KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

TAIKOMOSIOS INFORMATIKOS KATEDRA

1 Laboratorinis darbas
Nr. 10

Atliko:
IFK-8 grupės stud.
Tomas Odinas

Priėmė
lekt. Andrius Kriščiūnas

KAUNAS, 2020

TURINYS

1. UŽDUOTIS 2
2. PAGRINDINĖ DALIS 3
   2.1 "Grubus" daugianario šaknų įvertis 3
   2.2 "Tikslesnis" daugianario šaknų įvertis 3
   2.3 Daugianario šaknų radimas 5
   2.4 Transcendentinės funkcijos šaknų radimas 8
   2.5 Tekstinio uždavinio sprendimas 11
3. IŠVADOS 12
4. PRIEDAI 13
   4.1 Realizuotų algoritmų programinio kodo fragmentai: 13
   4.2 Nuoroda į pilną programos išeities kodą 14

1. UŽDUOTIS

1. Išspręsti netiesines lygtis:

a) Daugianaris $f(x) = 0$
b) Transcendentinė funkcija $g(x) = 0$

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, 5

Pav. 1

2. Pagal pateiktą uždavinio sąlygą sudaryti netiesinę lygtį ir pasirinktu skaitiniu metodu ją išspręsti.

Uždavinys variantams 6-10

Krentančio parašiutininko greitis užrašomas dėsniu $v(t) = \frac{mg}{c}\left(1 - e^{-\left(\frac{c}{m}\right)t}\right)$, čia $g = 9{,}8\ \mathrm{m/s^2}$, parašiutininko masė m. Koks pasipriešinimo koeficientas c veikia parašiutininką, jei žinoma, kad po t_1 laisvojo kritimo, jo greitis lygus v_1?

Varianto Nr.	m, kg	t_1, s	v_1, m/s
10	60	4	30

Pav. 2

2. PAGRINDINĖ DALIS

2.1 "Grubus” daugianario šaknų įvertis

Užduotyje nurodytas daugianaris:

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

Grubus įvertis apskaičiuojamas pagal formulę:

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

Įsistatę koeficientus iš uždavinio lygties gauname:

$$R = 1 + \frac{5{,}26}{0{,}25} = 22{,}04$$

Vadinasi grubus lygties šaknų įvertis yra:

$$-22{,}04 < x < 22{,}04$$

2.2 "Tikslesnis” daugianario šaknų įvertis

Tikslesnis įvertis apskaičiuojamas pagal formulę:

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

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

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

Neigiamoms šaknims:
Nagrinėjamas daugianaris $f(-x)$, jei n lyginis ir $-f(-x)$kei n nelyginis.

Pilna įverčio formulė:

$$-\min\left(R, R_{neig}\right) \leq x \leq \min\left(R, R_{teig}\right)$$

Skaičiuojame R_{teig}:

$$k = 5 - 3 = 2$$

$$B = 5{,}26$$

$$R_{teig} = 1 + \sqrt{\frac{5{,}26}{0{,}25}} \approx 5{,}587$$

Skaičiuojame R_{neig}:

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

$$k = 5 - 4 = 1$$

$$B = 1{,}91$$

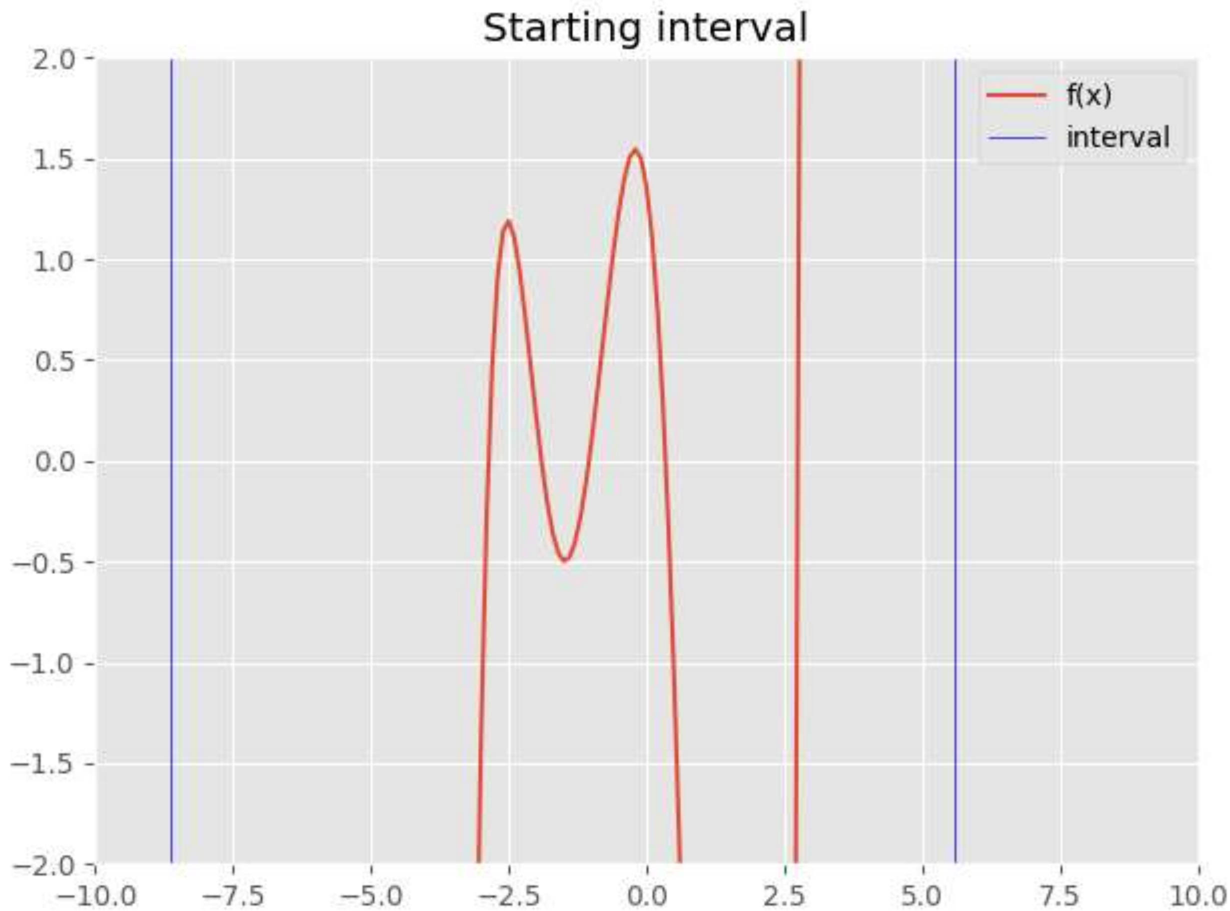
$$R_{neig} = 1 + \frac{1{,}91}{0{,}25} = 8{,}64$$

Galutinis įvertis:

$$-8{,}64 \leq x \leq 5{,}587$$

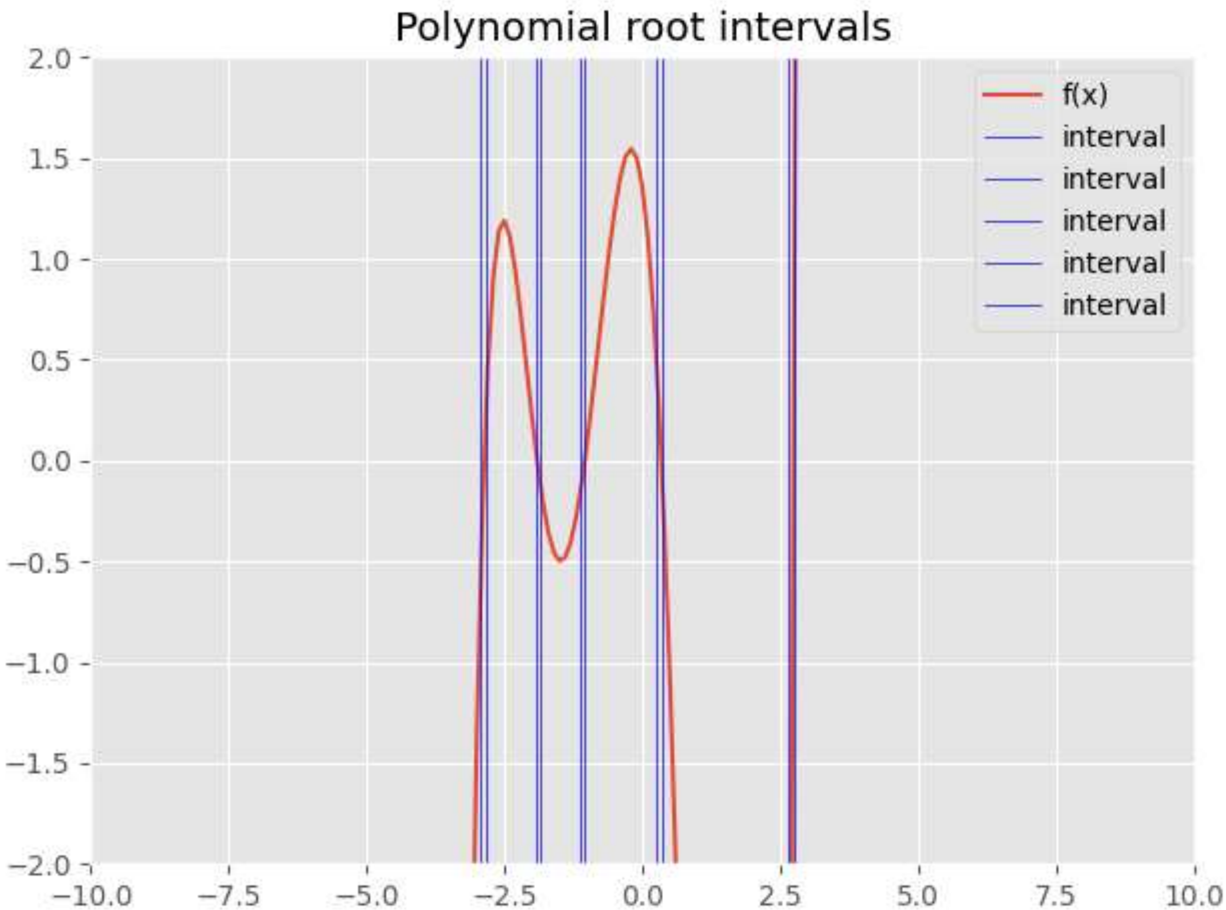
2.3 Daugianario šaknų radimas

Daugianarį $f(x)$ pavaizduojame nustatytame šaknų įverčio intervale:



Pav. 3

Naudodami skanavimo algoritmą su žingsniu 0,1 išskiriame daugianario šaknų intervalus:



Pav. 4

Gautas šaknis tiksliname „Stygų“, „Kirstinių“ ir „Skenavimo su mažėjančiu žingsniu“ metodais.

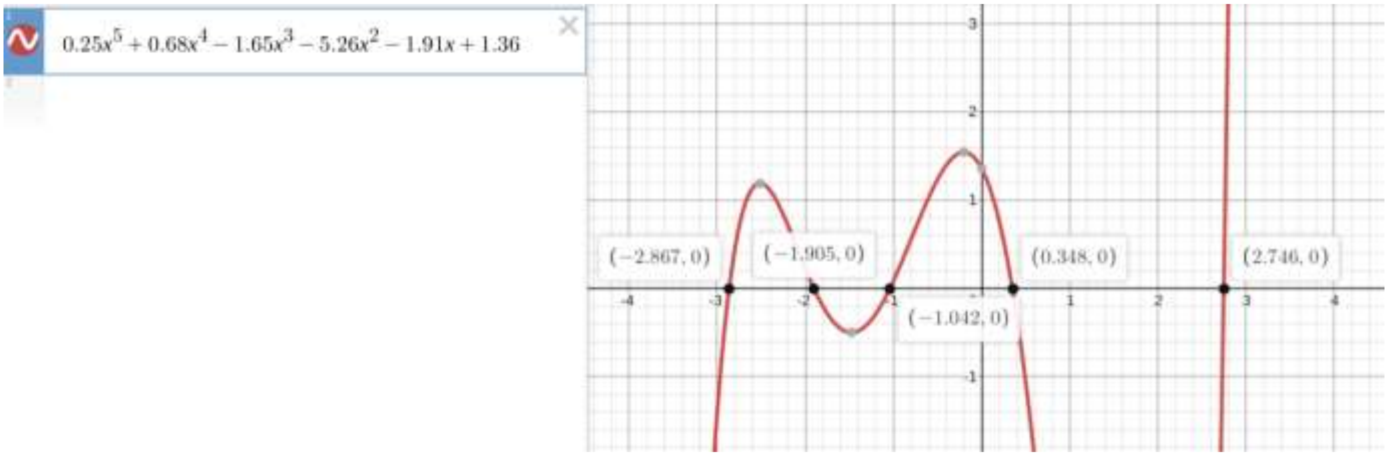
Gauti rezultatai pateikti žemiau esančioje lentelėje (Pastaba: algoritmų nutraukimo sąlyga – funkcijos reikšmės modulis taške mažesnis už 10^{-8}).

Metodo pavadinimas	**Intervalas**	**Šaknis (suapvalinta)**	**Funkcijos reikšmė šaknies taške (paklaida)**	**Iteracijų skaičius**
Stygų	[-2.940000000000016, -2.840000000000016]	-2.867	$3.824 * 10^{-9}$	9
Kirstinių			$3.824 * 10^{-9}$	9

Skenavimo su mažėjančiu žingsniu			$-9.414 * 10^{-9}$	39
Stygų	[-1.940000000000015, -1.840000000000015]	-1.905	$-1.703 * 10^{-9}$	5
Kirstinių			$-1.703 * 10^{-9}$	5
Skenavimo su mažėjančiu žingsniu			$4.031 * 10^{-9}$	38
Stygų	[-1.1400000000000143, -1.0400000000000142]	-1.042	$-9.641 * 10^{-10}$	3
Kirstinių			$1.753 * 10^{-9}$	6
Skenavimo su mažėjančiu žingsniu			$-5.875 * 10^{-9}$	40
Stygų	[0.25999999999998563, 0.35999999999998566]	0.348	$1.798 * 10^{-10}$	5
Kirstinių			$8.118 * 10^{-9}$	7
Skenavimo su mažėjančiu žingsniu			$2.425 * 10^{-9}$	35
Stygų	[2.659999999999987, 2.759999999999987]	2.746	$-3.639 * 10^{-9}$	5
Kirstinių			$7.103 * 10^{-9}$	8
Skenavimo su mažėjančiu žingsniu			$-9.934 * 10^{-9}$	44

Lentelė 1

Gautų šaknų patikrinimui to paties daugianario šaknis randame „Desmos Graphing Calculator“ įrankio pagalba:



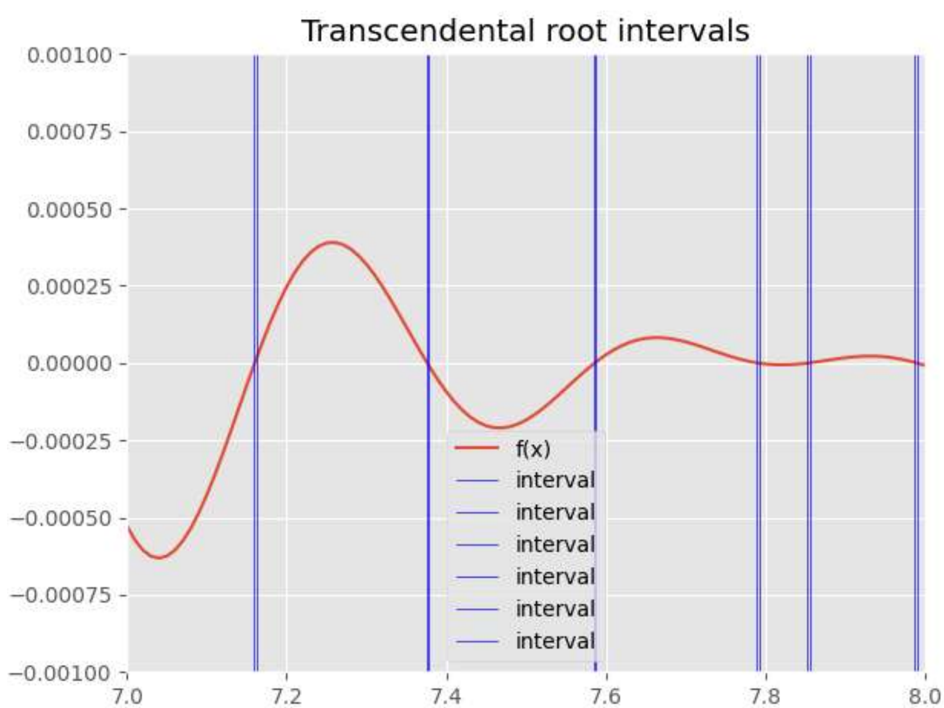
Pav. 5

2.4 Transcendentinės funkcijos šaknų radimas

Užduotyje duota transcendentinė funkcija:

$$g(x) = \mathrm{e}^{-\mathrm{x}}\cos(x)\sin(x^2 - 1)\,; 7 \leq x \leq 8$$

Pavaizduojame funkciją nurodytame intervale ir atskiriame šaknų intervalus naudojant žingsnį 0,03:



Pav. 6

Gautas šaknis tiksliname „Stygų“, „Kirstinių“ ir „Skenavimo su mažėjančiu žingsniu“ metodais.

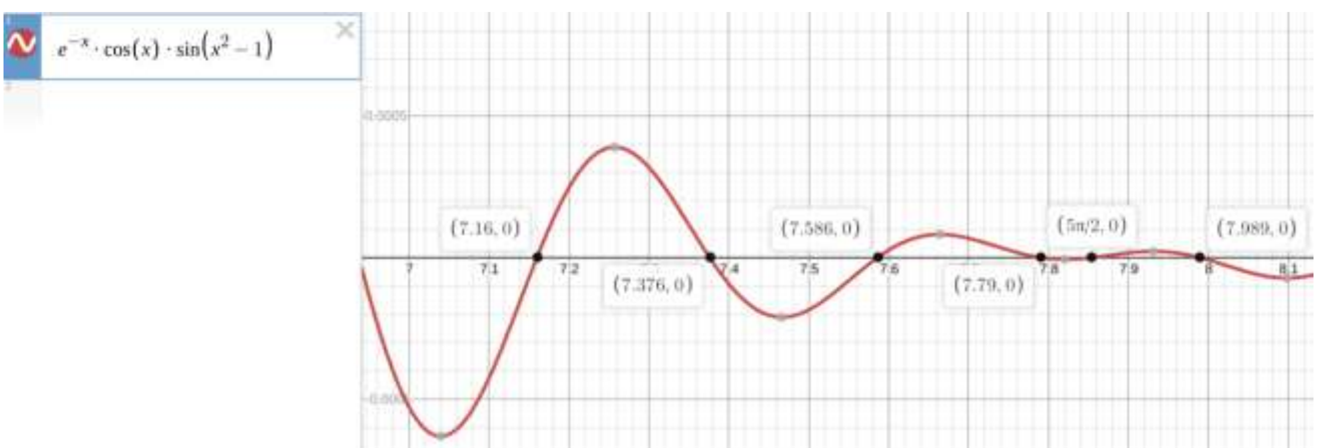
Gauti rezultatai pateikti žemiau esančioje lentelėje (Pastaba: algoritmų nutraukimo sąlyga – funkcijos reikšmės modulis taške mažesnis už 10^{-8}).

Metodo pavadinimas	**Intervalas**	**Šaknis (suapvalinta)**	**Funkcijos reikšmė šaknies taške (paklaida)**	**Iteracijų skaičius**
Stygų	[7.159000000000006, 7.162000000000006]	7.160	$6.392 * 10^{-11}$	2
Kirstinių			$6.392 * 10^{-11}$	2
Skenavimo su mažėjančiu žingsniu			$-5.220 * 10^{-9}$	23
Stygų	[7.375000000000014, 7.378000000000014]	7.376	$-8.118 * 10^{-11}$	2
Kirstinių			$-8.118 * 10^{-11}$	2
Skenavimo su mažėjančiu žingsniu			$5.461 * 10^{-9}$	22
Stygų	[7.585000000000022, 7.588000000000022]	7.586	$9.582 * 10^{-11}$	2
Kirstinių			$9.582 * 10^{-11}$	2
Skenavimo su mažėjančiu žingsniu			$-7.134 * 10^{-9}$	19
Stygų	[7.78900000000003, 7.79200000000003]	7.790	$-3.471 * 10^{-10}$	2
Kirstinių			$-3.471 * 10^{-10}$	2

Skenavimo su mažėjančiu žingsniu			$5.730 * 10^{-9}$	20
Stygų	[7.852000000000032, 7.855000000000032]	7.854	$-6.185 * 10^{-9}$	1
Kirstinių			$-6.185 * 10^{-9}$	1
Skenavimo su mažėjančiu žingsniu			$-9.279 * 10^{-9}$	11
Stygų	[7.987000000000037, 7.9900000000000375]	7.989	$6.170 * 10^{-9}$	1
Kirstinių			$6.170 * 10^{-9}$	1
Skenavimo su mažėjančiu žingsniu			$4.301 * 10^{-9}$	19

Lentelė 2

Gautų šaknų patikrinimui to pačios funkcijos šaknis randame „Desmos Graphing Calculator“ įrankio pagalba:



Pav. 7

2.5 Tekstinio uždavinio sprendimas

Krentančio parašiutininko greitis užrašomas dėsniu:

$$v(t) = \frac{mg}{c}\left(1 - e^{-\left(\frac{c}{m}\right)t}\right)$$

Į lygtį įrašome žinomus duomenis:

$$v(t) = \frac{60 * 9{,}8}{c}\left(1 - e^{-\left(\frac{c}{60}\right)t}\right) = 30$$

Pertvarkome lygtį, kad būtų galima rasti koeficientą c:

$$v(c) = \frac{60 * 9{,}8}{c}\left(1 - e^{-\left(\frac{c}{60}\right)t}\right) - 30$$

Žemiau esančioje lentelėje pateikti pradiniai metodo parametrai:

Pasirinktas metodas	Stygų
Šaknies atskyrimo intervalas	[0, 20]
Leistina paklaida	10^{-8}

Lentelė 3

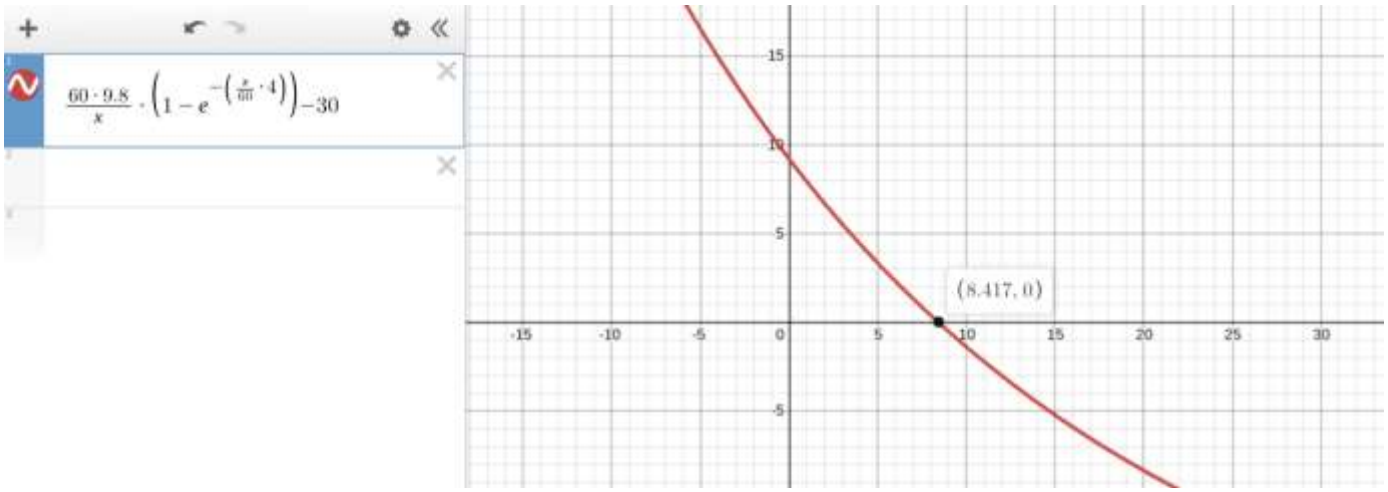
Stygų metodas pasirinktas dėl mažo iteracijų skaičiaus lyginant su skenavimo metodu. Taip pat dėl savybės jog nereikia skaičiuoti funkcijos išvestinės ar jos aproksimacijos kaip tai daroma Niutono (liestinių) ir Kvazi-Niutono (kirstinių) metoduose.

Stygų metodo rezultatai:

Šaknis (suapvalinta)	**Funkcijos reikšmė šaknies taške (paklaida)**	**Iteracijų skaičius**
8.417	$-6.596 * 10^{-9}$	12

Lentelė 4

Grafinis lygties sprendimas:



Pav. 8

3. IŠVADOS

Labai svarbu parinkti tinkamą pradinį artinį ar intervalą, nes nėra garantijų jog metodai konverguos.

Stygų metodas apskaičiuoja stygos susikirtimo su x ašimi tašką remdamasis trikampių panašumu.

Skenavimo algoritmas nėra tinkamas lygtims su keliais nežinomaisiais, nes skenuoti daugiamatę erdvę užtruks santykinai daug laiko.

Kirstinių metodas veiks ir tada, kai nėra žinoma funkcijos išvestinė arba funkcija užrašoma algoritmu, nes naudojama ne pati išvestinė, o jos aproksimacija.

4. PRIEDAI

4.1 Realizuotų algoritmų programinio kodo fragmentai:

Stygų metodas:

```
def chord_method(interval, func, tolerance=1e-8):
    iterations = 0
    x_n = interval[0]
    x_n1 = interval[1]
    x_mid = 0

    while abs(func(x_mid)) > tolerance:
        iterations += 1
        k = abs(func(x_n) / func(x_n1))
        x_mid = (x_n + k * x_n1) / (1 + k)
        if func(x_mid) * func(x_n) > 0:
            x_n = x_mid
        else:
            x_n1 = x_mid

    return iterations, x_mid
```

Kirstinių metodas:

```
def secant_method(starting_points, func, tolerance=1e-8):
    iterations = 0
    x_n = starting_points[0]
    x_n1 = starting_points[1]
    while abs(func(x_n1)) > tolerance:
        iterations += 1
        x_n1 = x_n1 - (x_n1 - x_n) / (func(x_n1) - func(x_n)) * func(x_n1)

    return iterations, x_n1
```

Skenavimo su mažėjančiu žingsniu metodas:

```
def scan_method(interval, func, step=0.1, tolerance=1e-8):
    iterations = 1
    x_n = interval[0]
    x_s = interval[0]

    while (abs(func(x_s))) > tolerance:
        iterations += 1
        x_s += step
        if func(x_s) * func(x_n) < 0:  # detects sign change
            x_s -= step
            step /= 2
            x_n = x_s

    return iterations, x_s
```

4.2 Nuoroda į pilną programos išeities kodą

https://github.com/d0ubletr0uble/Skaitiniai-metodai-ir-algoritmai/blob/master/main.py