006251	-1.72528658*10-12

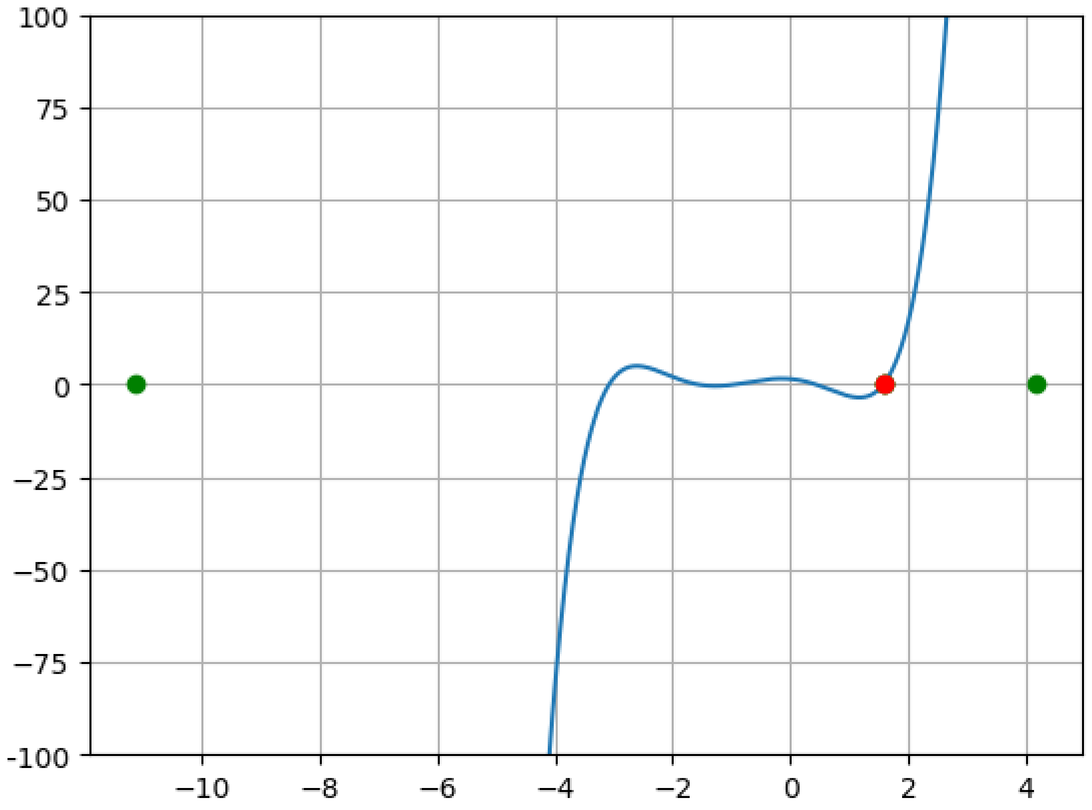
32	0.4379006251	1.365130231*10-12
33	0.4379006251	-1.800781746*10-13
34	0.4379006251	5.924150059*10-13
35	0.4379006251	2.06279438*10-13
36	0.4379006251	1.287858709*10-14
37	0.4379006251	-8.371081606*10-14
38	0.4379006251	-3.552713679*10-14
39	0.4379006251	-1.110223025*10-14
40	0.4379006251	8.881784197*10-16



intervalas:
1.5750000000003295
1.5800000000003294

iteration	root	function value
1	1.57875	-0.01872344738
2	1.579375	-0.006122093965
3	1.5796875	0.0001892648516
4	1.57953125	-0.002967305154
5	1.579609375	-0.001389242832
6	1.579648438	-0.0006000446645
7	1.579667969	-0.0002054038255
8	1.579677734	-8.072966758*10-06
9	1.579682617	9.059507246*10-05
10	1.579680176	4.126083536*10-05
11	1.579678955	1.659387993*10-05
12	1.579678345	4.260442991*10-06
13	1.57967804	-1.90626528*10-06
14	1.579678192	1.177088005*10-06
15	1.579678116	-3.645888453*10-07
16	1.579678154	4.062495296*10-07
17	1.579678135	2.083032813*10-08
18	1.579678125	-1.718792624*10-07
19	1.57967813	-7.552446757*10-08
20	1.579678133	-2.734706905*10-08
21	1.579678134	-3.258373127*10-09

22	1.579678134	8.785978167*10-09
23	1.579678134	2.763805185*10-09
24	1.579678134	-2.47278642*10-10
25	1.579678134	1.258259719*10-09
26	1.579678134	5.054858754*10-10
27	1.579678134	1.291029506*10-10
28	1.579678134	-5.908784573*10-11
29	1.579678134	3.500488788*10-11
30	1.579678134	-1.2046808*10-11
31	1.579678134	1.147837381*10-11
32	1.579678134	-2.833289159*10-13
33	1.579678134	5.598188579*10-12
34	1.579678134	2.660094367*10-12
35	1.579678134	1.19149135*10-12
36	1.579678134	4.534150833*10-13
37	1.579678134	8.659739592*10-14
38	1.579678134	-9.503509091*10-14
39	1.579678134	-1.776356839*10-15
40	1.579678134	4.352074257*10-14
41	1.579678134	2.176037128*10-14
42	1.579678134	6.661338148*10-15
43	1.579678134	3.552713679*10-15
44	1.579678134	-1.776356839*10-15
45	1.579678134	-1.776356839*10-15



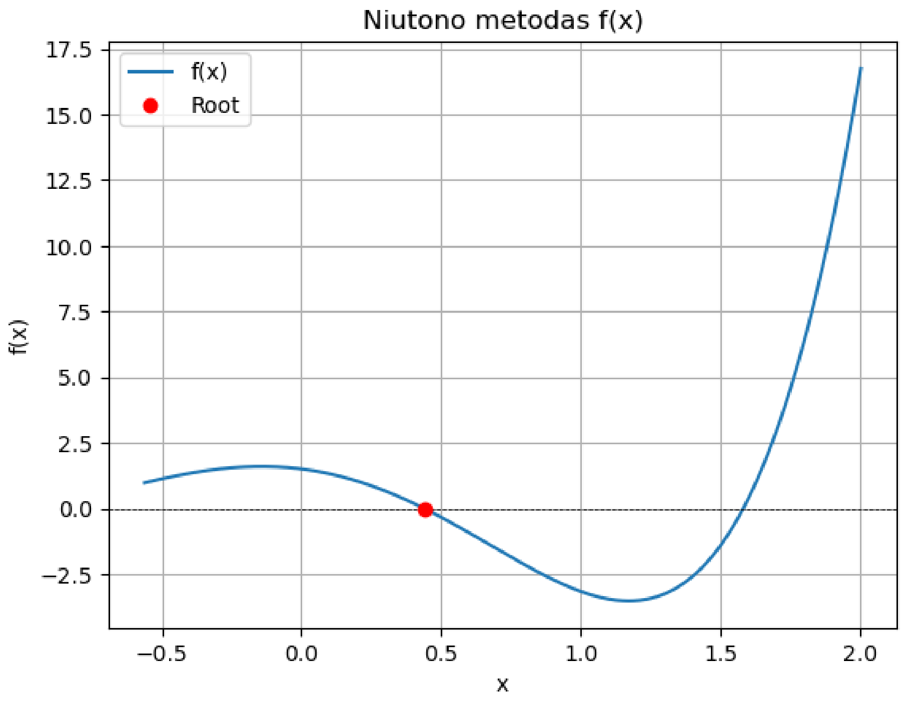
f(x) funkcijos Niutono metodo kodas:

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

def func(x):
    return 0.48 * x**5 + 1.71 * x**4 - 0.67 * x**3 - 4.86 * x**2 - 1.33 * x + 1.50
```

```
def derivative(x):
    # Derivative of the function
    return 5 * 0.48 * x**4 + 4 * 1.71 * x**3 - 3 * 0.67 * x**2 - 2 * 4.86 * x - 1.33

def newton_method(initial_guess, tolerance, max_iterations):
    x = initial_guess
    iteration = 50

    while iteration < max_iterations:
        f_x = func(x)
        f_prime_x = derivative(x)

        if abs(f_prime_x) < 1e-8:
            print("Isvestine per maza")
            return None

        x_new = x - f_x / f_prime_x

        if abs(x_new - x) < tolerance:
            return x_new

        x = x_new
        iteration += 1

    print("Newton's method did not converge after {} iterations.".format(max_iterations))
    return None

initial_guess = 1.0
tolerance = 1e-8
max_iterations = 100

root = newton_method(initial_guess, tolerance, max_iterations)

if root is not None:
    print("Approximate root:", root)

    # Plot the function and marked intervals
    x = np.linspace(min(initial_guess - 1, root - 1), max(initial_guess + 1, root + 1), 1000)
    y = func(x)

    plt.plot(x, y, label="f(x)")
    plt.axhline(0, color='black', linewidth=0.5, linestyle='--')

    plt.plot([root], [0], 'ro', label="Root")

    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.title("Niutono metodas f(x)")
    plt.legend()
    plt.grid(True)
    plt.show()
```

f(x) funkcijos Niutono metodo rezultatas:

```
0.4379006251075934
```



g(x) funkcijos pusiaukirtos metodo kodas:

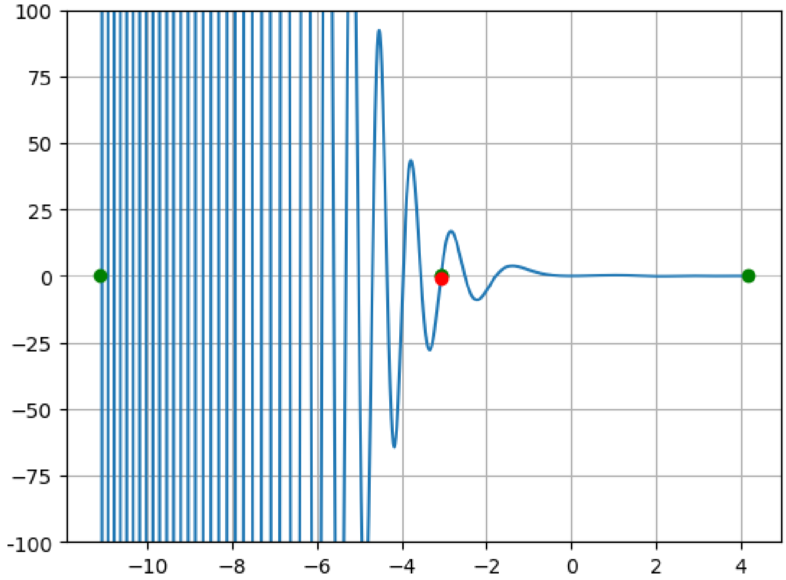
```
#g(x) pusiaukirtos metodas
########################################################################################################################


import numpy as np
import math
import sympy
from matplotlib import pyplot as plt

def f(x):
    return 0.48*x**5+1.71*x**4-0.67*x**3-4.86*x**2-1.33*x+1.50
def df(x):
    return 5*0.48*x**4+1.71*4*x**3-0.67*3*x**2-4.86*2*x**1-1.33
def g(x):
    return np.exp(-x)*np.sin(x**2)+0.001




xmin = -11.14
xmax = 4.19
x = np.arange(xmin, xmax, 0.01)

plt.plot(x, g(x))
plt.plot(xmin,0,'go')
```

```
plt.plot(xmax,0,'go')
plt.grid()
plt.ylim([-100, 100])


x1 = -3.0799999999996155 #čia rašome intervalo pradžią
x2 = -3.0749999999996156 #čia rašome intervalo pabaigą

plt.plot(x1,0,'go')
plt.plot(x2,0,'go')

x = (x1+x2)/2

print("{:>10s} {:>20s} {:>20s}".format("iteration", "root", "function value"))
i = 0
while np.abs(g(x))>1e-15:
    if( i >= 50):
        break
    if(np.sign(g(x1))==np.sign(g(x))):
        x1 = x
    else:
        x2 = x
    x = (x1+x2)/2
    i += 1
    print(" {:>10d} {:>20.10} {:>20.10}".format(i,x,g(x)))
plt.plot(x, g(x), 'ro')
```

g(x) funkcijos pusiaukirtos metodo rezultatas:

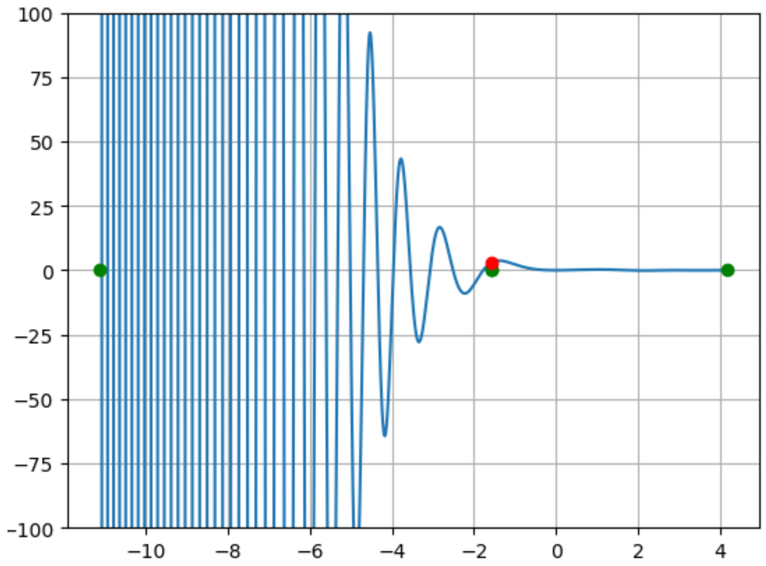
```
intervalas:
-3.0799999999996155
-3.0749999999996156

         1          -3.07625       -0.8341388444
         2         -3.075625       -0.7503787739
         3        -3.0753125       -0.7085399731
         4       -3.07515625       -0.6876309429
         5      -3.075078125       -0.6771790281
         6      -3.075039062       -0.6719537217
         7      -3.075019531       -0.6693412314
         8      -3.075009766        -0.668035027
         9      -3.075004883        -0.667381935
        10      -3.075002441       -0.6670553915
        11      -3.075001221       -0.6668921204
        12       -3.07500061        -0.666810485
        13      -3.075000305       -0.6667696674
        14      -3.075000153       -0.6667492586
        15      -3.075000076       -0.6667390542
        16      -3.075000038        -0.666733952
        17      -3.075000019       -0.6667314009
        18       -3.07500001       -0.6667301253
        19      -3.075000005       -0.6667294875
        20      -3.075000002       -0.6667291686
        21      -3.075000001       -0.6667290092
        22      -3.075000001       -0.6667289295
        23            -3.075       -0.6667288896
        24            -3.075       -0.6667288697
```

25	-3.075	-0.6667288597
26	-3.075	-0.6667288547
27	-3.075	-0.6667288522
28	-3.075	-0.666728851
29	-3.075	-0.6667288504
30	-3.075	-0.6667288501
31	-3.075	-0.6667288499
32	-3.075	-0.6667288498



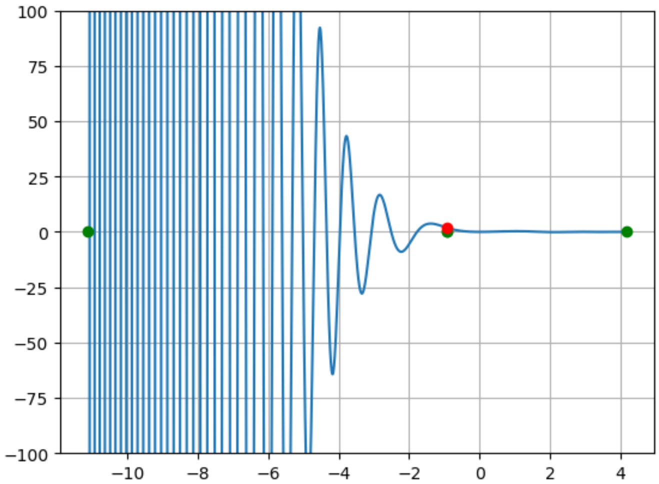
intervalas:
-1.5699999999996477
-1.5649999999996478

1	-1.56625	3.043443354
2	-1.565625	3.048770502
3	-1.5653125	3.051425677
4	-1.56515625	3.052751167
5	-1.565078125	3.053413387
6	-1.565039062	3.053744366
7	-1.565019531	3.053909823
8	-1.565009766	3.053992543
9	-1.565004883	3.054033901
10	-1.565002441	3.05405458
11	-1.565001221	3.054064919
12	-1.56500061	3.054070089
13	-1.565000305	3.054072673
14	-1.565000153	3.054073966
15	-1.565000076	3.054074612
16	-1.565000038	3.054074935
17	-1.565000019	3.054075096
18	-1.56500001	3.054075177
19	-1.565000005	3.054075218
20	-1.565000002	3.054075238
21	-1.565000001	3.054075248
22	-1.565000001	3.054075253
23	-1.565	3.054075255
24	-1.565	3.054075257
25	-1.565	3.054075257
26	-1.565	3.054075258



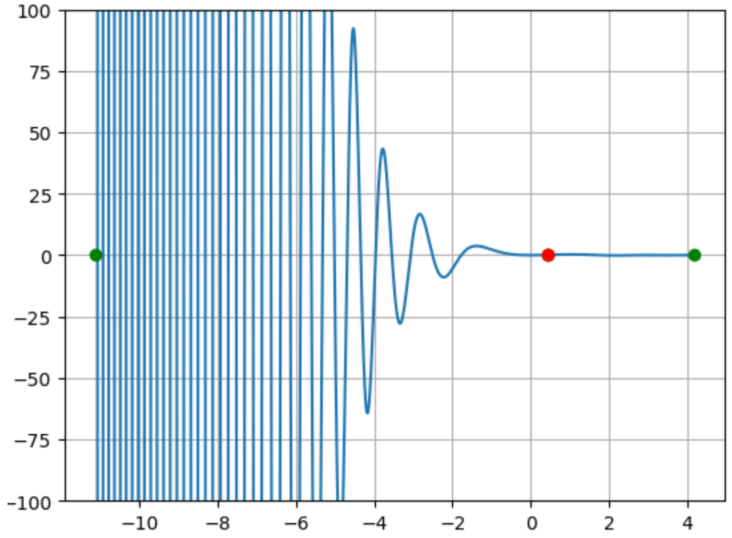
intervalas:
-0.9399999999996597
-0.9349999999996597

1	-0.93625	1.961097069
2	-0.935625	1.957963242
3	-0.9353125	1.956396984
4	-0.93515625	1.95561402
5	-0.935078125	1.955222579
6	-0.9350390625	1.955026869
7	-0.9350195312	1.954929016
8	-0.9350097656	1.954880091
9	-0.9350048828	1.954855628
10	-0.9350024414	1.954843397
11	-0.9350012207	1.954837281
12	-0.9350006104	1.954834223
13	-0.9350003052	1.954832694
14	-0.9350001526	1.95483193
15	-0.9350000763	1.954831548
16	-0.9350000381	1.954831357
17	-0.9350000191	1.954831261
18	-0.9350000095	1.954831213
19	-0.9350000048	1.954831189
20	-0.9350000024	1.954831177
21	-0.9350000012	1.954831171
22	-0.9350000006	1.954831169
23	-0.9350000003	1.954831167
24	-0.9350000001	1.954831166
25	-0.9350000001	1.954831166
26	-0.935	1.954831166



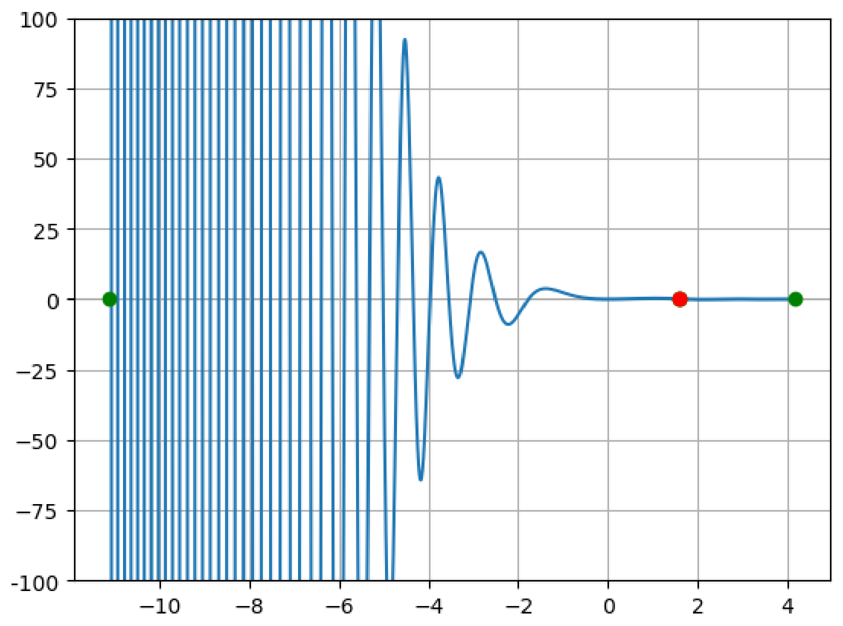
intervalas:
0.4350000000003414
0.4400000000003414

1	0.43875	0.1243678425
2	0.439375	0.1246378963
3	0.4396875	0.1247729499
4	0.43984375	0.1248404833
5	0.439921875	0.1248742516
6	0.4399609375	0.1248911362
7	0.4399804688	0.1248995786
8	0.4399902344	0.1249037999
9	0.4399951172	0.1249059105
10	0.4399975586	0.1249069658
11	0.4399987793	0.1249074934
12	0.4399993896	0.1249077573
13	0.4399996948	0.1249078892
14	0.4399998474	0.1249079551
15	0.4399999237	0.1249079881
16	0.4399999619	0.1249080046
17	0.4399999809	0.1249080128
18	0.4399999905	0.124908017
19	0.4399999952	0.124908019
20	0.4399999976	0.1249080201
21	0.4399999988	0.1249080206
22	0.4399999994	0.1249080208
23	0.4399999997	0.124908021
24	0.4399999999	0.124908021
25	0.4399999999	0.1249080211
26	0.44	0.1249080211



intervalas:
1.5750000000003295
1.5800000000003294

1	1.57875	0.125668183
2	1.579375	0.1252659782
3	1.5796875	0.1250648162
4	1.57984375	0.1249642205
5	1.579921875	0.1249139189
6	1.579960938	0.1248887672
7	1.579980469	0.1248761911
8	1.579990234	0.124869903
9	1.579995117	0.1248667589
10	1.579997559	0.1248651869
11	1.579998779	0.1248644009
12	1.57999939	0.1248640079
13	1.579999695	0.1248638114
14	1.579999847	0.1248637131
15	1.579999924	0.124863664
16	1.579999962	0.1248636394
17	1.579999981	0.1248636271
18	1.57999999	0.124863621
19	1.579999995	0.1248636179
20	1.579999998	0.1248636164
21	1.579999999	0.1248636156
22	1.579999999	0.1248636152
23	1.58	0.1248636151
24	1.58	0.124863615
25	1.58	0.1248636149



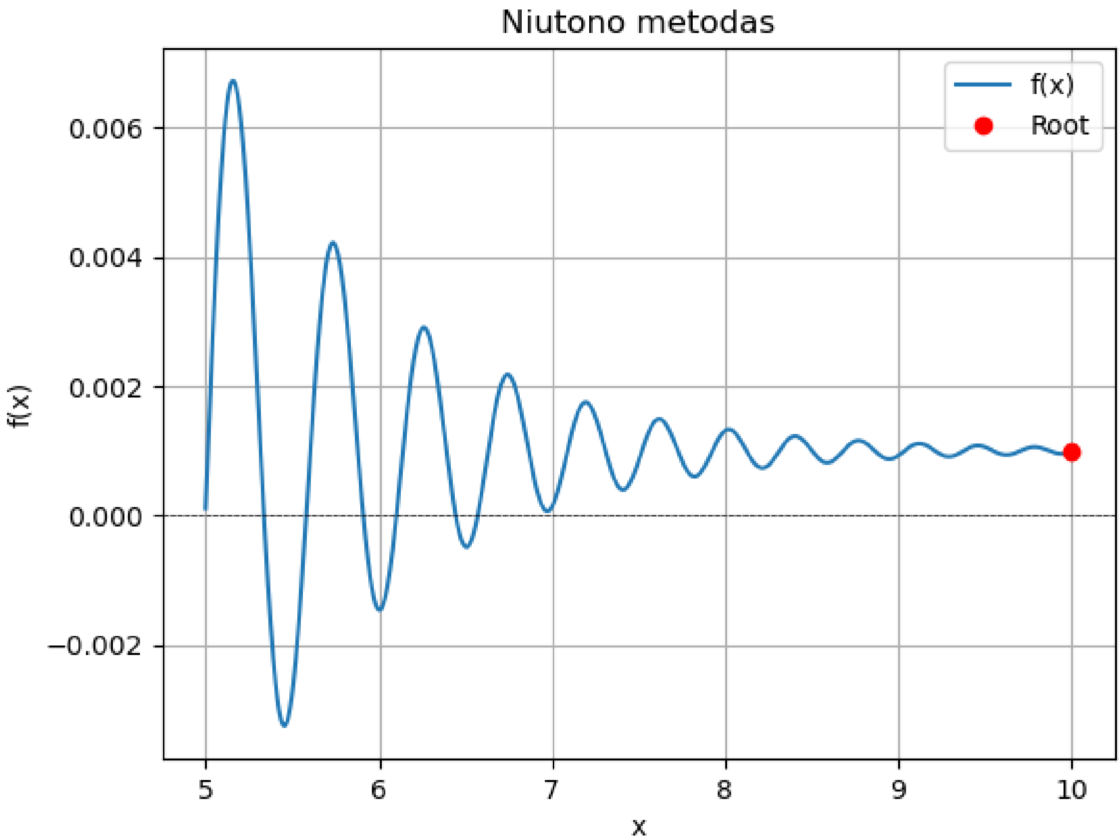
g(x) funkcijos Niutono metodo kodas:

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

def func(x):
    return np.exp(-x) * np.sin(x**2) + 0.001

def derivative(x):
    # Derivative of the function
    return -np.exp(-x) * np.sin(x**2) + 2 * x * np.exp(-x) * np.cos(x**2)

def newton_method(initial_guess, tolerance, max_iterations, interval_start, interval_end):
    x = initial_guess
    iteration = 0

    while iteration < max_iterations:
        f_x = func(x)
        f_prime_x = derivative(x)

        if abs(f_prime_x) < 1e-8:
            print("Derivative too small, Newton's method failed.")
            return None

        x_new = x - f_x / f_prime_x

        if abs(x_new - x) < tolerance or x_new < interval_start or x_new > interval_end:
            if x_new < interval_start:
                x_new = interval_start
            elif x_new > interval_end:
```

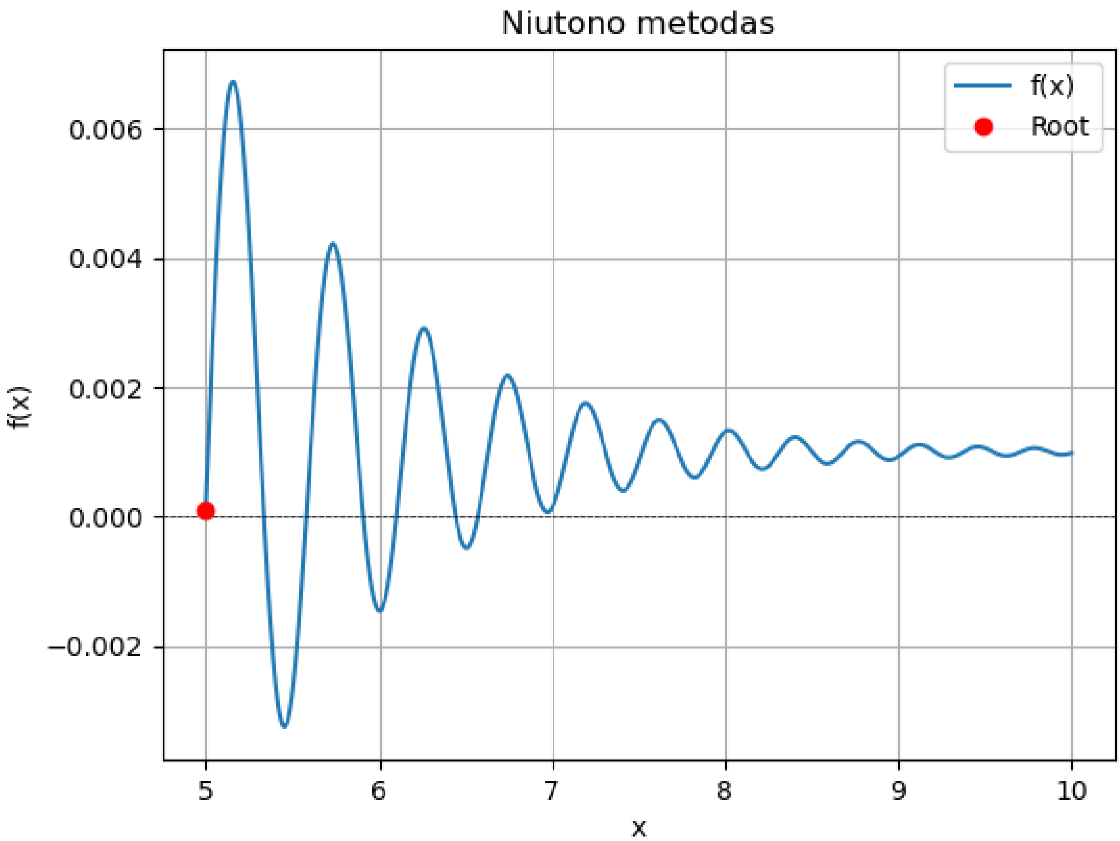
```
            x_new = interval_end
        return x_new

      x = x_new
      iteration += 1

   print("Newton's method did not converge after {} iterations.".format(max_iterations))
   return None

initial_guess = 7.5  # You can adjust the initial guess within the interval
tolerance = 1e-8
max_iterations = 100
interval_start = 5
interval_end = 10

root = newton_method(initial_guess, tolerance, max_iterations, interval_start, interval_end)

if root is not None:
   print("Apytiksle saknis:", root)

   # Plot the function and marked root
   x = np.linspace(interval_start, interval_end, 1000)
   y = func(x)

   plt.plot(x, y, label="f(x)")
   plt.axhline(0, color='black', linewidth=0.5, linestyle='--')

   plt.plot([root], [func(root)], 'ro', label="Root")

   plt.xlabel("x")
   plt.ylabel("f(x)")
   plt.title("Niutono metodas")
   plt.legend()
   plt.grid(True)
   plt.show()
```

g(x) funkcijos Niutono metodo rezultatas:

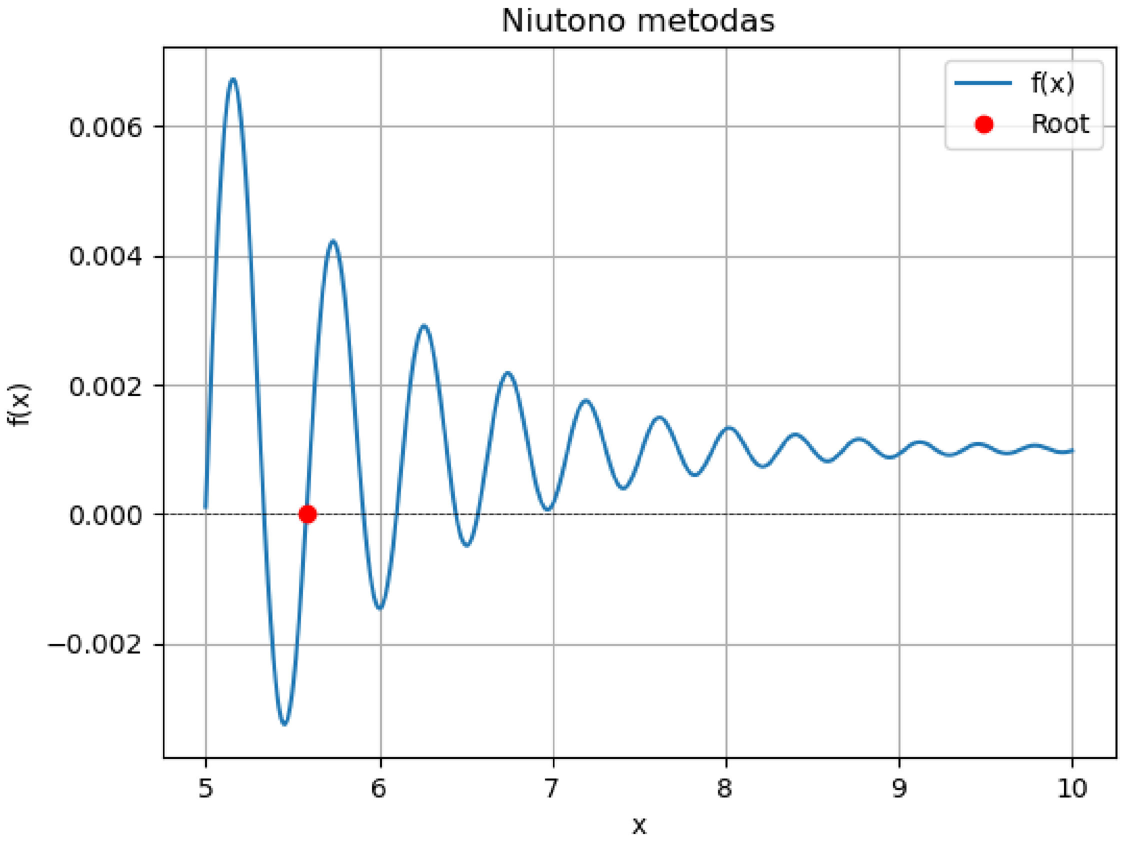
```
Apytiksle saknis: 10
```



Apytiksle saknis: 5



Apytiksle saknis:5.58098358851698



f(x) funkcijos rezultatų lentelė						
Metodas	Intervalas	Šaknis	Funkcijos reikšmė ties šaknimi	Tikslumas	Iteracijų skaičius	Gauta šaknis naudojant
Pusiaukirtos metodas	-3.0799999999996155 -3.0749999999996156	-3.07625	0.006401143875	Iki 100 iteracijų	50	-3.07625
	-1.5699999999996477 -1.5649999999996478	-1.56625	0.001493529495		50	-1.56625
	-0.9399999999996597 -0.9349999999996597	-0.93875	-0.002015431785		50	-0.93875
	0.4350000000003414 0.4400000000003414	0.43875	-0.004511945203		50	0.43875
	1.5750000000003295 1.5800000000003294	1.57875	-0.01872344738		50	1.57875
Niutono metodas		0.43790062510 75934			50	0.437900625 1075934

g(x) funkcijos rezultatų lentelė						
Metodas	Intervalas	Šaknis	Funkcijos reikšmė ties šaknimi	Tikslumas	Iteracijų skaičius	Gauta šaknis naudojant
Pusiaukirtos metodas	-3.0799999999996155 -3.0749999999996156	-3.07625	-0.8341388444	Iki 100 iteracijų / šimtųjų tikslumu	50	-0.8341388444
	-1.5699999999996477 -1.5649999999996478	-1.56625	3.043443354		50	-1.56625
	-0.9399999999996597 -0.9349999999996597	-0.93625	1.961097069		50	-0.93625
	0.4350000000003414 0.4400000000003414	0.43875	0.1243678425		50	0.43875
	1.5750000000003295 1.5800000000003294	1.57875	0.125668183		50	1.57875
Niutono metodas	6	5			50	5
	7.5	10			50	10
	8	6.571631348589032			50	6.571631348589032
	5.5	5.58098358851698			50	5.58098358851698

Žiūrint į rezultatus sprendžiu, kad teisingai padarius, Niutono metodas skaičiuoja tiksliau ir greičiau.

2 Dalis: Teiloro eilutės panaudojimas

3 lentelėje pateiktą funkciją $h(x)$ išskleiskite Teiloro eilute (TE) nurodyto intervalo vidurio taško aplinkoje. Nustatykite TE narių skaičių, su kuriuo visos TE šaknys esančios nurodytame intervale, skiriasi nuo funkcijos $h(x)$ šaknų ne daugiau negu |1e-4|. Tiek pateiktos funkcijos $h(x)$ šaknis, tiek TE šaknis raskite antru iš pirmoje dalyje realizuotų skaitinių metodų (Niutono arba Kvazi-Niutono, priklausomai nuo varianto)

Duota funkcija:

Varianto Nr.	Funkcijos $h(x)$	Nagrinėjamas intervalas

1. Pirma Dalis

Atvaizduoju tarpinius grafikus, kai drauge su pateikta funkcija h(x) nurodytame intervale TE, kai jos narių skaičius yra lygus 3, 4, ir 5.

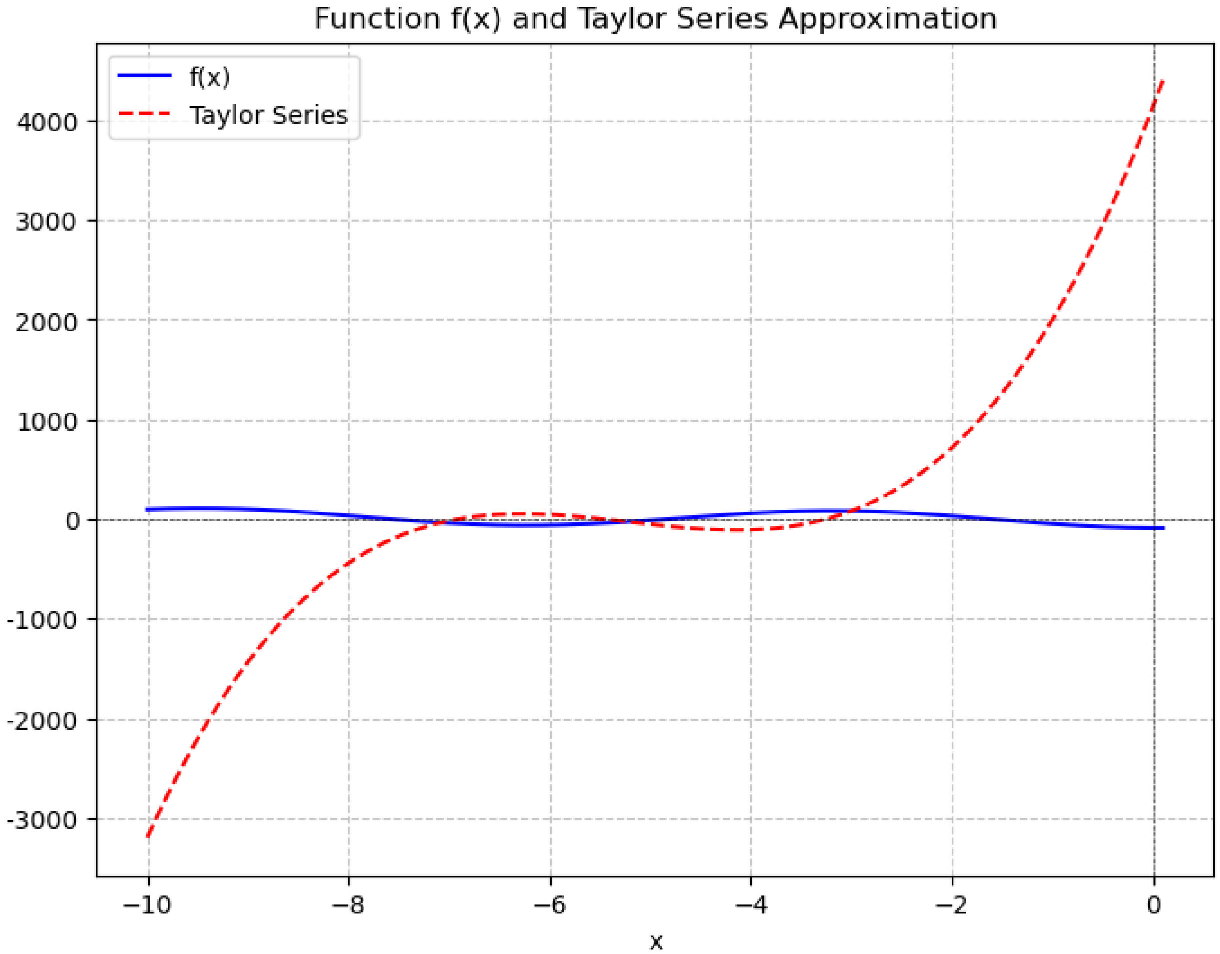
<u>Programos kodas:</u>

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

# Parametrai
x, f, fp, df = sympy.symbols(('x', 'f', 'fp', 'df'))
f = 2 * sympy.cos(x) + 47 * sympy.cos(2*x) +2
x0 = -6
N = 3

# Apskaičiuojame TE
fp = f.subs(x, x0)  # First Taylor series term
for i in range(1, N + 1):
    f = f.diff(x)
    fp = fp + f.subs(x, x0) / math.factorial(i) * (x - x0) ** i

# TE pavertimas į funkciją
taylor_function = sympy.lambdify(x, fp, 'numpy')

x_values = np.linspace(-10, 0.1, 400)
f_values = [-79 * np.cos(val) - 11 - 4 * val for val in x_values]  # Calculate f(x) values
taylor_values = [taylor_function(val) for val in x_values]  # Calculate Taylor series values

plt.figure(figsize=(8, 6))
plt.plot(x_values, f_values, label='f(x)', color='blue')
plt.plot(x_values, taylor_values, label='Taylor Series', color='red', linestyle='--')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Function f(x) and Taylor Series Approximation')
plt.axhline(0, color='black', linewidth=0.5, linestyle='--', alpha=0.7)
```

```
plt.axvline(0, color='black', linewidth=0.5, linestyle='--', alpha=0.7)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()
plt.show()

a=sympy.Poly(fp,x) # daugianaris simboliais
kf=np.array(a.all_coeffs()) # visi koeficientai nuo vyriausio
saknys=np.roots(kf)
print(saknys)
```

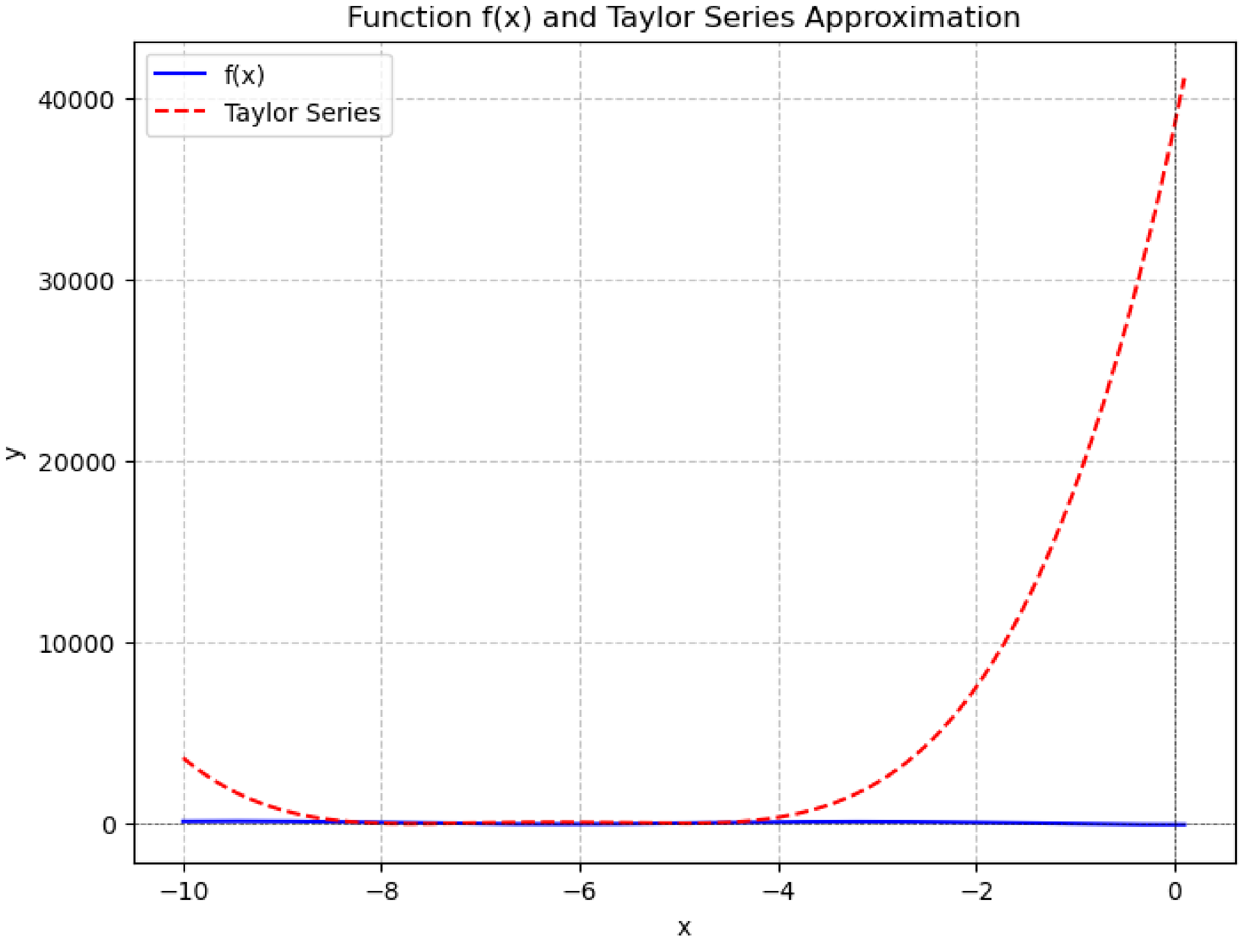
Grafikas kai TE narių skaičius: 3

```
[-6.89925291 -5.47910509 -3.24067189]
```

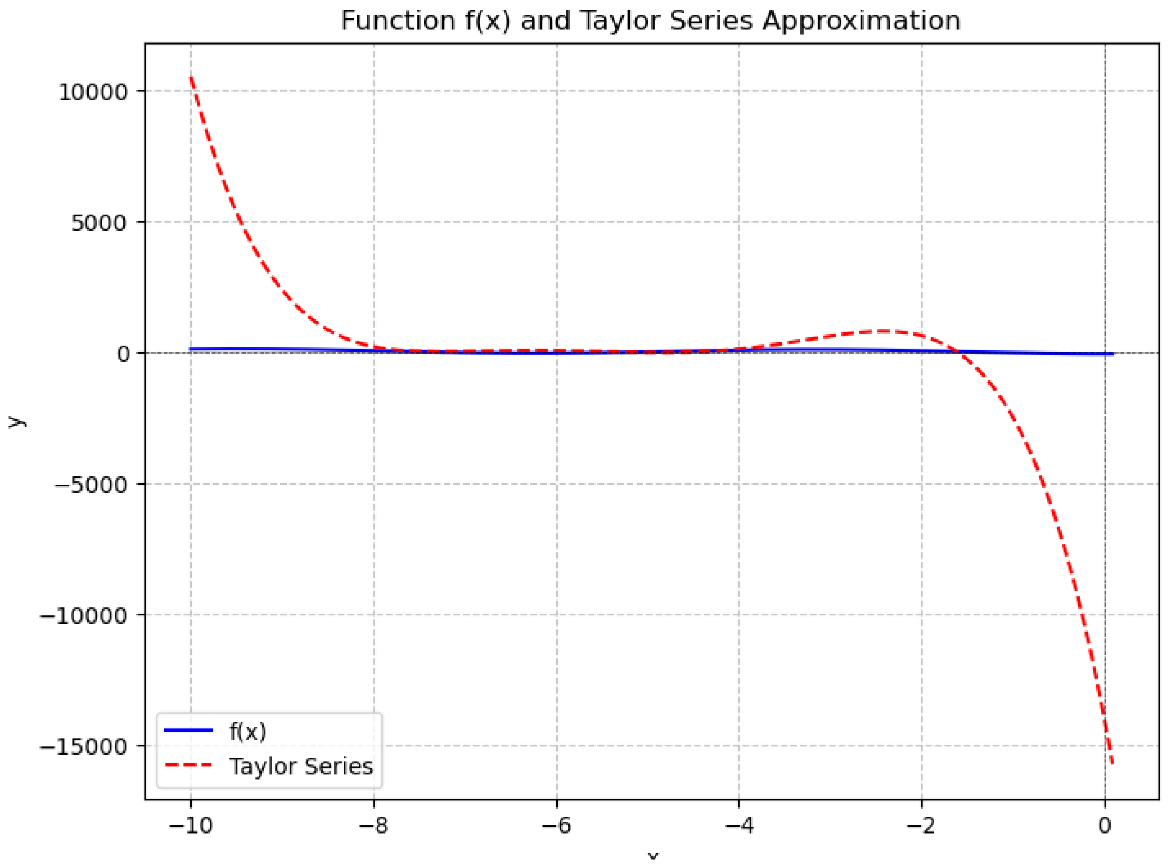


Grafikas kai TE narių skaičius: 4

[-8.09943304 -7.06708775 -5.45784143 -4.64703271]



Grafikas kai TE narių skaičius: 4


Function f(x) and Taylor Series Approximation
10000
5000
0
−5000
−10000
−15000
y
−10
−8
−6
−4
−2
0
x
f(x)
Taylor Series