KAUNO TECHNOLOGIJOS UNIVERSITETAS
Informatikos fakultetas

P170B115 Skaitiniai metodai ir algoritmai

Laboratorinis darbas nr. 1

Variantas 12

Dėstytojai:
Lekt. Dalia Čalnerytė

Studentai:
Justas Milišiūnas IFF-7/2

KAUNAS, 2019

1. Įvadas

Šio laboratorinio darbo esmė išmokti skaičiuoti sudėtingų lygčių nežinomuosius sprendinius pasinaudojant kompiuteriu. Išmokti paskaičiuoti grubius bei tiksliuosius intervalus. Naudoti skenavimo nekintančių žingsniu rasti šaknų intervalams, pritaikyti įvairius tikslinimo metodus.

2. Užduotis

Vr.	f(x)	g(x)	metodai
12	$0.16x^5 - 1.57x^4 + 4.38x^3 - 1.15x^2 - 6.29x + 0.15$	$2x\sin(x) - \left(\frac{x}{2} + 2\right)^2;\ -10 \le x \le 10$	2, 3, 5

1 pav. f(x) daugianaris, g(x) - transcendentinė funkcija

Sprendimo metodai: paprastųjų iteracijų, Niutono (liestinių), skenavimo su mažėjančiu žingsniu

Uždavinys variantams 11-15

Vertikaliai į viršų iššauto objekto greitis užrašomas dėsniu $v(t) = v_0 e^{-\frac{ct}{m}} + \frac{mg}{c}\left(e^{-\frac{ct}{m}} - 1\right)$, čia $g = 9{,}8\ \mathrm{m/s^2}$, pradinis greitis v_0, objekto masė m. Koks pasipriešinimo koeficientas c veikia objektą, jei žinoma, kad po t_1 laiko nuo iššovimo jo greitis lygus v_1?

Varianto Nr.	v_0, m/s	m, kg	t_1, s	v_1, m/s
11	100	1,5	5	22
12	80	0,5	4	10
13	50	0,45	3	10
14	100	0,75	3	49
15	50	2	3	14

2 pav. Netiesinių lygčių sprendimo užduotis

Pasirinktas sprendimo metodas - skenavimo su mažėjančiu žingsniu

2.1. f(x) ir g(x) šaknų radimas

Lygties $f(x) = 0$ šaknų grubus įvertis:

$$|x| < 1 + \frac{\max\limits_{0 \le i \le n-1} |a_i|}{a_n} = R$$

Lygties $f(x) = 0$ šaknų tikslesnis įvertis:

(teigiamoms šaknims)

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

(neigiamoms šaknims)

Nagrinėjamas daugianaris $f(-x)$, jeigu n lyginis, ir $-f(-x)$, jei n nelyginis.

Galutinis įvertis:

$$-\min(R, R_{neig}) \le x \le \min(R, R_{teig})$$

3 pav. Intervalų skaičiavimo formulės

Grubus lygties šaknų intervalų įvertis: (-28.375, 28.375)
Tikslesnis lygties šaknų intervalų įvertis: (3.50399043568, 40.3125)
-min(-28.375, 3.504) ≤ x ≤ min(28.375, 40.3125)
Šaknų intervalas: -3.504 ≤ x ≤ 28.375

Skenavimas nekintančiu žingsniu:

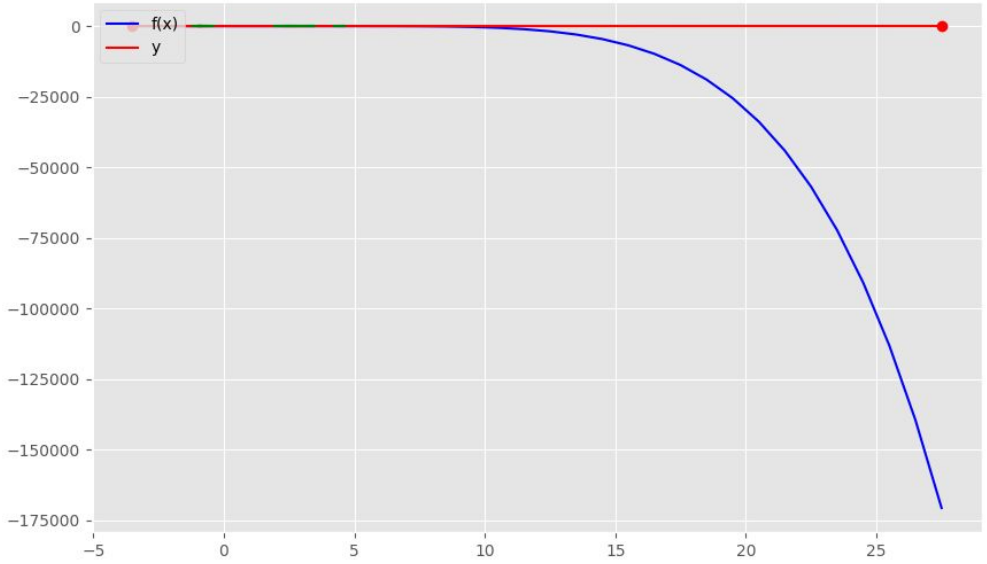
Šaknų intervalams rasti naudojamas skenavimo nekintančių žingsniu metodas. Žingsnis gaunamas padalinus visą intervalą iš 40. Tada eina per visą intervalą ir tikrina ar dabartinio x ženklas nesutampa su x + step ženklu. Jei nesutampa reiškias tame intervale yra šaknis.

Metodo kodas:

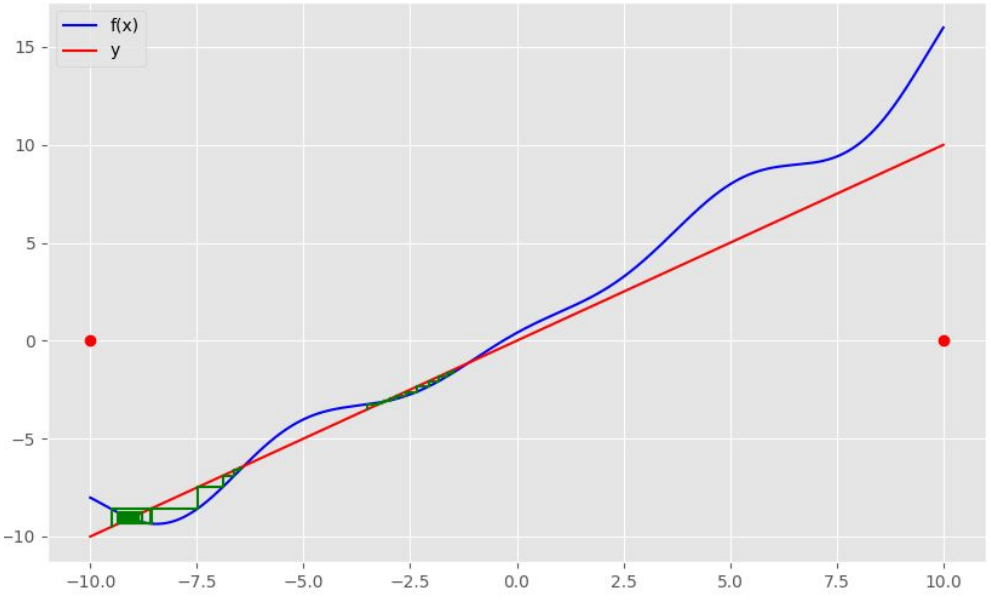
```
def scan(f, start_x, end_x):
    step = (end_x - start_x) / 40
    x = start_x
    intervals = []

    while x < end_x:
        x_next = x + step
        if (f(x) > 0 > f(x_next)) or (f(x_next) > 0 > f(x)):
            intervals.append([x, x_next])

        x = x_next

    return intervals
```

4 pav. Skenavimo nekintančiu žingsniu kodas

Paprastųjų iteracijų metodas:



5 pav. f(x) ir y=x grafikai



6 pav. g(x) ir y=x grafikai

f(x) šaknų radimo metode naudojama: alpha = -10, max_iterations = 1000, eps=1e-4

Paprastųjų iteracijų metodas

INTERVALAS	TIKSLUMAS	ITERACIJŲ SKAIČIUS	ŠAKNIS
[-1.1790000000000003 , -0.40400000000000025]	6.133895242754761e-05	9	-0.9448762649286857
[-0.40400000000000025, 0.3709999999999998]	4.971583691383216e-05	10	0.023724738020666613
[1.9209999999999998 , 2.6959999999999997]	8.818524356613011e-05	20	2.508624779060542
[3.4709999999999996 , 4.2459999999999996]	8.536718436680246e-05	19	3.600290416866638
[4.2459999999999996 , 5.021]	5.0557521611871437e-05	7	4.624287007421092

7 pav. Gautos f(x) funkcijos šaknis naudojant paprastųjų iteracijų metodą

g(x) šaknų radimo metode naudojama: alpha = -10, max_iterations = 1000, eps=1e-4

Paprastųjų iteracijų metodas

INTERVALAS	TIKSLUMAS	ITERACIJŲ SKAIČIUS	ŠAKNIS
[-9.500025000000278 , -9.000050000000556]	9.638714801418757e-05	68	-9.06341686644354
[-6.500175000001943 , -6.00020000000222]	9.845778482819867e-05	71	-9.063417834488366
[-3.5003250000036052 , -3.0003500000038823]	8.042600319813076e-05	10	-3.1097886932175487
[-3.5003250000036052 , -3.0003500000038823]	3.8166888107760144e-05	10	-6.395580090605429

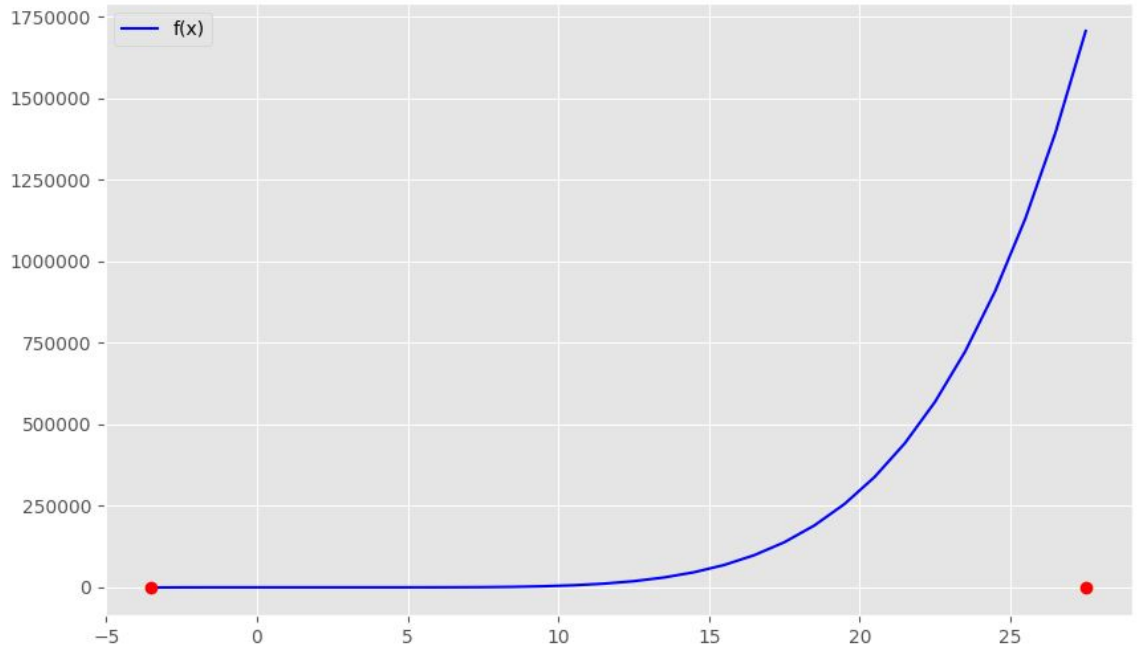
8 pav. Gautos g(x) funkcijos šaknys naudojant paprastųjų iteracijų metodą

Metodo kodas:

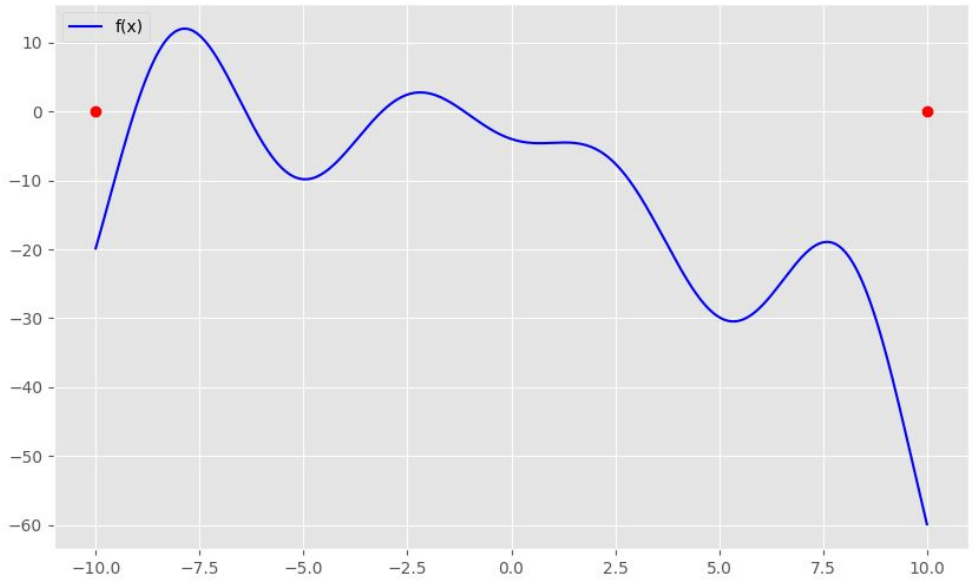
```
def simple_iteration_method(self):
    for interval in self.intervals:
        x = interval[0]
        iteration = 0  # Iteration count
        prec = self.precision
        while prec > self.eps and x >= self.function_x_values[0]:
            iteration += 1
            if iteration == self.max_iterations:
                print("Reached max iterations limit")
                return

            if x > self.function_x_values[-1]:
                print('X out of bounds')
                return

            next_x = (self.function(x) / self.alpha) + x
            prec = abs(x - next_x)
            x = next_x
```

9 pav. Paprastųjų iteracijų metodo kodas

Niutono (liestinių) metodas:



10 pav. f(x) grafikas



11 pav. g(x) frafikas

f(x) šaknų radimo metode naudojama: max_iterations = 1000, eps=1e-4

Niutono (liestinių) metodas

INTERVALAS	TIKSLUMAS	ITERACIJŲ SKAIČIUS	ŠAKNIS
[-1.1790000000000003 , -0.40400000000000025]	6.708589108872509e-05	4	-0.9448602089690443
[-0.40400000000000025, 0.3709999999999998]	1.2660190468888333e-08	3	0.023753472522068893
[1.9209999999999998 , 2.6959999999999997]	2.5446787343952337e-08	4	2.5088146084427816
[3.4709999999999996 , 4.2459999999999996]	1.942263350862561e-05	3	3.6004987207595436
[4.2459999999999996 , 5.021]	4.317478374815664e-08	3	4.6242933996619575

12 pav. Gautos f(x) funkcijos šaknys naudojant Niutono (liestinių) metodą

g(x) šaknų radimo metode naudojama: max_iterations = 1000, eps=1e-4

Niutono (liestinių) metodas

INTERVALAS	TIKSLUMAS	ITERACIJŲ SKAIČIUS	ŠAKNIS
[-9.500025000000278 , -9.000050000000556]	1.5303070185268552e-05	3	-9.06337180372566
[-6.500175000001943 , -6.00020000000222]	8.162260130006871e-07	3	-6.395585739123464
[-3.5003250000036052 , -3.0003500000038823]	1.6755191842410255e-05	3	-3.109728213312953
[-1.5004250000047135 , -1.0004500000049905]	5.29539716742633e-07	3	-1.1336261249556197

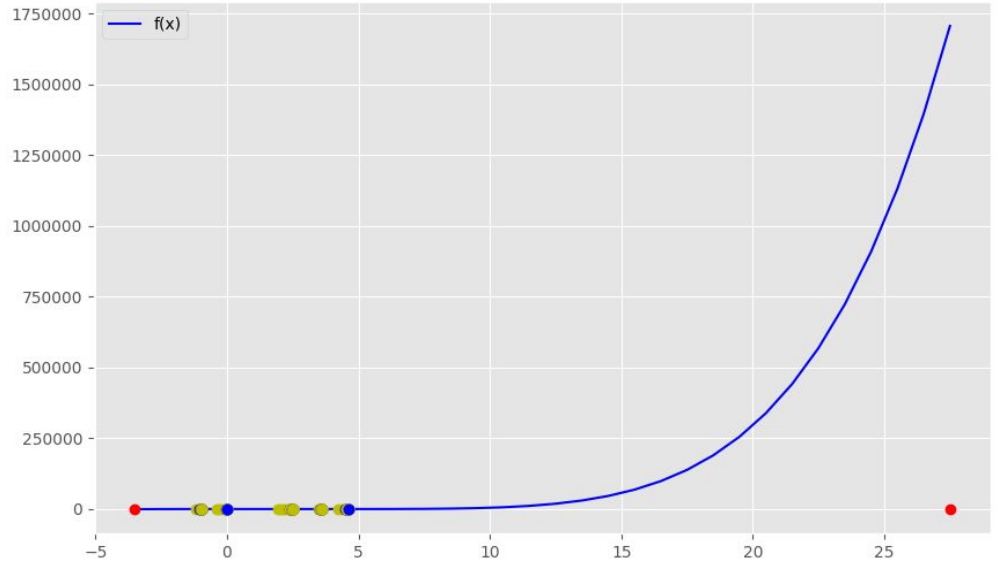
13 pav. Gautos g(x) funkcijos šaknys naudojant Niutono (liestinių) metodą

Metodo kodas:

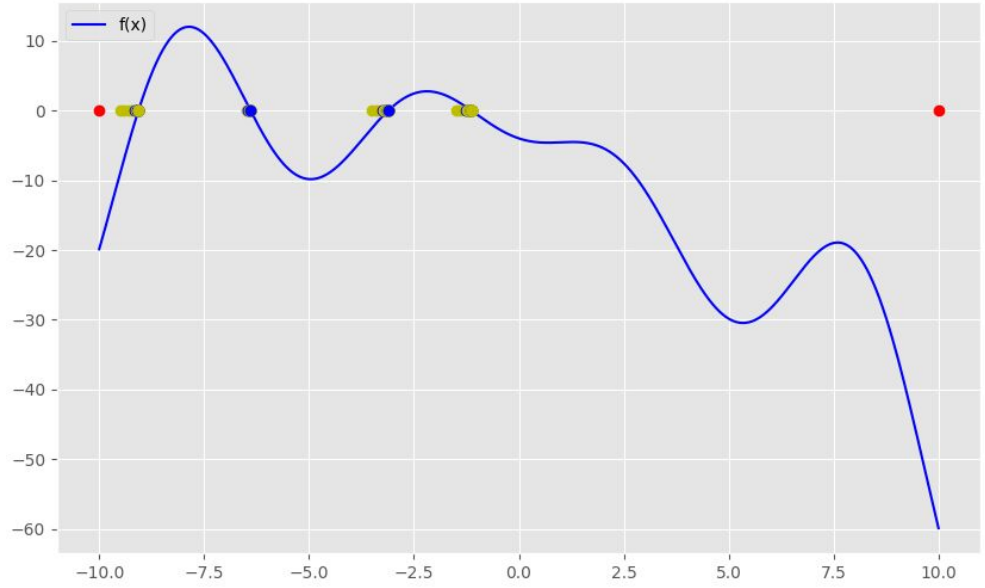
```python
def newton_method(self):
    for interval in self.intervals:
        prec = 0.1
        x = (interval[0] + interval[-1]) / 2
        current_iteration = 0

        while prec > self.eps:
            if current_iteration >= self.max_iterations:
                print("Reached max iterations limit")
                return

            next_x = (x - self.function(x) / self.derivative(x))
            prec = abs(next_x - x)

            x = next_x
            current_iteration += 1
```

14 pav. Niutono (liestinių) metodo kodas

Skenavimo su mažėjančiu žingsniu metodas:



15 pav. f(x) grafikas, melyni taškai intervalo galai, geltonas sumažinto intervalo pradžia



16 pav. g(x) grafikas, melyni taškai intervalo galai, geltonas sumažinto intervalo pradžia

f(x) šaknų radimo metode naudojama: max_iterations = 1000, eps=1e-4, žingsnis mažinimas 2 kartus, pradinis žingsnis step = 0.1

Skenavimo su mažėjančiu žingsniu

INTERVALAS	TIKSLUMAS	ITERACIJŲ SKAIČIUS	ŠAKNIS
[-1.1790000000000003 , -0.40400000000000025]	3.842383258303994e-05	48	-0.9448569335937502
[-0.40400000000000025, 0.3709999999999998]	3.369255118940151e-05	41	0.023734374999999745
[1.9209999999999998 , 2.6959999999999997]	6.862807345711652e-05	39	2.508890625
[3.4709999999999996 , 4.2459999999999996]	1.8993442875425703e-05	35	3.6005898437499995
[4.2459999999999996 , 5.021]	2.218547740118204e-05	43	4.624271484375

17 pav. Gautos f(x) funkcijos šaknys naudojant skenavimo su mažėjančiu žingsniu metodą

g(x) šaknų radimo metode naudojama: max_iterations = 1000, eps=1e-4, žingsnis mažinimas 2 kartus, pradinis žingsnis step = 0.1

Skenavimo su mažėjančiu žingsniu

INTERVALAS	TIKSLUMAS	ITERACIJŲ SKAIČIUS	ŠAKNIS
[-9.500025000000278 , -9.000050000000556]	2.9766986673784857e-05	51	-9.063367285156527
[-6.500175000001943 , -6.00020000000222]	6.841156516168567e-06	35	-6.395682812501942
[-3.5003250000036052 , -3.0003500000038823]	2.1683300363450897e-05	49	-3.109748828128605
[-1.5004250000047135 , -1.0004500000049905]	8.410199090480575e-06	33	-1.133237500004713

18 pav. Gautos g(x) funkcijos šaknys naudojant skenavimo su mažėjančiu žingsniu metodą

Metodo kodas:

```
def scanning_method(self):
    for interval in self.intervals:
        prec = 0.1
        step = 0.1

        current = interval[0]
        is_positive = self.function(current) > 0
        iteration_counter = 0

        while prec > self.eps:
            if iteration_counter >= self.max_iterations:
                print("Reached max iterations limit")
                return
            y = self.function(current)
            if is_positive is not (y > 0):
                current -= step
                step /= 2

            current += step

            prec = abs(y)
            iteration_counter += 1
            is_positive = y > 0
```

19 pav. Skenavimo su kintančiu žingsniu metodo kodas

2.2. Netiesinės lygties sprendimas

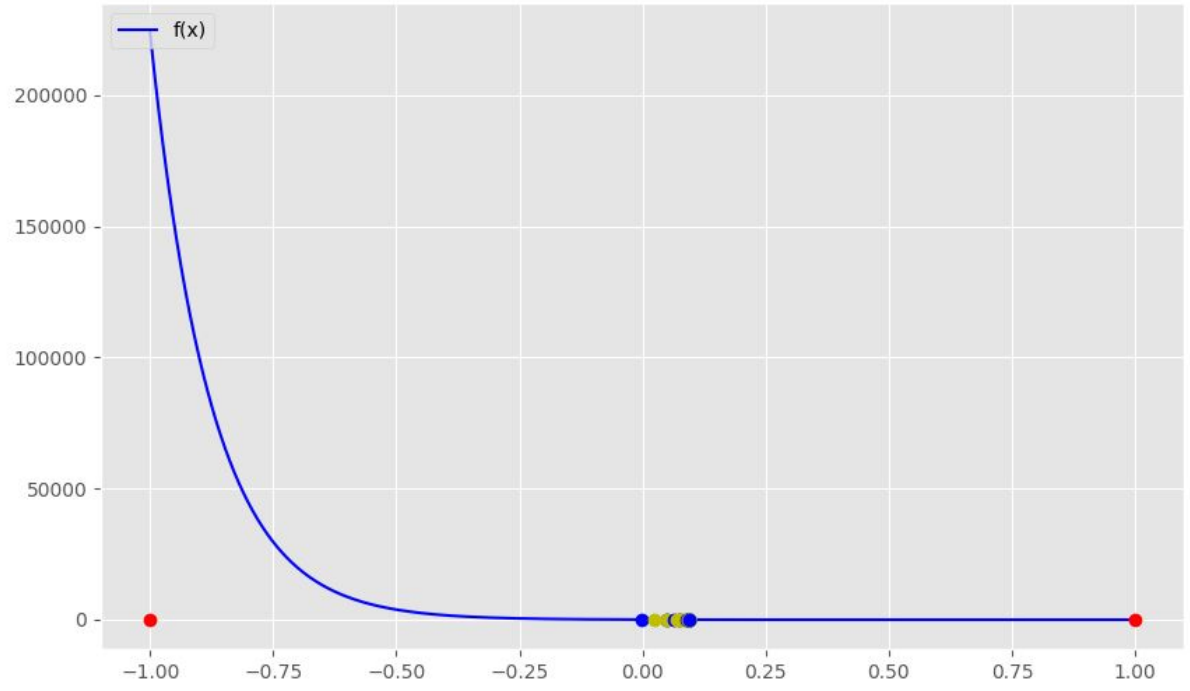
Vertikaliai į viršų iššauto objekto greitis užrašomas dėsniu $v(t) = v_0 e^{-\frac{ct}{m}} + \frac{mg}{c}\left(e^{-\frac{ct}{m}} - 1\right)$, čia $g = 9{,}8\ \mathrm{m/s^2}$, pradinis greitis v_0, objekto masė m. Koks pasipriešinimo koeficientas c veikia objektą, jei žinoma, kad po t_1 laiko nuo iššovimo jo greitis lygus v_1?

Varianto Nr.	v_0, m/s	m, kg	t_1, s	v_1, m/s
12	80	0,5	4	10

20 pav. Užduotis

Šaknies radimui pasirinkau skenavimo su mažėjančių žingsniu metodą

Šios funkcijos grafikas:



21 pav. Funkcijos grafikas, mėlyni taškai intervalo galai, geltonas taškas sumažinto intervalo pradžia

Rasti šaknį naudojau: max_iterations = 1000, step = 0.01, eps=1e-4, intervalas (-1, 1)

Skenavimo su mažėjančiu žingsniu			
INTERVALAS	TIKSLUMAS	ITERACIJŲ SKAIČIUS	ŠAKNIS
[0.049475000000001546, 0.09945000000000159]	3.418645203367987e-05	59	0.09478292236328283

22 pav. Gautos šaknies intervalas, tikslumas, iteracijų skaičius, pati šaknis

Pasipriešinimo koeficientas c=0.09478292236328283

3. Išvados

Pagal gautus šaknų rezultatus matome, kad Niutono (liestini) metodas yra greičiausias, nes užtrunka mažiausią iteracijų skaičių. Niutono (liestinių) metodas taip pat yra ir tiksliausias. Trūkumas - reikia turėti funkcijos išvestinę.
Kitas pagal greitį yra paprastųjų iteracijų metodas. Bet jame greitis labai priklauso nuo gerai pasirinktos alfa.
Skenavimo metodas su mažėjančiu žingsniu gavosi lėčiausias. Jo tikslumas panašus į paprastųjų iteracijų metodo.