Kauno technologijos universitetas

Informatikos fakultetas

P170B115 Skaitiniai metodai ir algoritmai

1 projektinė užduotis. Netiesinių lygčių sprendimas.

Nedas Liaudanskis

Studentas

doc. Čalnerytė Dalia

Dėstytoja

KAUNAS, 2023

Turinys

Įvadas

Laboratorinį darbą sudaro 2 dalys: 1. Netiesinių lygčių sprendimas; 2. Teiloro eilutės naudojimas ir apskaičiavimas. Atliekant darbą naudojamės Python aplinka internete. Visus skaičiavimus atliekame naudojant skaitinius metodus per programavimą, nes šie skaičiavimai fiziškai užtruktų labai ilgą laiką.

1 Dalis: Netiesinių lygčių sprendimas

Užduotis: Išspręsti netiesines lygtis f(x) ir g(x), kai lygties funkcija yra daugianaris f(x) = 0 ir transcendentinė funkcija g(x) = 0.

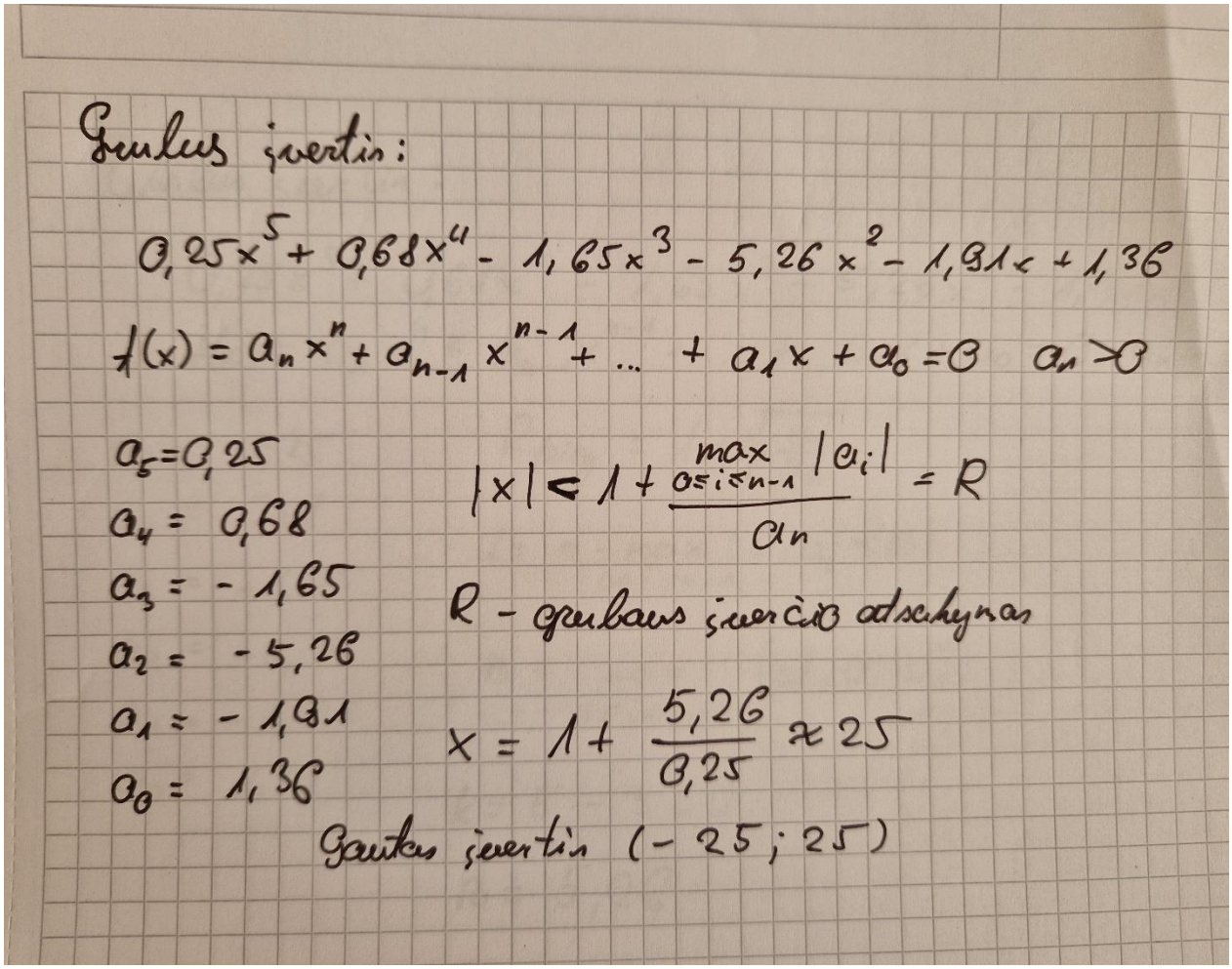
Variantas: 10

Lygtys:

Varianto Nr.	Daugianariai $f(x)$	Funkcijos g(x)	Metodai[1]
10	$0.25x^5 + 0.68x^4 - 1.65x^3 - 5.26x^2 - 1.91x + 1.36$	$e^{-x}\cos(x)\sin(x^2-1); 7 \le x \le 8$	1, 4

1. Pirmas Punktas

Pirmiausia reikia nustatyti daugianario f(x) šaknų intervalus, kad galėtume lokalizuoti šaknis, taikant „grubų“ ir „tikslesnį“ įverčius. Abu įverčius skaičiuojame sąsiuvinio lape ir gautus atsakymus naudojame braižant grafiką, kuris prasideda ir baigiasi gautame intervale. g(x) funkcijai intervalai yra duodami tai skaičiuoti nieko nereikės.



Tikslus įvertis:

$$0{,}25x^5 + 0{,}68x^4 - 1{,}65x^3 - 5{,}26x^2 - 1{,}91x + 1{,}36$$

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = 0 \quad a_n > 0$$

$a_5 = 0{,}25$
$a_4 = 0{,}68$
$a_3 = -1{,}65$
$a_2 = -5{,}26$
$a_1 = -1{,}91$
$a_0 = 1{,}36$

$$R_{teig} = 1 + \sqrt[k]{\frac{B}{a_n}}$$

$$L = n - \max_{0 \le i \le n-1} (i, a_i < 0)$$

$$B = \max_{0 \le i \le n-1} (|a_i|, a_i < 0)$$

$k = 5 - 3 = 2$

$B = 5{,}26$

$$R_{teig} = 1 + \sqrt[2]{\frac{5{,}26}{0{,}25}} \approx 6 //$$

Keičiant vietomis R_{neig} reikia $x \Rightarrow -x$

$$-0{,}25x^5 + 0{,}68x^4 + 1{,}65x^3 - 5{,}26x^2 + 1{,}91x + 1{,}36 \mid \cdot(-1)$$

$$0{,}25x^5 - 0{,}68x^4 - 1{,}65x^2 + 5{,}26x^2 - 1{,}91x - 1{,}36$$

$a_5 = 0{,}25$
$a_4 = -0{,}68$
$a_3 = -1{,}65$
$a_2 = 5{,}26$
$a_1 = -1{,}91$
$a_0 = -1{,}36$

$k = 5 - 4 = 1$

$B = 1{,}91$

$$R_{neig} = 1 + \frac{1{,}91}{0{,}25} \approx -8{,}6 //$$

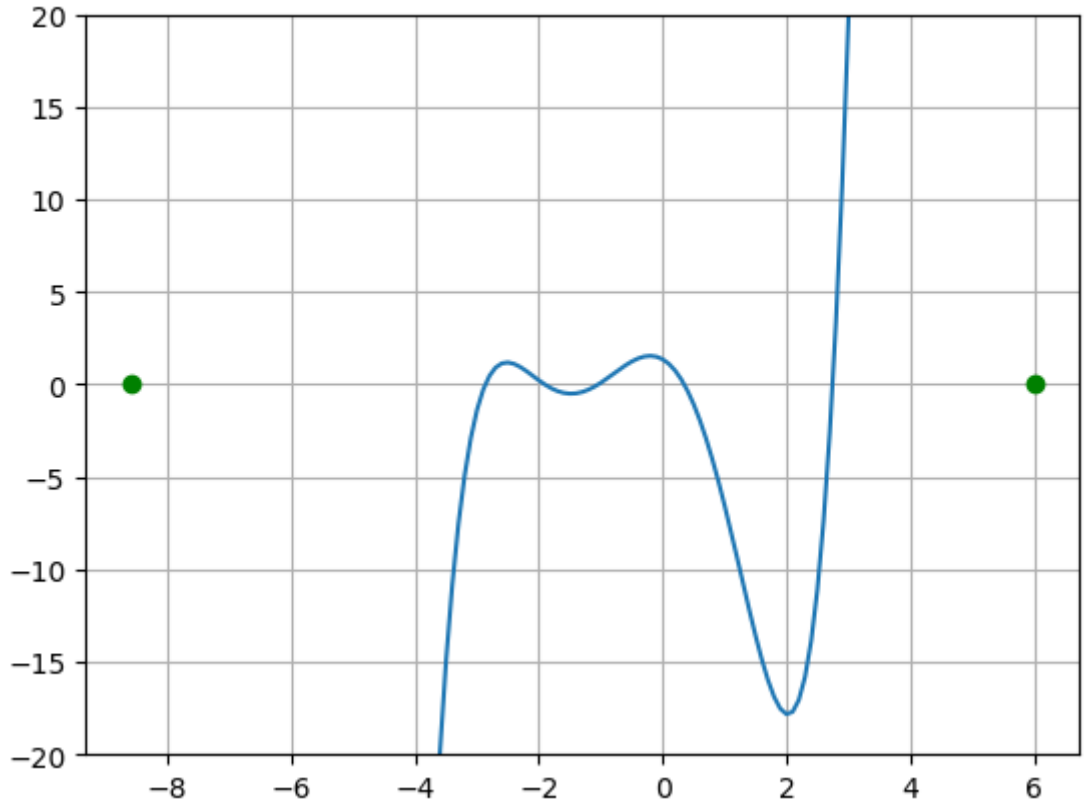
Tikslaus įvertis intervalas $(-8{,}6, 6)$ //

f(x) Braižymo kodas:

```
def f(x):
    return 0.25*x**5+0.68*x**4-1.65*x**3-5.26*x**2-1.91*x+1.36
def df(x):
    return 5*0.25*x**4+0.68*4*x**3-1.65*3*x**2-5.26*2*x**1-1.91
def g(x):
    return np.exp(-x)*np.cos(x)*np.sin((x**2)-1)



xmin = -8.6
xmax = 6
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) Rezultatas:

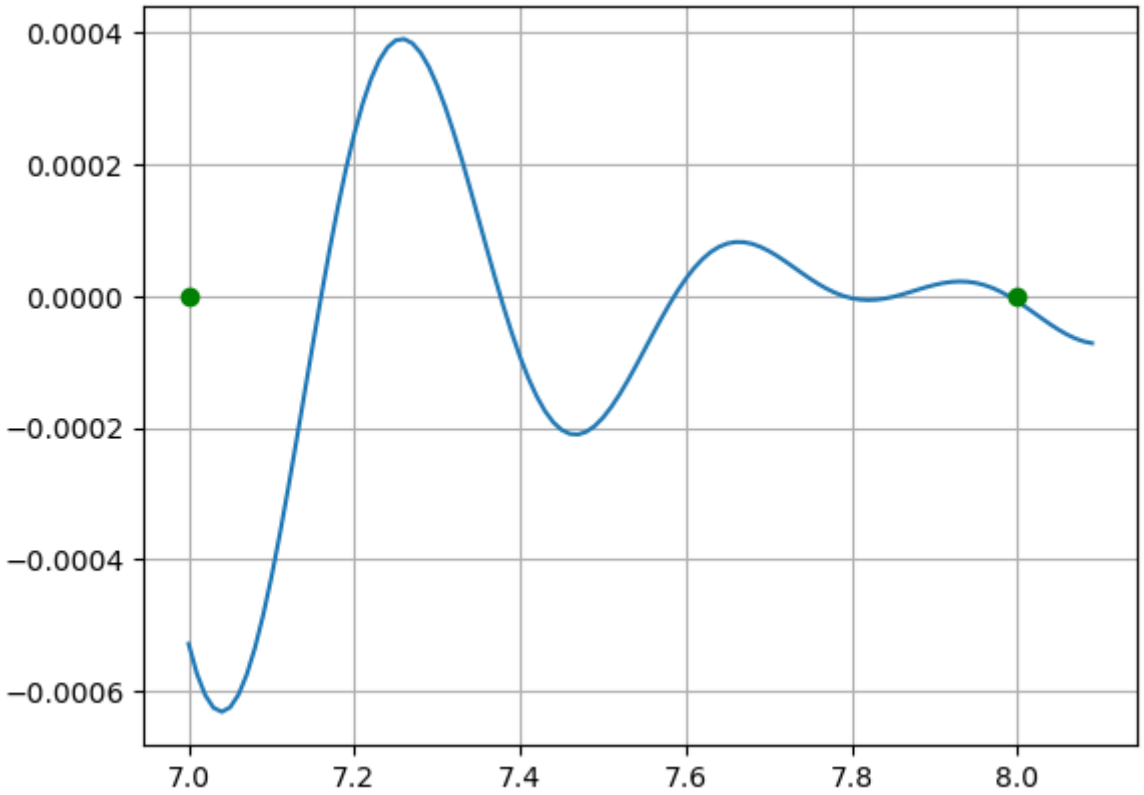
```
(-20.0, 20.0)
```



g(x) Braižymo kodas:

```
def g(x):
    return np.exp(-x)*np.cos(x)*np.sin((x**2)-1)


x = np.arange(7, 8.1, 0.01)
#print(x)
plt.plot(x, g(x))
plt.plot(7, 0, 'go')
plt.plot(8, 0, 'go')
plt.grid()
#plt.ylim([-20, 20])
```

g(x) Rezultatai:



2. Antras Punktas

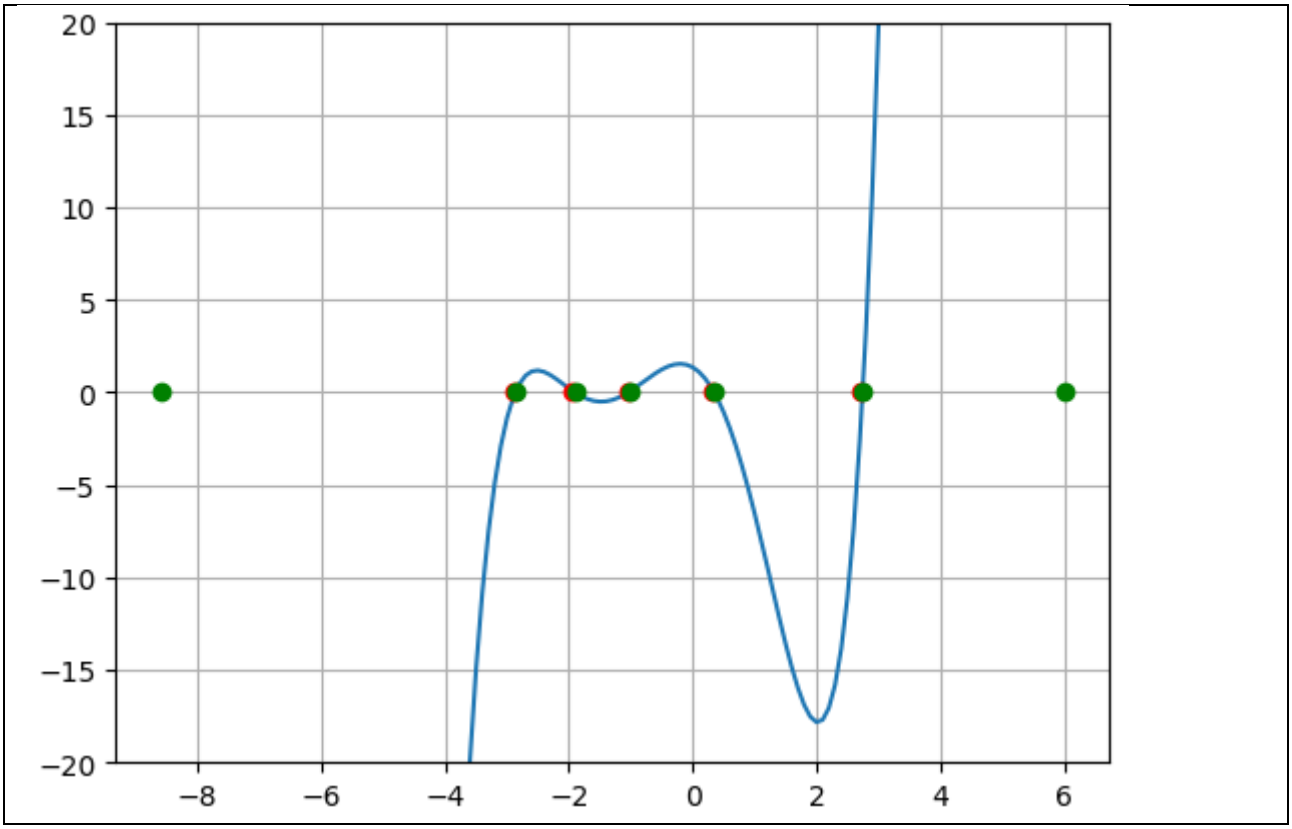
Jau turiu brėžinį, tačiau dar nežinome pagrindinės užduoties dalies, duotos funkcijos šaknų. Norint surasti šaknis reikia pirma surasti šaknų intervalus naudojant skenavimo algoritmą, kuris skenuojant tam tikru žingsniu. Šis algoritmas priklausomai nuo žingsnio randa funkcijos pasikeitimą iš teigiamo į neigiamą, ta vieta yra skaitoma kaip šaknies intervalas.

f(x) Intervalo skenavimo kodas:

```
xmin = -8.6
xmax = 6
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 = 0.05
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:
-2.90000000000001
-2.8500000000000103
intervalas:
-1.9500000000000133
-1.9000000000000132
intervalas:
-1.0500000000000125
-1.0000000000000124
intervalas:
0.29999999999998794
0.34999999999998793
intervalas:
2.699999999999987
2.7499999999999867
```



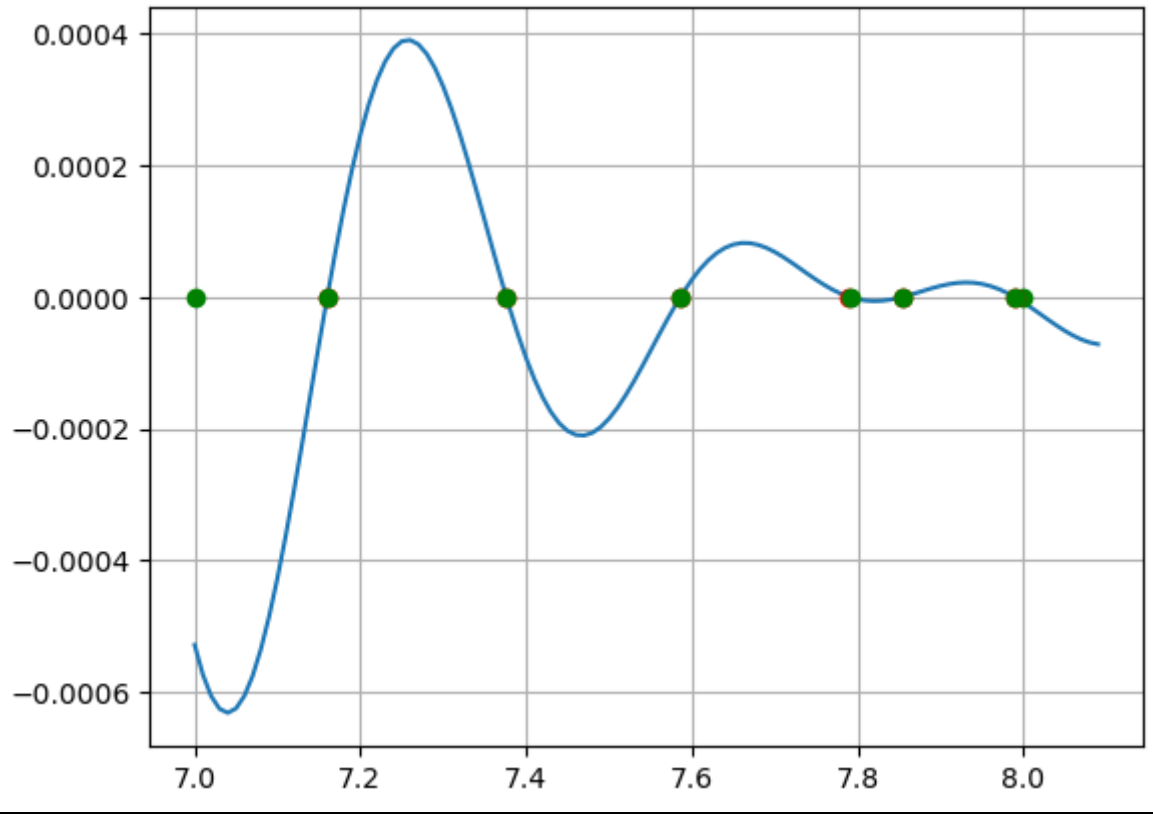
g(x) Intervalo skenavimo kodas:

```
def g(x):
    return np.exp(-x)*np.cos(x)*np.sin((x**2)-1)

x = np.arange(7, 8.1, 0.01)
#print(x)
plt.plot(x, g(x))
plt.plot(7, 0, 'go')
plt.plot(8, 0, 'go')
plt.grid()
#plt.ylim([-20, 20])

x = 7
dx = 0.0005
while x < (8 + 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:
7.159499999999912
7.159999999999911
intervalas:
7.375999999999792
```

7.376499999999791
intervalas:
7.585999999999675
7.586499999999675
intervalas:
7.789999999999562
7.790499999999562
intervalas:
7.853499999999527
7.853999999999527
intervalas:
7.988999999999452
7.989499999999452



3. Trečias Punktas

Skenavimo metodu atskirtas funkcijas tiksliname naudojant užduotyje pateiktus metodus. Skaičiavimo scenarijuje turi būti panaudotos skaičiavimų pabaigos sąlygos. Visus rezultatus pateikiu lentele, kurioje nurodau šaknies tikslinimui naudotą metodą, pradinį artinį arba atkyrimo intervalą, gautą sprendinį(šaknį), funkcijos reikšmę ties šaknimi, tikslumą, iteracijų skaičių. Ir palyginu kuriuo metodu yra naudota mažiau iteracijų.

1 lentelė. Netiesinių lygčių sprendimas. Metodai.

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

10	$0.25x^5 + 0.68x^4 - 1.65x^3 - 5.26x^2 - 1.91x + 1.36$	$e^{-x}\cos(x)\sin(x^2-1); 7 \le x \le 8$	1, 4

Mano variantui priklauso 1,4 metodai. Tai norint išspęsti šią dalį aš naudosiu Stygų ir Kvazi-Niutono algoritmus.

f(x) funkcijos stygų metodo kodas:

```
xmin = -8.6
xmax = 6
#styginis metodas
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])

x1 = -4
x2 = -2
plt.plot(x1, 0, 'go')
plt.plot(x2, 0, 'go')
k = np.abs(f(x1)/f(x2))
x = (x1+k*x2)/(1+k)
#x = (x1+x2)/2
print(" {:>10s} {:>20s} {:>20s}".format("iteration",  "root", "function value"))
i = 0
while np.abs(f(x))>1e-9:
```

$$k = \left|\frac{f(x_n)}{f(x_{n1})}\right| = \frac{x_{mid} - x_n}{x_{n1} - x_{mid}}$$

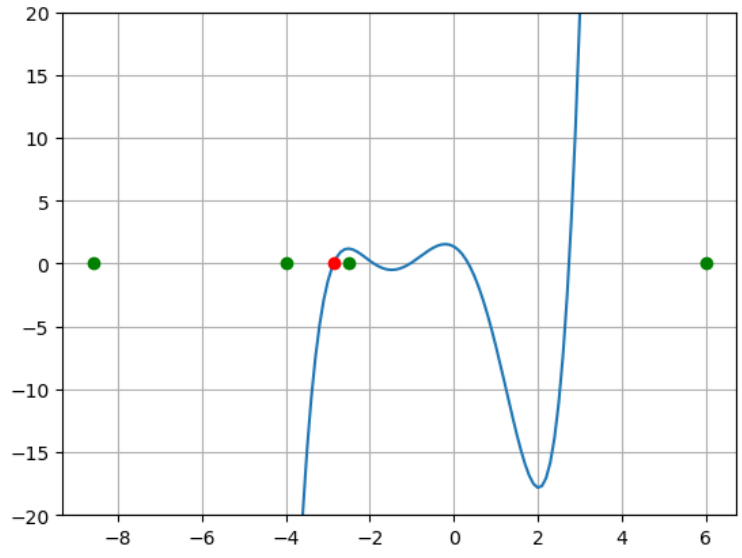
$$x_{mid} = \frac{x_n + kx_{n1}}{1+k}$$

```
    if np.sign(f(x1)) == np.sign(f(x)):
        x1 = x
    else:
        x2 = x
    k = np.abs(f(x1)/f(x2))
    x = (x1+k*x2)/(1+k)
    #x = (x1+x2)/2
    i += 1
    if i > 100:
        break
    print("{:>10d} {:>20.5e} {:>20.5e}".format(i, x, f(x)))
plt.plot(x, f(x), 'ro')
```

f(x) funkcijos stygų metodo rezultatas:

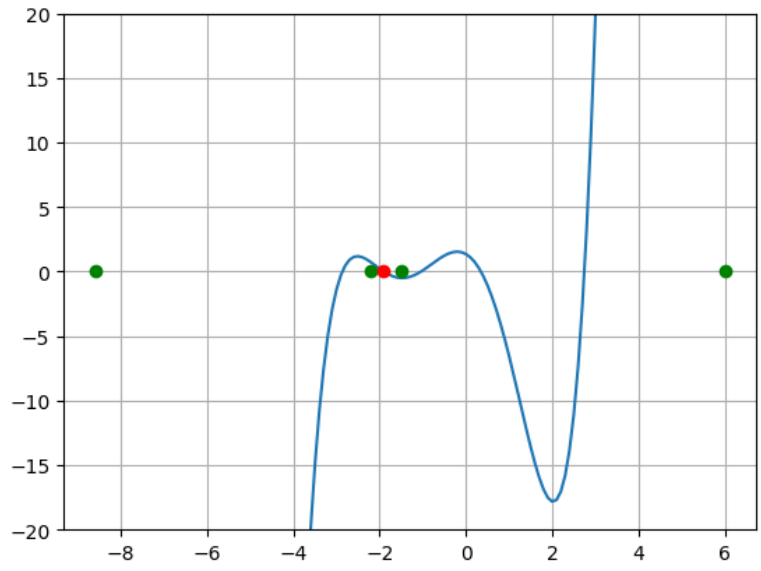
Intervalas: x= -4; x= -2.5

iteration	root	function value
1	-2.01778e+00	2.63494e-01
2	-2.02787e+00	2.88417e-01
3	-2.03886e+00	3.15709e-01
.	.	.
.	.	.
.	.	.
99	-2.86714e+00	2.60276e-06
100	-2.86714e+00	2.14896e-06



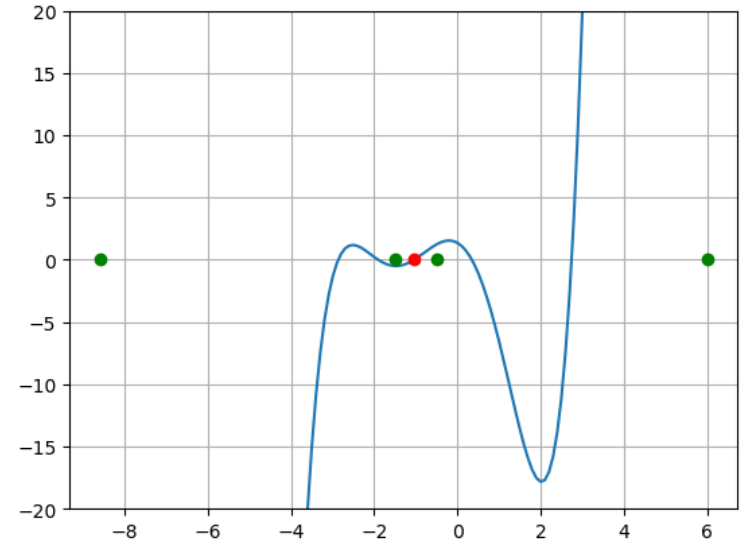
Intervalas: x= -2.2; x= -1.5

iteration	root	function value
1	-1.88691e+00	-3.86075e-02
2	-1.90289e+00	-4.42586e-03
3	-1.90471e+00	-4.75084e-04
4	-1.90490e+00	-5.06090e-05
5	-1.90492e+00	-5.38677e-06
6	-1.90492e+00	-5.73312e-07
7	-1.90493e+00	-6.10169e-08
8	-1.90493e+00	-6.49393e-09
9	-1.90493e+00	-6.91142e-10



Intervalas: x= -1.5; x= -0.5

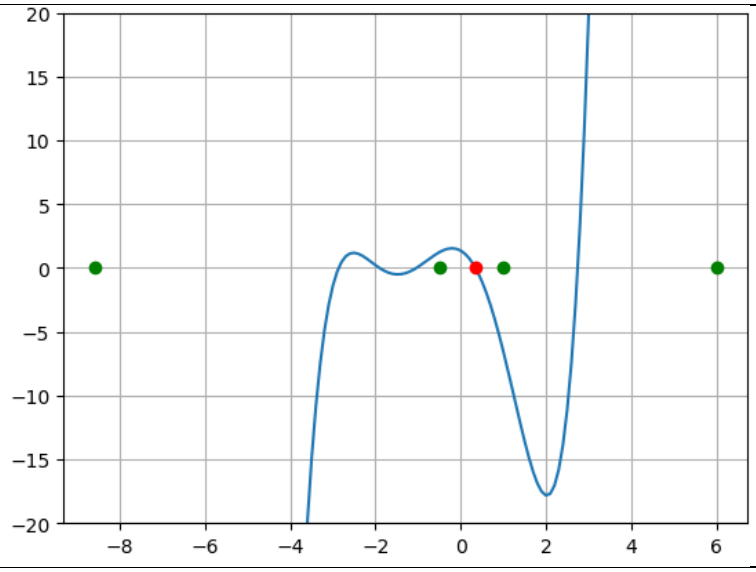
iteration	root	function value
1	-1.07368e+00	-6.37433e-02
2	-1.04566e+00	-7.16381e-03
3	-1.04252e+00	-6.89149e-04
4	-1.04222e+00	-6.50829e-05
5	-1.04219e+00	-6.13542e-06
6	-1.04219e+00	-5.78294e-07
7	-1.04219e+00	-5.45062e-08
8	-1.04219e+00	-5.13739e-09
9	-1.04219e+00	-4.84215e-10



Intervalas: x= -0.5; x= 1

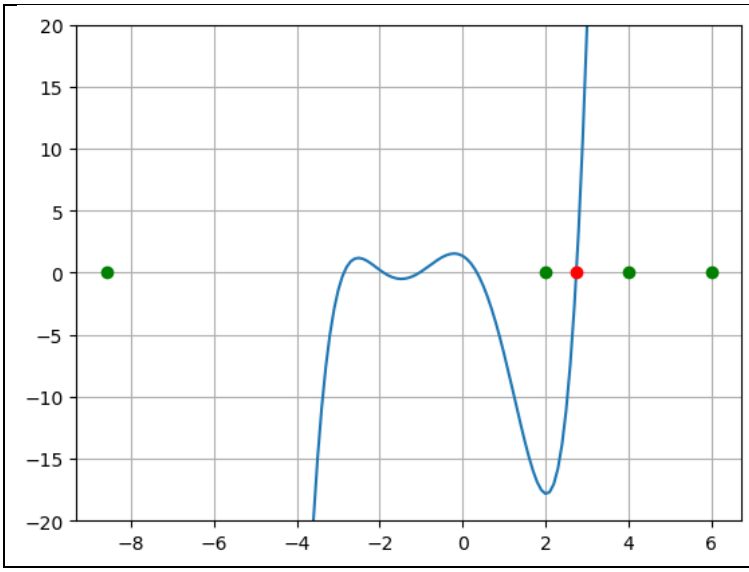
iteration	root	function value
1	-2.08636e-02	1.39758e+00
2	1.59107e-01	9.16763e-01
3	2.62629e-01	4.69235e-01
4	3.12063e-01	2.08770e-01
5	3.33375e-01	8.69556e-02
6	3.42136e-01	3.52099e-02
7	3.45664e-01	1.40932e-02
8	3.47073e-01	5.61478e-03
9	3.47634e-01	2.23280e-03
10	3.47857e-01	8.87249e-04
11	3.47946e-01	3.52463e-04
12	3.47981e-01	1.40001e-04

13	3.47995e-01	5.56069e-05
14	3.48000e-01	2.20861e-05
15	3.48003e-01	8.77213e-06
16	3.48003e-01	3.48410e-06
17	3.48004e-01	1.38380e-06
18	3.48004e-01	5.49616e-07
19	3.48004e-01	2.18295e-07
20	3.48004e-01	8.67018e-08
21	3.48004e-01	3.44360e-08
22	3.48004e-01	1.36772e-08
23	3.48004e-01	5.43226e-09
24	3.48004e-01	2.15757e-09
25	3.48004e-01	8.56938e-10



Intervalas: x= 2; x= 4

1	2.27078e+00	-1.62454e+01
2	2.38302e+00	-1.42496e+01
3	2.47582e+00	-1.18457e+01
4	2.54925e+00	-9.39360e+00
5	2.60523e+00	-7.16409e+00
6	2.64665e+00	-5.29865e+00
7	2.67662e+00	-3.82906e+00
8	2.69792e+00	-2.72032e+00
.	.	.
.	.	.
.	.	.
64	2.74625e+00	-1.50258e-09
65	2.74625e+00	-1.02523e-09
66	2.74625e+00	-6.99507e-10



f(x) funkcijos Kvazi-niutono metodo kodas:

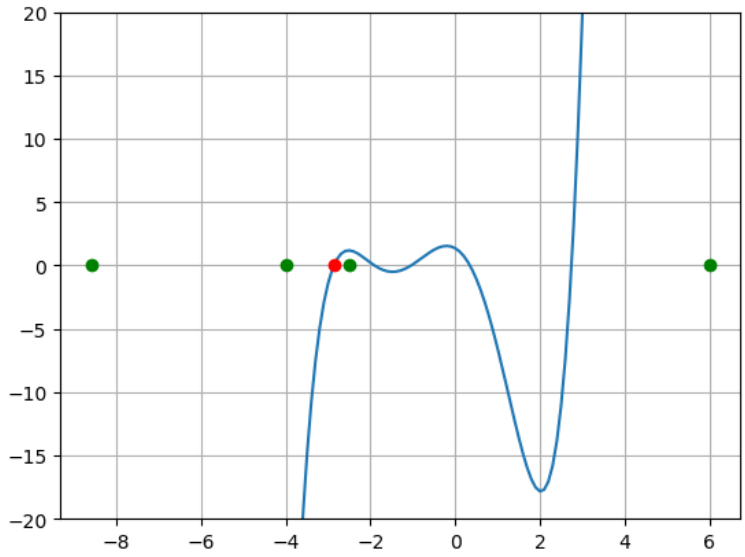
```
import numpy as np
import matplotlib.pyplot as plt
#kvazi niutono
def f(x):
    return 0.25*x**5+0.68*x**4-1.65*x**3-5.26*x**2-1.91*x+1.36

xmin = -8.6
xmax = 6
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])

x2 = -2
x1 = -4

plt.plot(x1, 0, 'go')
plt.plot(x2, 0, 'go')
print(" {:>10s} {:>20s} {:>20s}".format("iteration",  "root", "function value"))
i = 0
while np.abs(f(x2))>1e-9:
    x = x2 - f(x2)*(x2-x1)/(f(x2)-f(x1))
    x1 = x2
    x2 = x
    i += 1
    if i > 10:
        break
    print("{:>10d} {:>20.5e} {:>20.5e}".format(i, x, f(x)))
plt.plot(x, f(x), 'ro')
```

$$x^{i+1} = x^i - \left(\frac{f(x^i) - f(x^{i-1})}{x^i - x^{i-1}} \right)^{-1} f(x^i)$$

f(x) funkcijos Kvazi-niutono metodo rezultatas:

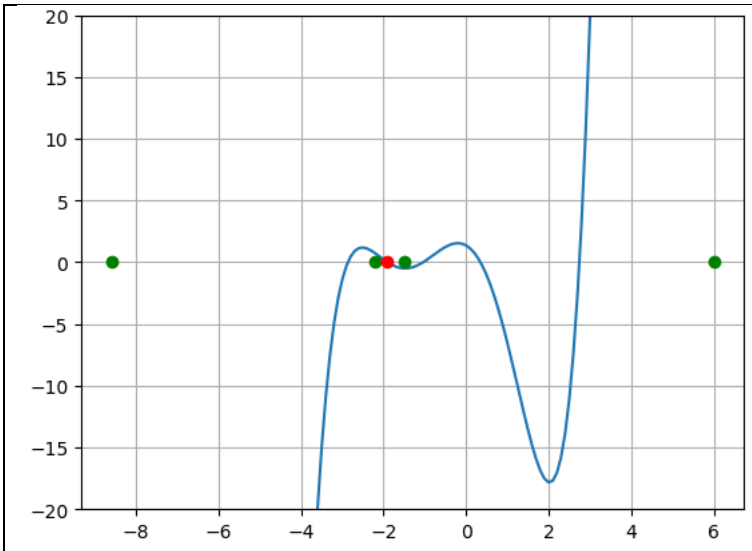
Intervalas: x= -4; x= -2.5

iteration	root	function value
1	-2.53388e+00	1.18944e+00
2	-2.56699e+00	1.17324e+00
3	-4.96566e+00	-2.58169e+02
4	-2.57784e+00	1.16422e+00
5	-2.58856e+00	1.15341e+00
6	-3.73231e+00	-2.81070e+01
7	-2.63365e+00	1.08593e+00
8	-2.67452e+00	9.91166e-01
9	-3.10199e+00	-2.92089e+00
10	-2.78282e+00	5.57601e-01
11	-2.83398e+00	2.45010e-01
12	-2.87408e+00	-5.58256e-02
13	-2.86664e+00	3.92692e-03
14	-2.86713e+00	5.61808e-05
15	-2.86714e+00	-5.79194e-08
16	-2.86714e+00	8.67528e-13



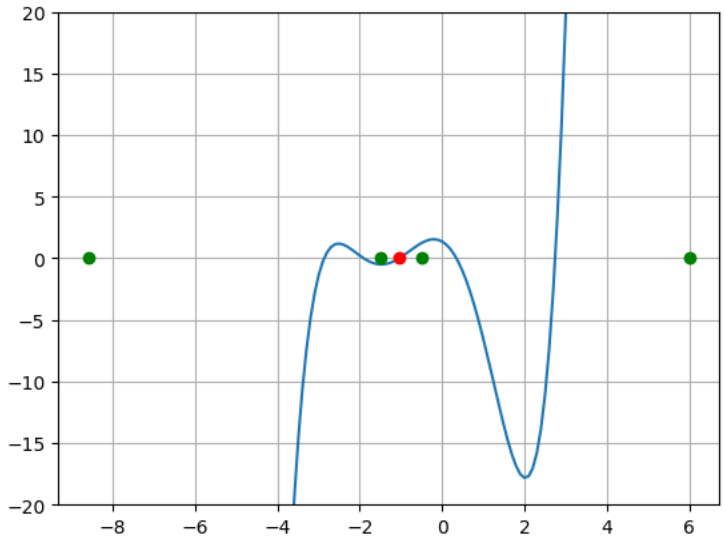
Intervalas: x= -2.2; x= -1.5

iteration	root	function value
1	-1.78637e+00	-2.30615e-01
2	-1.88691e+00	-3.86075e-02
3	-1.90713e+00	4.80294e-03
4	-1.90489e+00	-6.95106e-05
5	-1.90493e+00	-1.18799e-07
6	-1.90493e+00	2.95741e-12



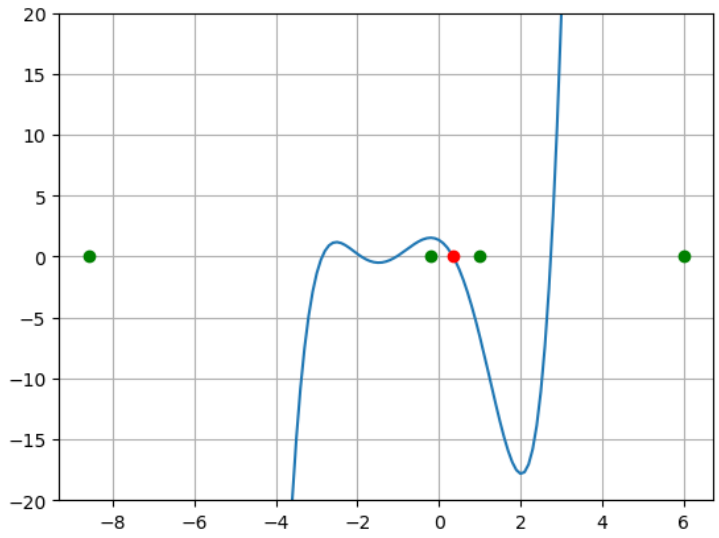
Intervalas: x= -1.5; x= -0.5

iteration	root	function value
1	-1.21395e+00	-3.03412e-01
2	-7.66060e-01	6.46358e-01
3	-1.07087e+00	-5.81743e-02
4	-1.04570e+00	-7.25612e-03
5	-1.04211e+00	1.60544e-04
6	-1.04219e+00	-4.10416e-07
7	-1.04219e+00	-2.30000e-11



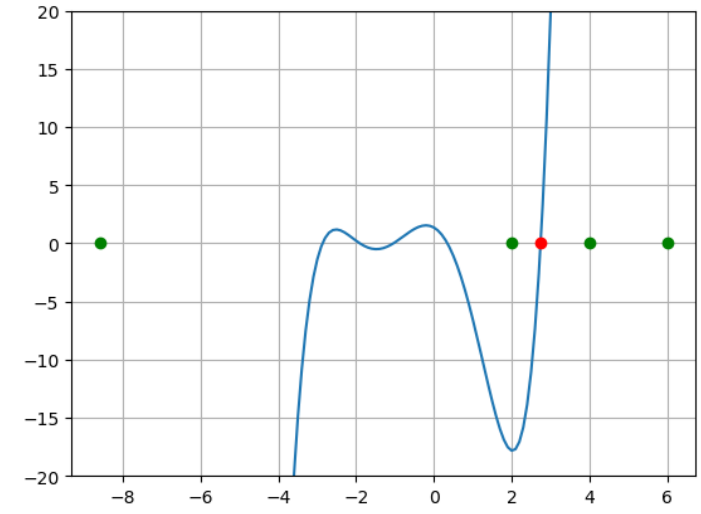
Intervalas: x= -0.2; x= -1

iteration	root	function value
1	2.96946e-02	1.29860e+00
2	1.23631e+00	-9.84832e+00
3	1.70264e-01	8.74773e-01
4	2.57230e-01	4.95825e-01
5	3.71019e-01	-1.42341e-01
6	3.45639e-01	1.42429e-02
7	3.47948e-01	3.40397e-04
8	3.48004e-01	-8.53224e-07
9	3.48004e-01	5.08649e-11



Intervalas: x= 2; x= 4

iteration	root	function value
1	2.14151e+00	-1.74960e+01
2	9.78323e+00	2.65688e+04
3	2.14654e+00	-1.74658e+01
4	2.15155e+00	-1.74343e+01
5	4.92799e+00	7.94367e+02
6	2.21118e+00	-1.69491e+01
7	2.26794e+00	-1.62842e+01
8	3.65798e+00	1.28716e+02
9	2.42404e+00	-1.32772e+01
10	2.53943e+00	-9.75174e+00
11	2.85858e+00	7.50049e+00
12	2.71982e+00	-1.52329e+00
13	2.74325e+00	-1.77703e-01
14	2.74634e+00	5.16533e-03
15	2.74625e+00	-1.67101e-05
16	2.74625e+00	-1.56302e-09
17	2.74625e+00	1.04361e-14



g(x) funkcijos stygų metodo kodas:

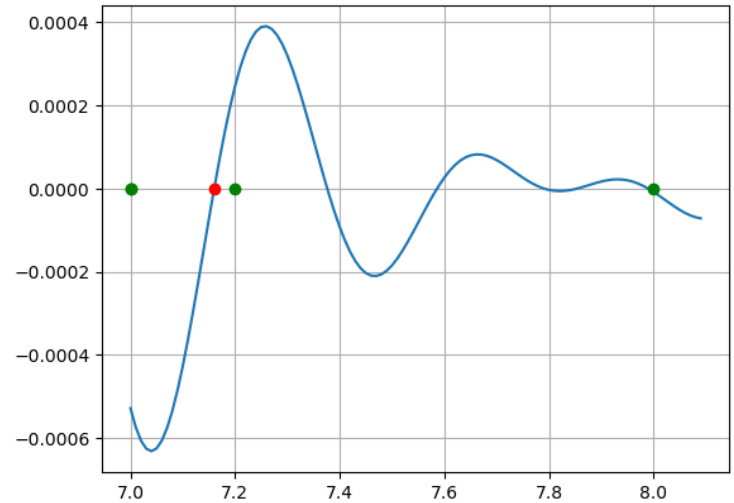
```
def g(x):
    return np.exp(-x)*np.cos(x)*np.sin((x**2)-1)
x = np.arange(7, 8.1, 0.01)
```

```
#print(x)
plt.plot(x, g(x))
plt.plot(7, 0, 'go')
plt.plot(8, 0, 'go')
plt.grid()

x1 = 7
x2 = 7.2
plt.plot(x1, 0, 'go')
plt.plot(x2, 0, 'go')
k = np.abs(g(x1)/g(x2))
x = (x1+k*x2)/(1+k)
print(x)
#x = (x1+x2)/2
print(" {:>10s} {:>20s} {:>20s}".format("iteration",  "root", "function value"))
i = 0
while np.abs(g(x))>1e-9:
    if np.sign(g(x1)) == np.sign(g(x)):
        x1 = x
    else:
        x2 = x
    k = np.abs(g(x1)/g(x2))
    x = (x1+k*x2)/(1+k)
    #x = (x1+x2)/2
    i += 1
    if i > 100:
        break
    print("{:>10d} {:>20.5e} {:>20.5e}".format(i, x, g(x)))
plt.plot(x, g(x), 'ro')
```

g(x) funkcijos stygų metodo rezultatas:

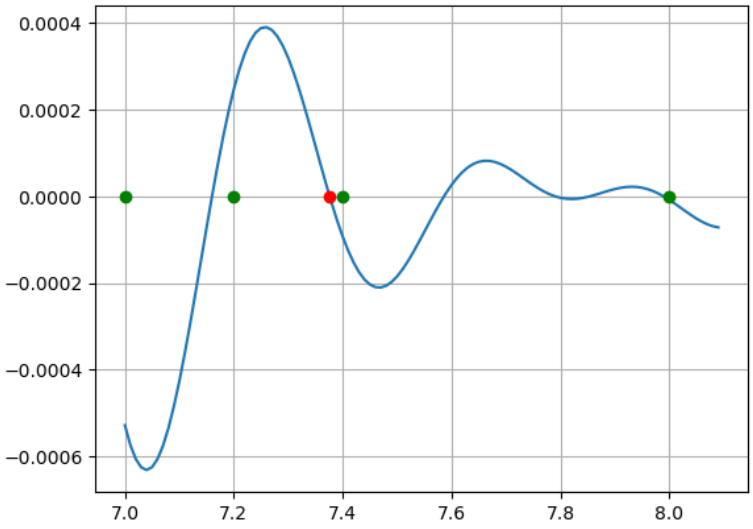
Intervalas: x= 7; x= 7.2

iteration	root	function value
1	7.16263e+00	1.86438e-05
2	7.16008e+00	6.07022e-07
3	7.15999e+00	1.83641e-08
4	7.15999e+00	5.54207e-10



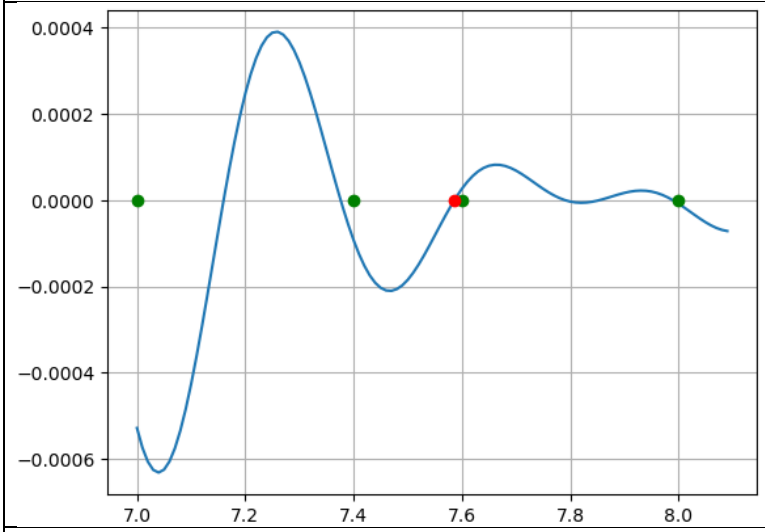
Intervalas: x= 7.2; x= 7.4

iteration	root	function value
1	7.37800e+00	-7.97848e-06
2	7.37621e+00	-4.20120e-07
3	7.37612e+00	-2.11320e-08
4	7.37612e+00	-1.06035e-09
5	7.37612e+00	-5.31993e-11



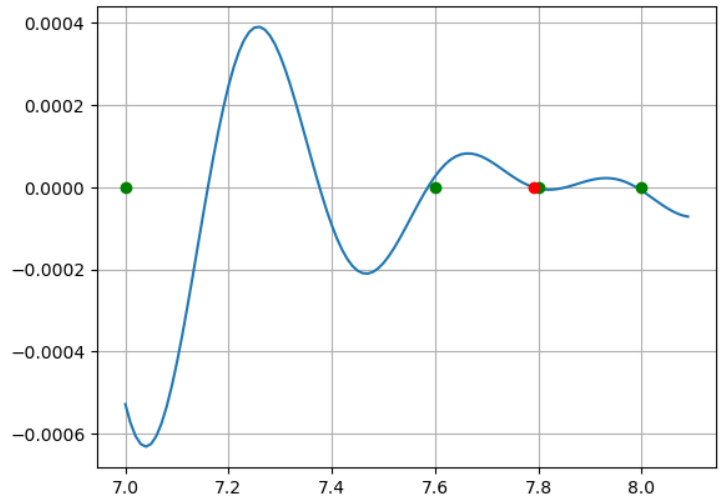
Intervalas: x= 7.4; x= 7.6

iteration	root	function value
1	7.58765e+00	3.17573e-06
2	7.58623e+00	3.03744e-07
3	7.58610e+00	2.82791e-08
4	7.58609e+00	2.62606e-09
5	7.58608e+00	2.43804e-10



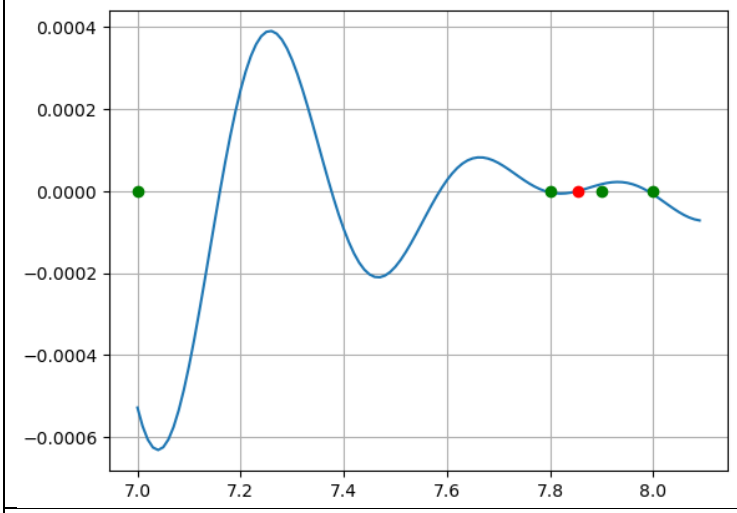
Intervalas: x= 7.6; x= 7.8

iteration	root	function value
1	7.79230e+00	-7.57040e-07
2	7.79073e+00	-1.36109e-07
3	7.79045e+00	-2.33484e-08
4	7.79041e+00	-3.97269e-09
5	7.79040e+00	-6.75005e-10



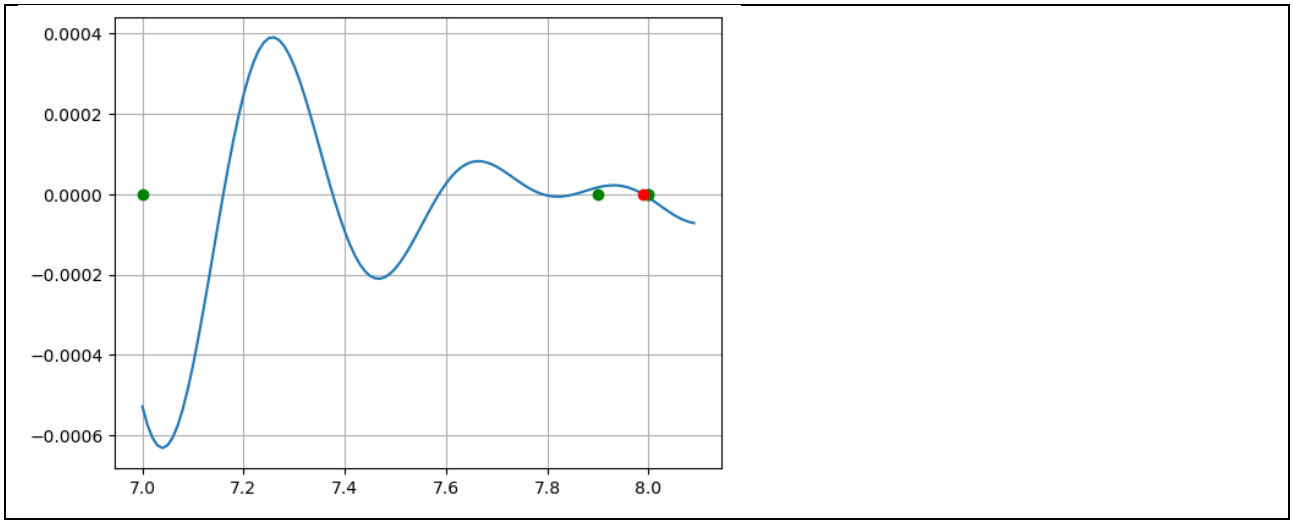
Intervalas: x= 7.8; x= 7.9

iteration	root	function value
1	7.83824e+00	-4.22099e-06
2	7.85060e+00	-1.06410e-06
3	7.85353e+00	-1.44956e-07
4	7.85393e+00	-1.66476e-08
5	7.85398e+00	-1.86592e-09
6	7.85398e+00	-2.08552e-10



Intervalas: x= 7.9; x= 8

iteration	root	function value
1	7.98773e+00	1.26963e-06
2	7.98938e+00	7.62506e-08
3	7.98948e+00	4.42659e-09
4	7.98948e+00	2.56462e-10



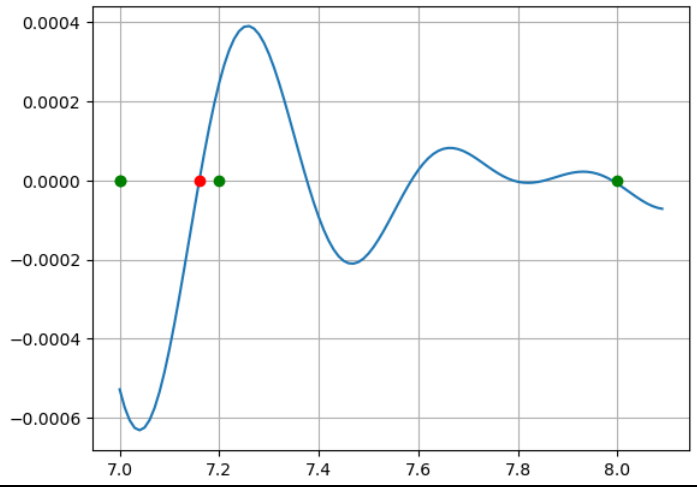
g(x) funkcijos Kvazi-niutono metodo kodas:

```
def g(x):
    return np.exp(-x)*np.cos(x)*np.sin((x**2)-1)
x = np.arange(7, 8.1, 0.01)
#print(x)
plt.plot(x, g(x))
plt.plot(7, 0, 'go')
plt.plot(8, 0, 'go')
plt.grid()

x1 = 7.4
x2 = 7.6

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

print(" {:>10s} {:>20s} {:>20s}".format("iteration",  "root", "function value"))
i = 0
while np.abs(g(x2))>1e-9:
    x = x2 - g(x2)*(x2-x1)/(g(x2)-g(x1))
    x1 = x2
    x2 = x
    i += 1
    if i > 10:
        break
    print("{:>10d} {:>20.5e} {:>20.5e}".format(i, x, g(x)))
plt.plot(x, g(x), 'ro')
```

g(x) funkcijos Kvazi-niutono metodo rezultatas:

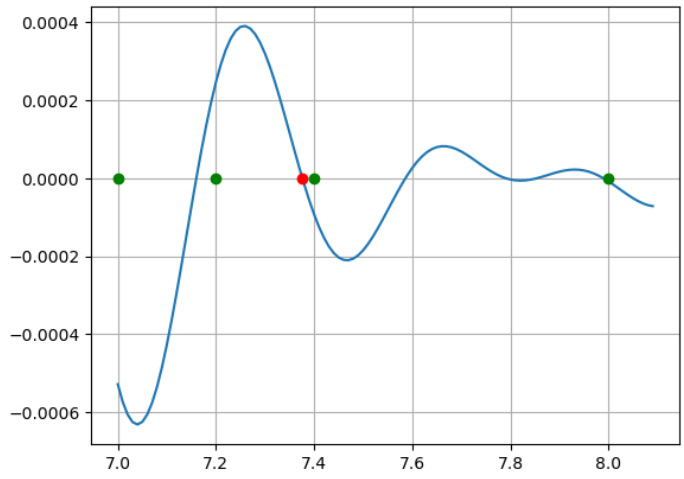
Intervalas: x= 7; x= 7.2

iteration	root	function value
1	7.13630e+00	-1.73849e-04
2	7.16263e+00	1.86438e-05
3	7.16008e+00	6.07022e-07
4	7.15999e+00	-3.57561e-09
5	7.15999e+00	6.51262e-13



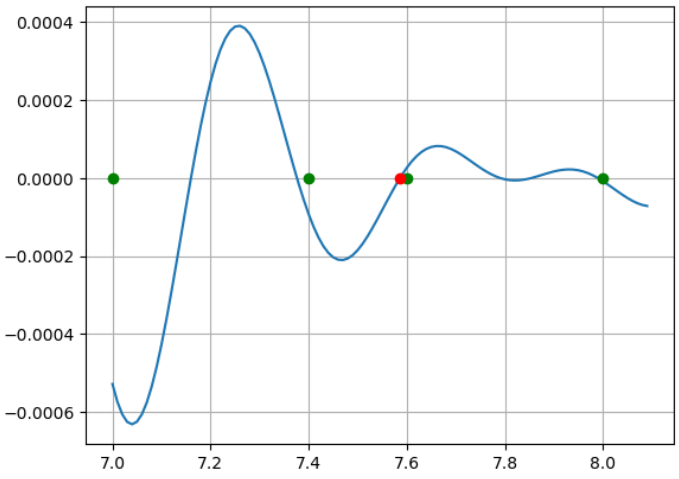
Intervalas: x= 7.2; x= 7.4

iteration	root	function value
1	7.34541e+00	1.37287e-04
2	7.37800e+00	-7.97848e-06
3	7.37621e+00	-4.20120e-07
4	7.37611e+00	2.34062e-09
5	7.37612e+00	-6.64043e-13



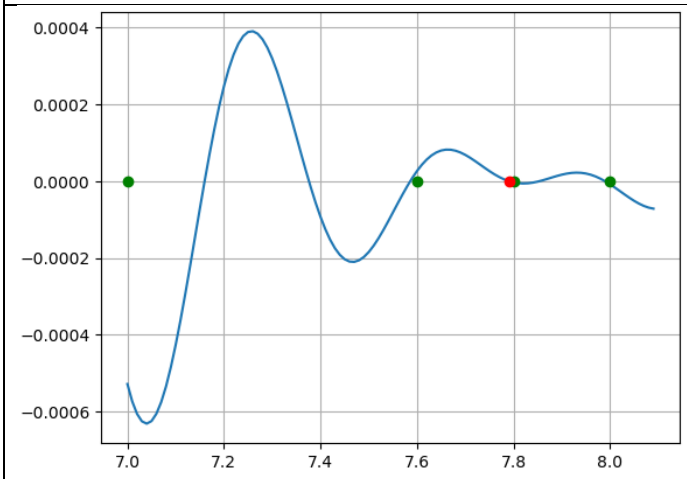
Intervalas: x= 7.4; x= 7.6

iteration	root	function value
1	7.55568e+00	-6.83024e-05
2	7.58765e+00	3.17573e-06
3	7.58623e+00	3.03744e-07
4	7.58608e+00	-2.22998e-09
5	7.58608e+00	1.52499e-12



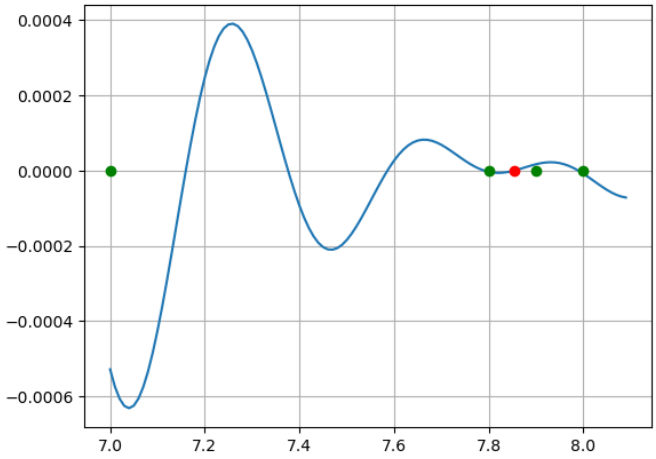
Intervalas: x= 7.6; x= 7.8

iteration	root	function value
1	7.77777e+00	6.22818e-06
2	7.79230e+00	-7.57040e-07
3	7.79073e+00	-1.36109e-07
4	7.79038e+00	4.53345e-09
5	7.79040e+00	-2.53552e-11



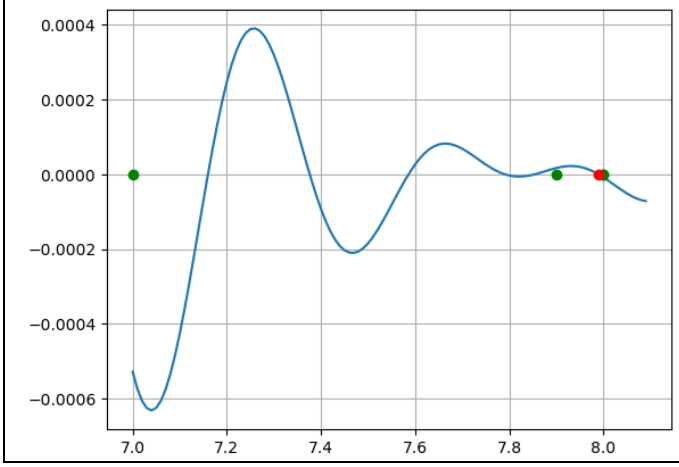
Intervalas: x= 7.8; x= 7.9

iteration	root	function value
1	7.81636e+00	-5.97668e-06
2	7.83824e+00	-4.22099e-06
3	7.89086e+00	1.37932e-05
4	7.85057e+00	-1.07436e-06
5	7.85348e+00	-1.62422e-07
6	7.85400e+00	5.62431e-09
7	7.85398e+00	-2.62365e-11



Intervalas: x= 7.9; x= 8

iteration	root	function value
1	7.96737e+00	1.35519e-05
2	7.98773e+00	1.26963e-06
3	7.98983e+00	-2.56289e-07
4	7.98948e+00	2.92850e-09
5	7.98948e+00	6.51507e-12



f(x) funkcijos rezultatų lentelė

Metodas	Intervalas	Šaknis	Funkcijos reikšmė ties šaknimi	Tikslumas	Iteracijų skaičius	Gauta šaknis naudojant https://www.symbolab.com/solver/roots-calculator
Stygos metodas	(-4; -2.5)	-2.86	$2.149*10^{-06}$	Iki 100 iteracijų/ šimtųjų tikslumu	100	-2.87
	(-2.2; -1.5)	-1.9	$-6.91*10^{-10}$		9	-1.9
	(-1.5; -0.5)	-1.04	$-4.84*10^{-10}$		9	-1.04
	(-0.5; 1)	0.348	$8.569*10^{-10}$		25	0.348
	(2; 4)	2.75	$-6.9*10^{-10}$		66	2.75
	(-4; -2.5)	-2.867	$8.675*10^{-13}$		16	-2.87

Kvazi-niutono metodas	(-2.2; -1.5)	-1.9	$2.957*10^{-12}$		6	-1.9
	(-1.5; -0.5)	-1	$-2.3*10^{-11}$		7	-1.04
	(-0.5; 1)	0.348	$5*10^{-11}$		9	0.348
	(2; 4)	2.746	$1*10^{-14}$		17	2.75

g(x) funkcijos rezultatų lentelė						
Metodas	Intervalas	Šaknis	Funkcijos reikšmė ties šaknimi	Tikslumas	Iteracijų skaičius	Gauta šaknis naudojant https://www.symbolab.com/solver/roots-calculator
Stygos metodas	(7; 7.2)	7.16	$5.54*10^{-10}$	Iki 100 iteracijų / šimtųjų tikslumu	4	7.16
	(7.2; 7.4)	7.38	$-5.32*10^{-11}$		5	7.38
	(7.4; 7.6)	7.59	$2.44*10^{-10}$		5	7.59
	(7.6; 7.8)	7.79	$-6.75*10^{-10}$		5	7.79
	(7.8; 7.9)	7.85	$-2*10^{-10}$		6	7.85
	(7.9; 8)	7.99	$2.56*10^{-10}$		4	7.99
Kvazi-niutono metodas	(7; 7.2)	7.16	$6.5*10^{-13}$		5	7.16
	(7.2; 7.4)	7.38	$-6.64*10^{-13}$		5	7.38
	(7.4; 7.6)	7.59	$1.52*10^{-12}$		5	7.59
	(7.6; 7.8)	7.79	$-2.54*10^{-11}$		5	7.79
	(7.8; 7.9)	7.85	$-2.62*10^{-11}$		7	7.85
	(7.9; 8)	7.99	$6.51*10^{-12}$		5	7.99

Žiūrint į rezultatų lenteles lengva pastebėti, kai yra didesni intervalai, tai Kvazi-niutono metodas susitvarko per mažiau iteracijų, tačiau kai intervalai mažėja, stygų metodas yra greitesnis. Visais atvejais aiškiai matosi, jog Kvazi-niutono metodas yra tikslesnis, nes gaunamas funkcijos reikšmė labiau artėja prie 0.

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
10	$-79\cos(x) - 11 - 4x$	$-10 \le x \le 0$

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.

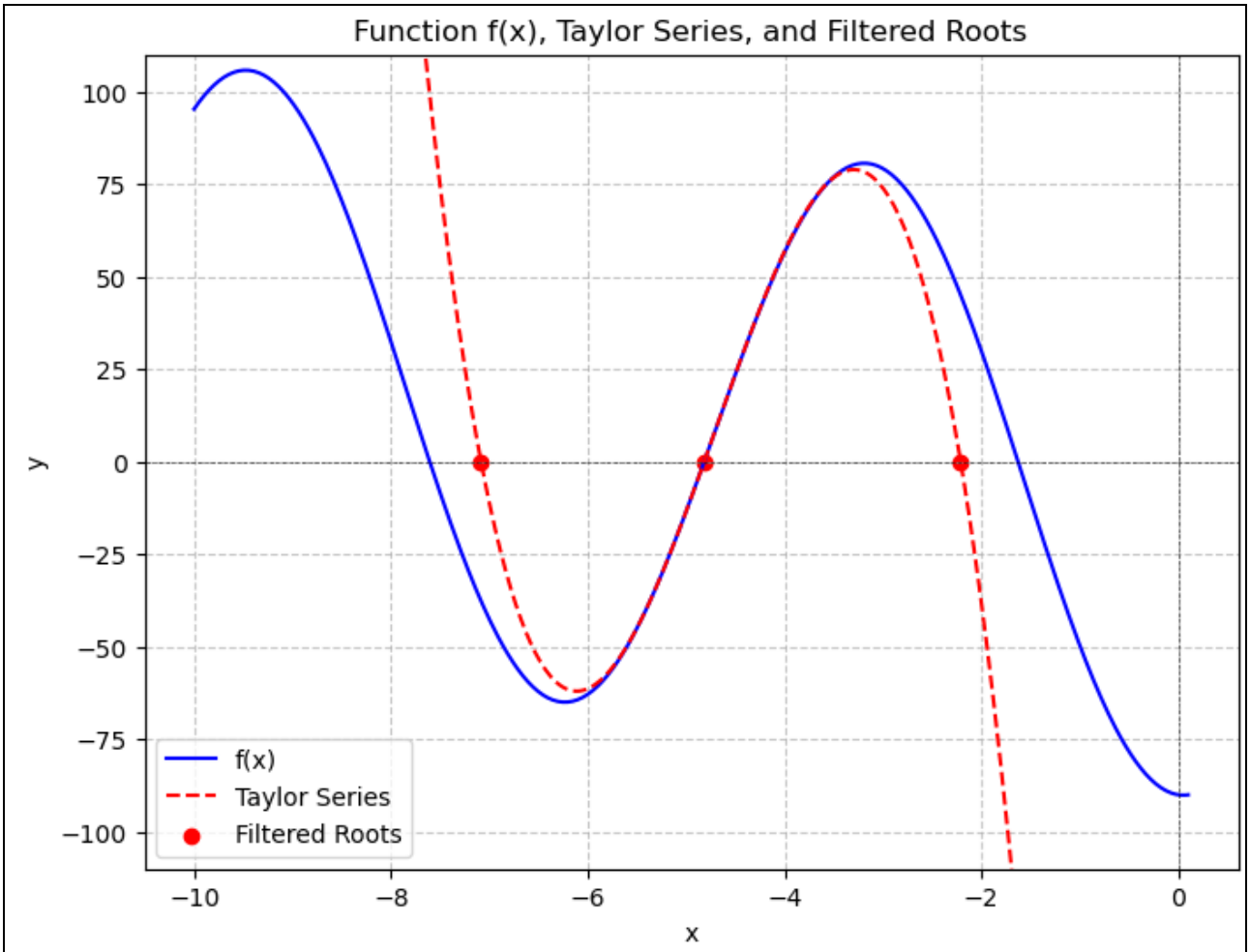
$$\tilde{f}(x) = f(x_0) + (x - x_0)\left.\frac{df}{dx}\right|_{x_0} + \frac{(x - x_0)^2}{2}\left.\frac{d^2 f}{dx^2}\right|_{x_0} + \frac{(x - x_0)^3}{6}\left.\frac{d^3 f}{dx^3}\right|_{x_0} + \cdots + \frac{(x - x_0)^n}{n!}\left.\frac{d^n f}{dx^n}\right|_{x_0} + \cdots$$

Programos kodas:

```
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 = (-79) * sympy.cos(x) - 11 - 4 * x
x0 = -5
N = 10

# 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
```

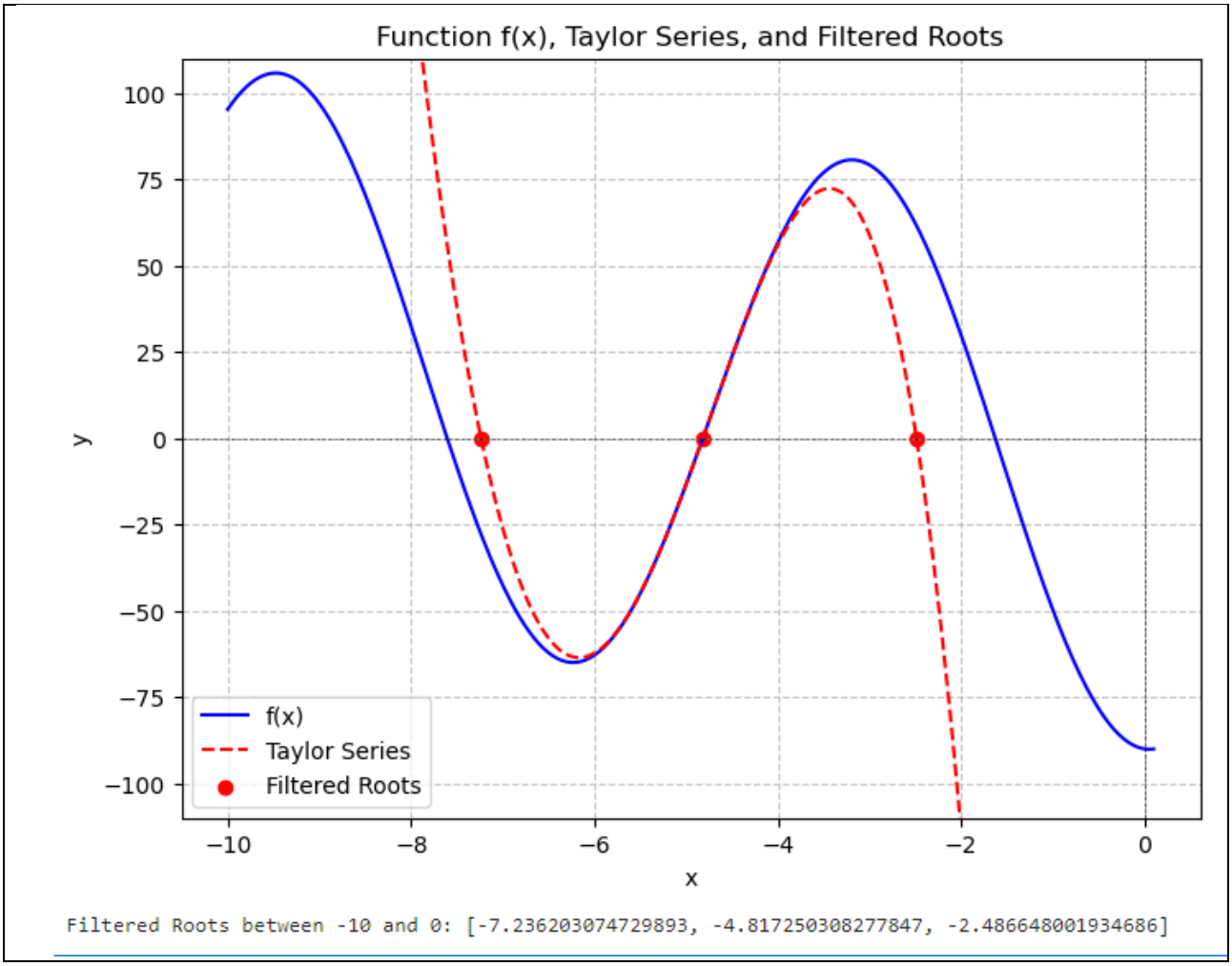
```
taylor_values = [taylor_function(val) for val in x_values]  # Calculate Taylor series values

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

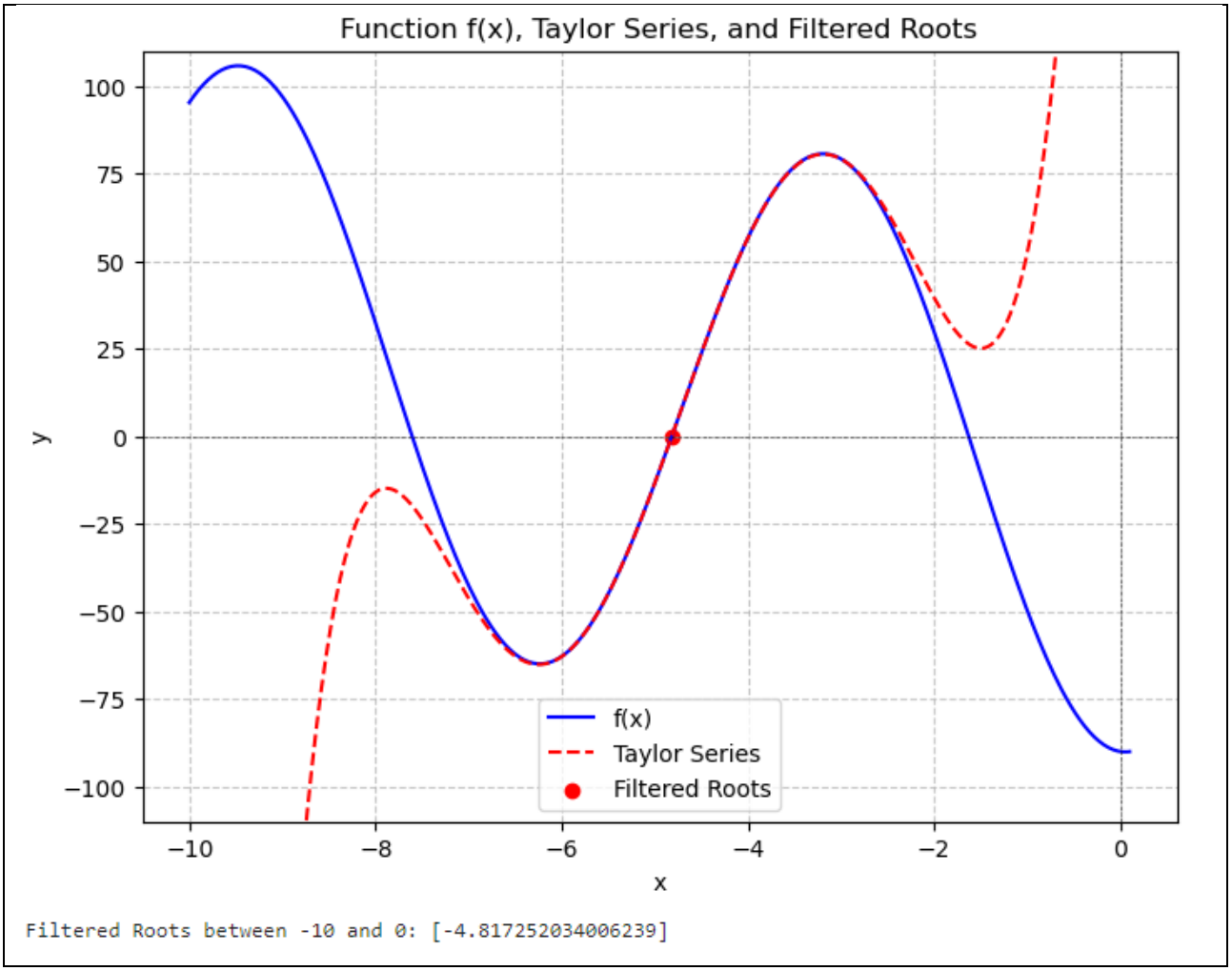



```
Filtered Roots between -10 and 0: [-7.084062927642108, -4.817264271672404, -2.211234054087255]
```

Grafikas kai TE narių skaičius: 4



Grafikas kai TE narių skaičius: 5




```
Filtered Roots between -10 and 0: [-4.817252034006239]
```

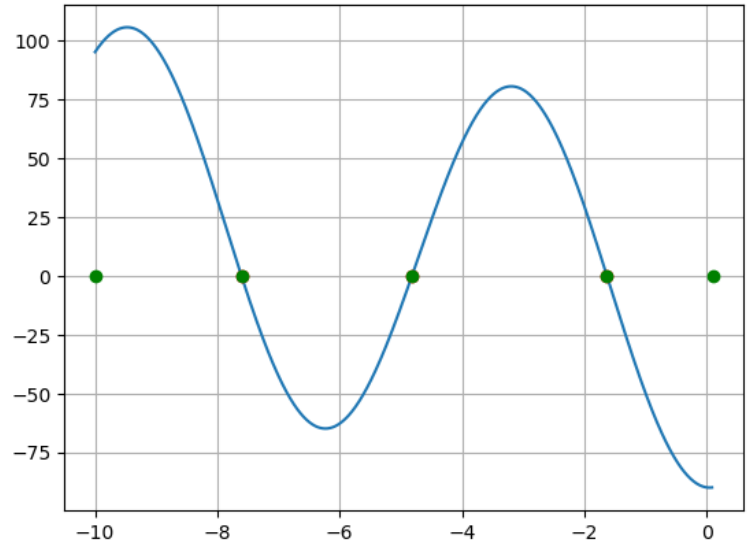
2. Šaknų intervalų radimas naudojant skenavimo metodą:

Kodas:

```
def z(x):
    return (-79)*np.cos(x)-11-4*x
x = np.arange(-10, 0.1, 0.01)
#print(x)
plt.plot(x, z(x))
plt.plot(-10, 0, 'go')
plt.plot(0.1, 0, 'go')
plt.grid()

x = -10
dx = 0.0005
while x < (0 + dx):
    if( np.sign(z(x)) != np.sign(z(x+dx))):
        print("intervalas:")
        print(x)
        print(x+dx)
        plt.plot(x, 0, 'ro')
        plt.plot(x+dx, 0, 'go')
    x += dx
```

Rezultatai:

```
intervalas:
-7.605999999997775
-7.605499999997775
intervalas:
-4.81749999999932
-4.816999999999321
intervalas:
-1.6279999999991464
-1.6274999999991464
```

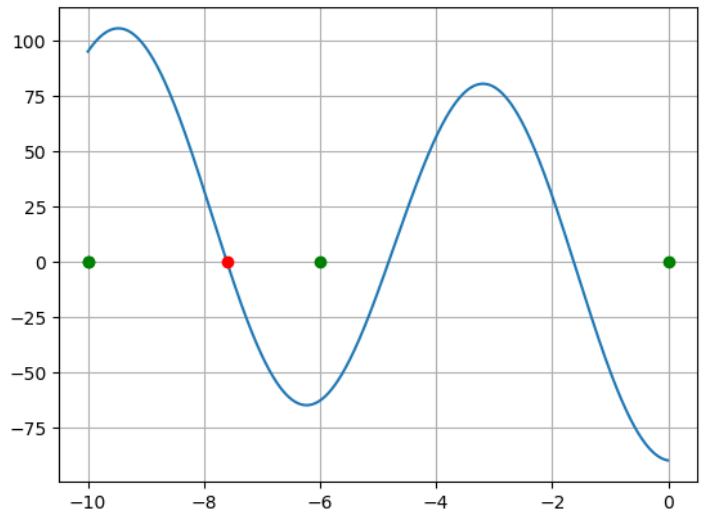


3. Tikslinu šaknis naudojant Stygų ir Kvazi-niutono metodus:

H(x) funkcijos šaknų tikslinimas naudojant stygų metodą:

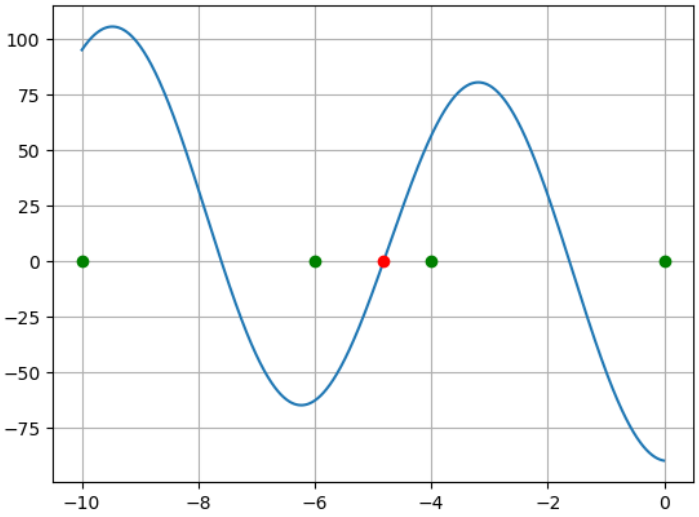
Intervalas: x=-10; x= -6

iteration	root	function value
1	-7.62147e+00	1.28215e+00
2	-7.60555e+00	-2.43134e-03
3	-7.60558e+00	-4.54859e-06
4	-7.60558e+00	-8.50918e-09
5	-7.60558e+00	-1.59623e-11



Intervalas: x=-6; x= -4

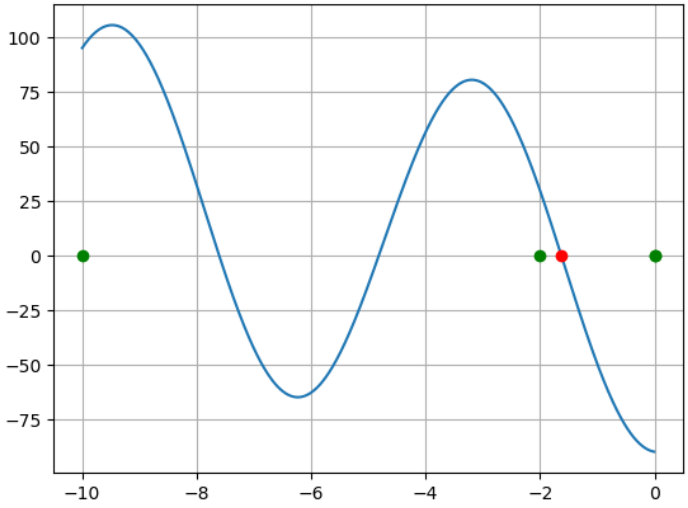
iteration	root	function value
1	-4.81000e+00	5.41038e-01
2	-4.81733e+00	-5.49997e-03
3	-4.81725e+00	-2.16073e-06
4	-4.81725e+00	-8.48690e-10



Intervalas: x=-2; x= 0

iteration	root	function value
1	-1.63081e+00	2.61396e-01
2	-1.62766e+00	2.49162e-04
3	-1.62765e+00	2.21767e-07

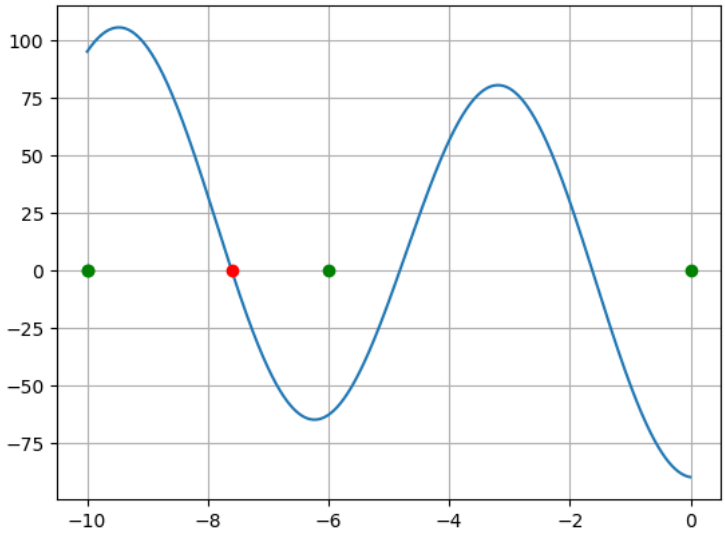
4	-1.62765e+00	1.97383e-10



H(x) funkcijos šaknų tikslinimas naudojant Kvazi-niutono metodą:

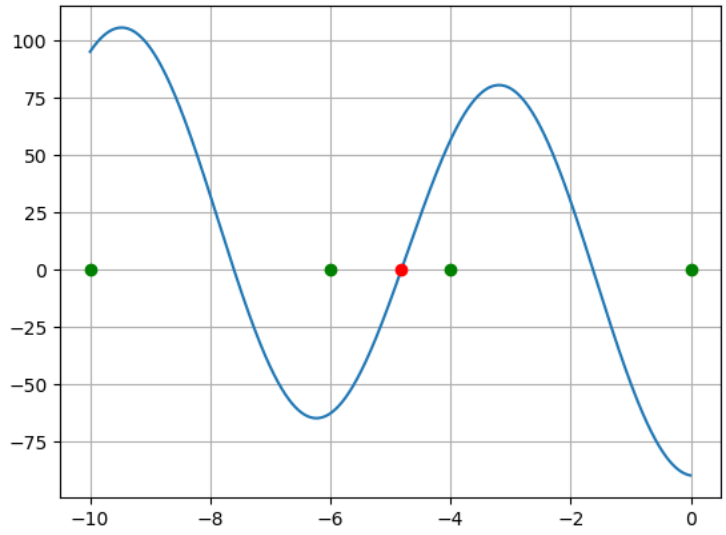
Intervalas: x=-10; x= -6

iteration	root	function value
1	-7.58982e+00	-1.26787e+00
2	-7.62255e+00	1.36961e+00
3	-7.60555e+00	-2.59285e-03
4	-7.60558e+00	-5.17283e-06
5	-7.60558e+00	2.00338e-11



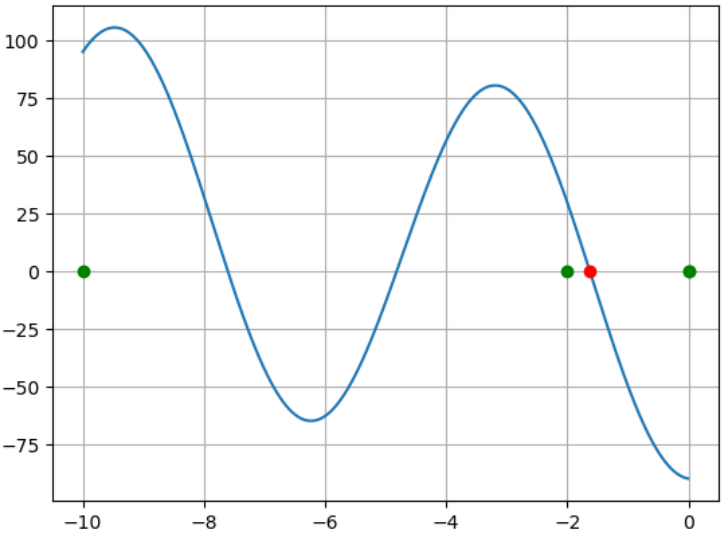
Intervalas: x=-6; x= -4

iteration	root	function value
1	-4.94798e+00	-9.64828e+00
2	-4.81000e+00	5.41038e-01
3	-4.81733e+00	-5.49997e-03
4	-4.81725e+00	-2.16073e-06
5	-4.81725e+00	8.83205e-12



Intervalas: x=-2; x= 0

iteration	root	function value
1	-1.50156e+00	-1.04593e+01
2	-1.69901e+00	5.89690e+00
3	-1.62782e+00	1.37136e-02
4	-1.62765e+00	-3.76932e-05
5	-1.62765e+00	1.69119e-10

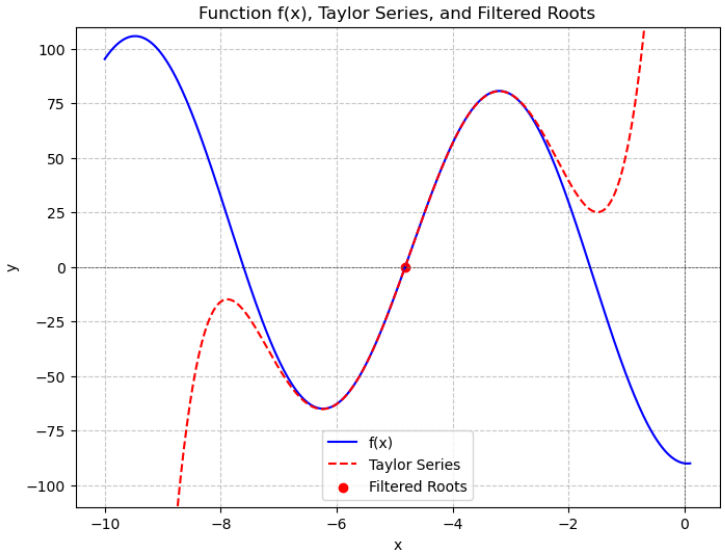


H(f) funkcijos gautos šaknys naudojant stygų ir Kvazi-niutono metodus:

Šaknys		
-7.6	-4.82	-1.627

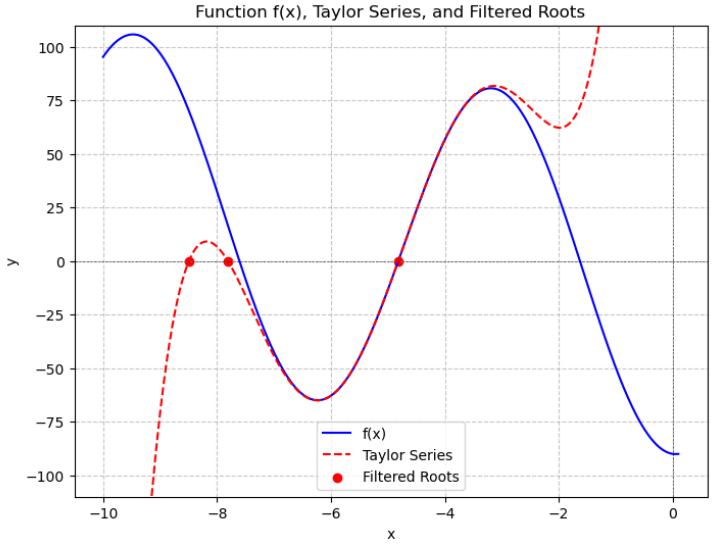
4. Progresyviai augantis TE nariu skaičius, iki tol kol gauname tinkamas šaknis:

Kai n=5;



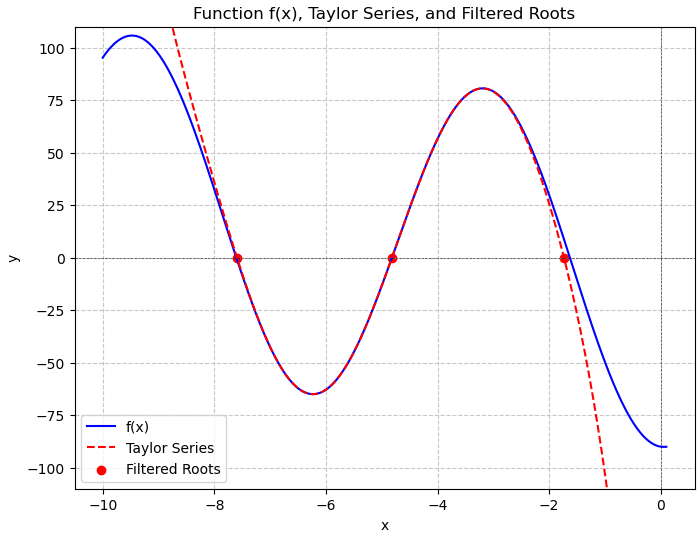

```
Filtered Roots between -10 and 0: [-4.817252034006239]
```

Kai n=6;



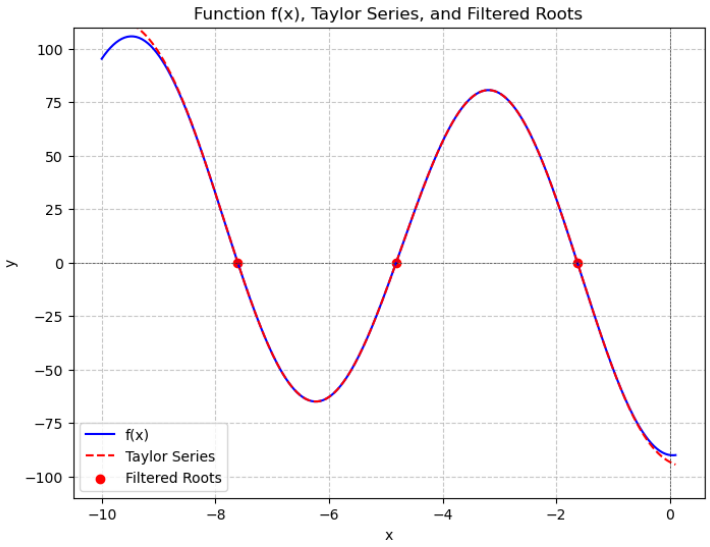

```
Filtered Roots between -10 and 0: [-8.493291409289666, -7.808164657265181, -4.8172520495543365]
```

Kai n=8;



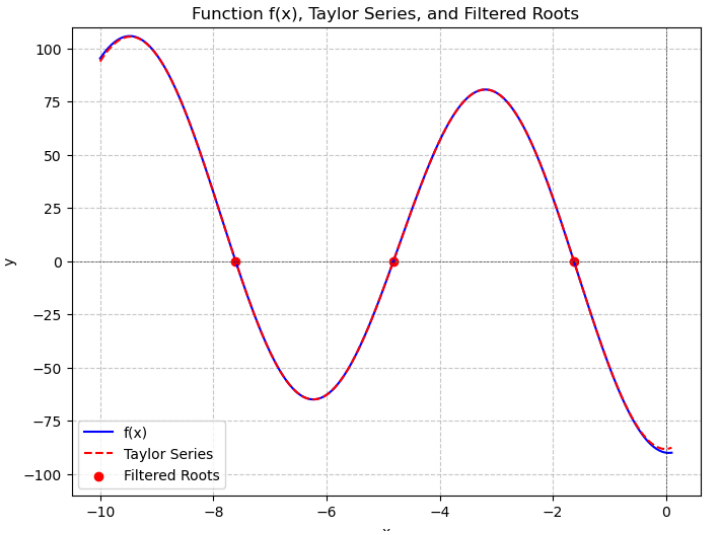

```
Filtered Roots between -10 and 0: [-7.593615955713709, -4.817252048172338, -1.7343976604943674]
```

Kai n=11;



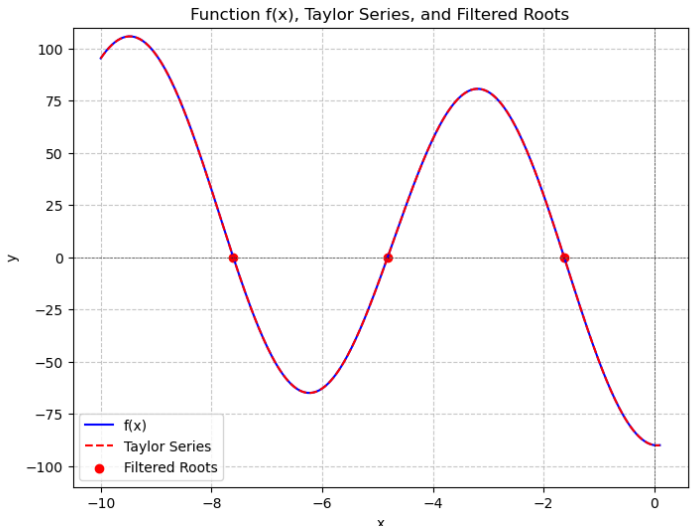

```
Filtered Roots between -10 and 0: [-7.605490703372122, -4.817252048173406, -1.6275214255677681]
```

Kai n=14;




```
Filtered Roots between -10 and 0: [-7.605583916422343, -4.817252048173251, -1.627595411990994]
```

Kai n=17;




```
Filtered Roots between -10 and 0: [-7.605582762163496, -4.817252048173333, -1.6276546261124687]
```

5. TE gautų šaknų palyginimas su gautomis šaknimis iš stygų ir Kvazi-niutono metodų

Šaknys			
TE šaknys	-7.6	-4.8	-1.63
Kiti metodai	-7.6	-4.82	-1.627