KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

(P170B115) Skaitiniai metodai ir algoritmai

Pirmo laboratorinio darbo ataskaita

Atliko:
IFF-8/12 Mykolas Paulauskas

Tikrino:
lekt. Andrius Krisčiūnas

KAUNAS 2020

Turinys

1. Užduotis

1 Išspręskite netiesines lygtis (1 ir 2 lentelės):

a) daugianaris $f(x) = 0$;
b) transcendentinė funkcija $g(x) = 0$.

1. (tik lygčiai su daugianariu $f(x)$) Nustatykite daugianario $f(x)$ šaknų intervalą, taikydami „grubų“ ir tikslesnį įverčius. Grafiškai pavaizduokite apskaičiuotų šaknų intervalo galus.
2. Daugianarį $f(x)$ grafiškai pavaizduokite nustatytame šaknų intervale. Grafiko ašis pakeiskite taip, kad būtų aiškiai matomos daugianario šaknys. Funkciją $g(x)$ grafiškai pavaizduokite užduotyje nurodytame intervale.
3. Naudodami skenavimo algoritmą su nekintančiu skenavimo žingsniu atskirkite šaknų intervalus. Daugianariui skenavimo intervalas parenkamas pagal įverčių reikšmes, funkcija skenuojama užduotyje nurodytame intervale. Šaknies atskyrimo intervalai naudojami kaip pradiniai intervalai (artiniai) šaknų tikslinimui.
4. Skenavimo metodu atskirtas daugianario ir funkcijos šaknis tikslinkite užduotyje nurodytais metodais. Užrašykite skaičiavimų pabaigos sąlygas. Skaičiavimų rezultatus pateikite lentelėje, kurioje nurodykite šaknies tikslinimui naudojamą metodą, pradinį artinį ar intervalą, gautą sprendinį (šaknį), tikslumą, iteracijų skaičių. Palyginkite, kuris metodas randa sprendinį su mažesniu iteracijų skaičiumi.
5. Gautas šaknų reikšmes patikrinkite naudodami išorinius išteklius (pvz., MATLAB funkcijas roots arba fzero, tinklapį wolframalpha.com ir t.t.).

2 Pagal pateiktą uždavinio sąlygą (3 lentelė) sudarykite netiesinę lygtį ir pasirinktu skaitiniu metodu iš 1 lentelės ją išspręskite. Ataskaitoje pateikite pradinius metodo parametrus (metodo žingsnį, pradinį artinį, izoliacijos intervalą ir pan.), iteracijų pabaigos sąlygą, tikslumą, gautą lygties sprendinį ir sudarytos funkcijos reikšmę, argumentus, kodėl pasirinkote šį metodą. Pateikite grafinį lygties sprendimą.

17	$1.03x^5 - 2.91x^4 - 1.44x^3 + 5.56x^2 - 0.36x - 1.13$	$\sin(x)\ln(x) - \frac{x}{6}; 1 \le x \le 20$	1, 3, 5

pav. 1 Man priskirto 17 varianto pirma užduotis

Pagal varianto numerį priskirti tikslinimo metodai:

- Stygū
- Niutono
- Skenavimas mažėjančiu žingsniu

Uždavinys variantams 16-20

T_0 temperatūros kūnas patalpinamas į aplinką, kurios temperatūra T_A. Tariama, kad aplinkos temperatūra yra palaikoma išorinių šaltinių ir kūno temperatūra neturi įtakos aplinkos temperatūrai. Kūno temperatūra $T(t)$ užrašoma dėsniu $T(t) = (T_0 - T_A)e^{kt} + T_A$, k – proporcingumo koeficientas. Kam lygus proporcingumo koeficientas k, jeigu žinoma, kad praėjus laikui t_1 kūno temperatūra bus T_1?

Varianto Nr.	T_0, K	T_A, K	t_1, s	T_1, K
16	473	373	20	378
17	400	320	30	344
18	250	320	15	290
19	240	350	15	349
20	380	295	10	320

pav. 2 17 varianto antra užduotis

2. Pirmoji užduotis

Pradžiai duotas f(x) ir g(x) funkcijas Python aplinkoje apsibrėžiu kaip metodus. Tadanaudodamas *sympy* biblioteką apsiskaičiavau abiejų funkcijų išvestines, jos bus reikalingos naudojant Niutono tikslinimo metodą.

```
# Pirma funkcija (f(x))
def fx(ax):
    return 1.03*ax**5 - 2.91*ax**4 - 1.44*ax**3 + 5.56*ax**2 - 0.36*ax -1.13

#f(x) isvestine
fxD = sympy.lambdify(sx, (1.03*sx**5 - 2.91*sx**4 - 1.44*sx**3 + 5.56*sx**2 - 0.36*sx - 1.13).diff(), 'numpy')

#Antra funkcija g(x)
def gx(ax):
    return np.sin(ax) * np.log(ax) - ax/6

#g(x) isvestine
gxD = sympy.lambdify(sx, (sympy.sin(sx) * sympy.log(sx) - sx/6).diff(), 'numpy')
```

pav. 3 abi funkcijos ir jų išvestinės

- Intervalų skaičiavimas

Grubų intervalą skaičiavau pagal šią formulę:

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

$$\max_{0\le i\le n-1}|a_i| = 5.56$$
$$a_n = 1.03$$
$$|x| < 1 + \frac{5.56}{1.03} = R$$
$$-6{,}4 < x < 6{,}4$$

Toliau skaičiavau tikslesnį intervalą pagal šias formules:

Teigiamoms šaknims

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

Neigiamoms šaknims vietoje daugianario *f(x)* imti *f(-x)* . Kai *n* nelyginis, imti *–f(-x)*, kad koeficientas prie aukiščiausio laipsnio taptų teigiamas.
Apskaičiuoti R_{neig} pagal tas pačias formules, kaip ir R_{teig}

$$B = \max_{0\le i\le n-1}(|a_i|, a_i < 0) = 2{,}91$$
$$k = n - \max_{0\le i\le n-1}(i, a_i < 0) = 1$$
$$R_{teig} = 1 + \sqrt[1]{\frac{2.91}{1.03}} = 3.83$$

Kadangi mano daugianaryje *n* yra nelyginis vietoj funkcijos *f(x)* turiu imti *-f(-x)*. Perdaryta funkcija atrodo taip:

$$1{,}03x^5 + 2{,}91x^4 - 1.44x^3 - 5.56x^2 - 0.36x + 1.13$$

$$B = \max_{0 \le i \le n-1} (|a_i|, a_i < 0) = 5.56$$

$$k = n - \max_{0 \le i \le n-1} (i, a_i < 0) = 2$$

$$R_{neig} = 1 + \sqrt[1]{\frac{2.91}{1.03}} = 3.21$$

$$-\min(R, R_{neig}) = -3.32$$

$$\min(R, R_{teig}) = 3.83$$

Galiausiai gauname, kad tikslus intervalas yra lygus:

$$-3{,}32 \le x \le 3{,}83$$

- Skenavimas nekintančiu žingsniu

Skenavimas nekintančiu žingsniu yra ganėtinai nesudėtingas algoritmas, kurio paskirtis yra išskirti mažesniu intervalus, kuriuose būtų funkcijos šaknis. Šiam metodui yra paduodama funkcija, kurios šaknų ieškome, pradinis taškas nuo kurio skenuos, pabaigos taškas iki kurio skenuos ir žingsnis kuriame bus ieškoma ar yra šaknis.

```
#Skenavimas nekintanciu zingsniu
def scanFixed(func, xFrom, xTo, step):
    intervals = []
    current = xFrom

    while current <= xTo:                          #Skenavimas
        left = current
        right = current + step
        current += step

        if np.sign(func(left)) != np.sign(func(right)):
            intervals.append((left, right))     #Saknies intervalas

    return intervals
```

pav. 4 Skenavimas nekintančiu žingsniu

- Stygų metodas

Taipogi pakankamai paprastas algoritmas, kurio paskirtis yra ieškoti šaknies vertės duotajame intervale. Metodui padavus funkciją, intervalo pradžios ir pabaigos tašką bei tikslumą (kuris mano kode pagal nutylėjimą yra *1e-8*) *while* cikle metodas brėžia stygą tarp funkcijos verčių intervalo galuose. Kur styga kerta x ašį yra skaitomas vidurio tašku, tada tikrindamas vidurio taško ženklus su intervalo galų ženklais vieną iš galų pritraukia prie vidurio. Veiksmai kartojami kol vidurio taško vertė yra mažesnė nei nustatyto tikslumo.

```python
#Tikslinimas stygu metodu
def stringAdjusment(func, xFrom, xTo, precision = 1e-8):
    xLeft = xFrom
    xRight = xTo

    iterations = 0

    while True:
        iterations += 1

        k = abs(func(xLeft)/func(xRight))
        xMid = (xLeft + k * xRight) / (1 + k)

        if np.sign(func(xMid)) == np.sign(func(xLeft)):
            xLeft = xMid
        else:
            xRight = xMid

        if abs(func(xMid)) < precision:
            return xMid, iterations
```

pav. 5 stygų tikslinimas

- Niutono metodas

Niutono metodas priima skaičiuojama matematinę funkciją, jos išvestinę, pradžios ir galo taškus(skenavimo intervalas) ir tikslumą. Vykdomas ciklas iki nustatyto maksimalaus iteracijų kiekio (šiuo atveju 40). Cikle yra ieškoma dabartinės x iteracijos funkcinė ir išvestinė reikšmė. Jeigu išvestinė reikšmė 0 arba funkcinė reikšmė atitinka norimą tikslumą, tada rastas atsakymas, kitu atveju perskaičiuojamas x kitai iteracijai.

```python
#Tikslinimas niotono(liestiniu) metodu
def newtonAdjusment(func, funcD, xFrom, xTo, precision = 1e-8):
    xo = xFrom
    iterations = 0

    for i in range(0, MAX_ITER):
        iterations += 1

        if abs(func(xo)) < precision:
            return xo, iterations

        if funcD(xo) == 0:
            return xo, iterations

        xo = xo - (func(xo)/funcD(xo))

    return None, iterations
```

- Skenavimas mažėjančiu žingsniu

Veikia panašiai kaip skenavimas nemažėjančiu žingsniu, tačiau šis metodas atradęs šaknį intervale žingsnį sumažina (šiuo atveju 4 kartus) ir toliau tokiu pačiu principu ieško šaknies. Procesas yra kartojamas tol kol galiausiai žingsnis tampa mažesnis nei nurodytas tikslumas.

```python
#Skenavimas mazejanciu zingsniu
def scanDecreasing(func, xFrom, xTo, step, precision = 1e-8):

    width = step

    left = xFrom
    right = xFrom + width

    iterations = 0

    while width > precision:                        #Skenavimas
        iterations += 1

        if np.sign(func(left)) == np.sign(func(right)): #Saknies intervale nera
            left = right
            right = right + width
        else:                                       #Saknis yra intervale
            width = width / 4                       #Zingsnis mazinamas
            right = left + width

    return left, iterations
```

- Sprendimas f(x)

Ranka apskaičiuotas tikslus sprendimų intervalas (-3.32; 3.83) yra paduodamas į skenavimo funkciją su nekintančiu žingsniu. Šis algoritmas grąžina visus šaknų intervalus:

$$[(-1.3233721726032561, -1.223372172603256),$$
$$(-0.423372172603256, -0.323372172603256),$$
$$(0.576627827396744, 0.676627827396744),$$
$$(1.2766278273967442, 1.3766278273967443),$$
$$(2.576627827396745, 2.676627827396745)]$$

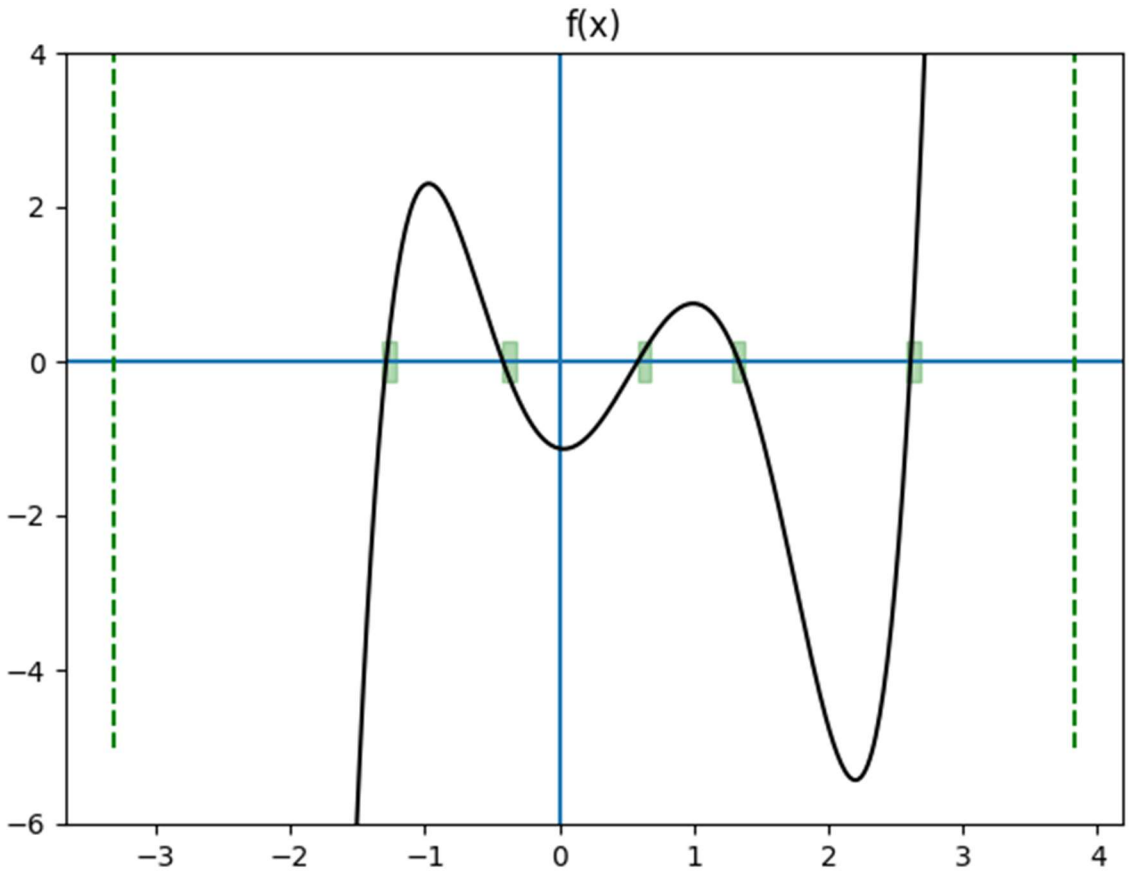
Numpy apskaičiuotos šaknys:

$$[-1.286520634667408,$$
$$-0.41902248780924284,$$
$$0.5837632640352431,$$
$$1.3343225156209342,$$
$$2.6127000612670814]$$

Gautus intervalus tikslinama su duotais metodais:

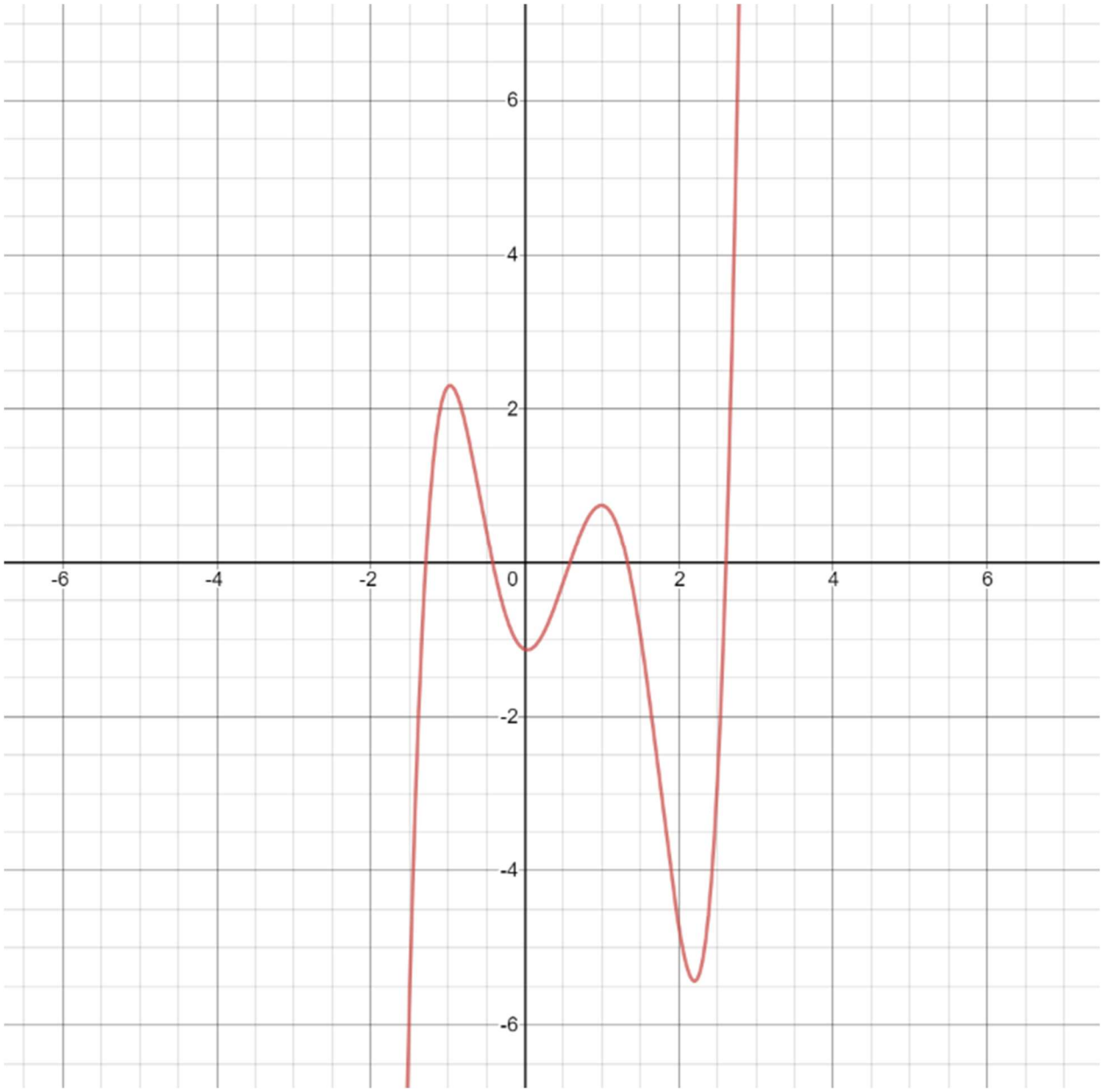
Metodas	Artinys/Intervalas	Šaknis	Tikslumas	Iteracijos
Stygų	(-1.3233; -1.2233)	-1.2865206330645	0.00000001	7
Niutono	-1.3233	-1.2865206354640	0.00000001	4
Skenavimo	(-1.3233; -1.2233)	-1.2865206711384	0.00000001	27
Stygų	(-0.4233; -0.3233)	-0.4190224839289	0.00000001	3
Niutono	-0.4233	-0.4190224879514	0.00000001	3

Skenavimo	(-0.4233; -0.3233)	-0.4190226547809	0.00000001	17
Stygų	(0.5766; 0.6766)	0.58376326523178	0.00000001	3
Niutono	0.5766	0.58376326398010	0.00000001	3
Skenavimo	(0.5766; 0.6766)	0.58376321863209	0.00000001	19
Stygų	(1.2766; 1.3766)	1.33432251286561	0.00000001	6
Niutono	1.2766	1.33432252033240	0.00000001	4
Skenavimo	(1.2766; 1.3766)	1.33432245325123	0.00000001	24
Stygų	(2.5766; 2.6766)	2.61270006043650	0.00000001	8
Niutono	2.5766	2.61270006158267	0.00000001	4
Skenavimo	(2.5766; 2.6766)	2.61269998621022	0.00000001	24



pav. 6Matplotlib nubraižytas f(x) grafikas

Žalia punktyras žymi tikslesnius intervalus. Žali kvadratai žymi funkcijos šaknis gautas skenavimo nekintančiu žingsniu.



pav. 7 desmos.com gautas grafikas f(x)

Niutono metodas yra greičiausias iš trijų metodų, skenavimo metodas gali būti greitesnis optimizuojant žingsnio mažėjimą.

- Sprendimas g(x)

Duotame intervale yra paleidžiamas skenavimo su nekintančiu žingsniu algoritmas. Šis algoritmas grąžina visus šaknų intervalus g(x) funkcijai (gauti rezultatai yra suapvalinti iki dešimčių nes toliau einantys skaičiai yra periodu):

$$[(1.2, 1.3),$$
$$(2.6, 2.7),$$
$$(6.9, 7),$$

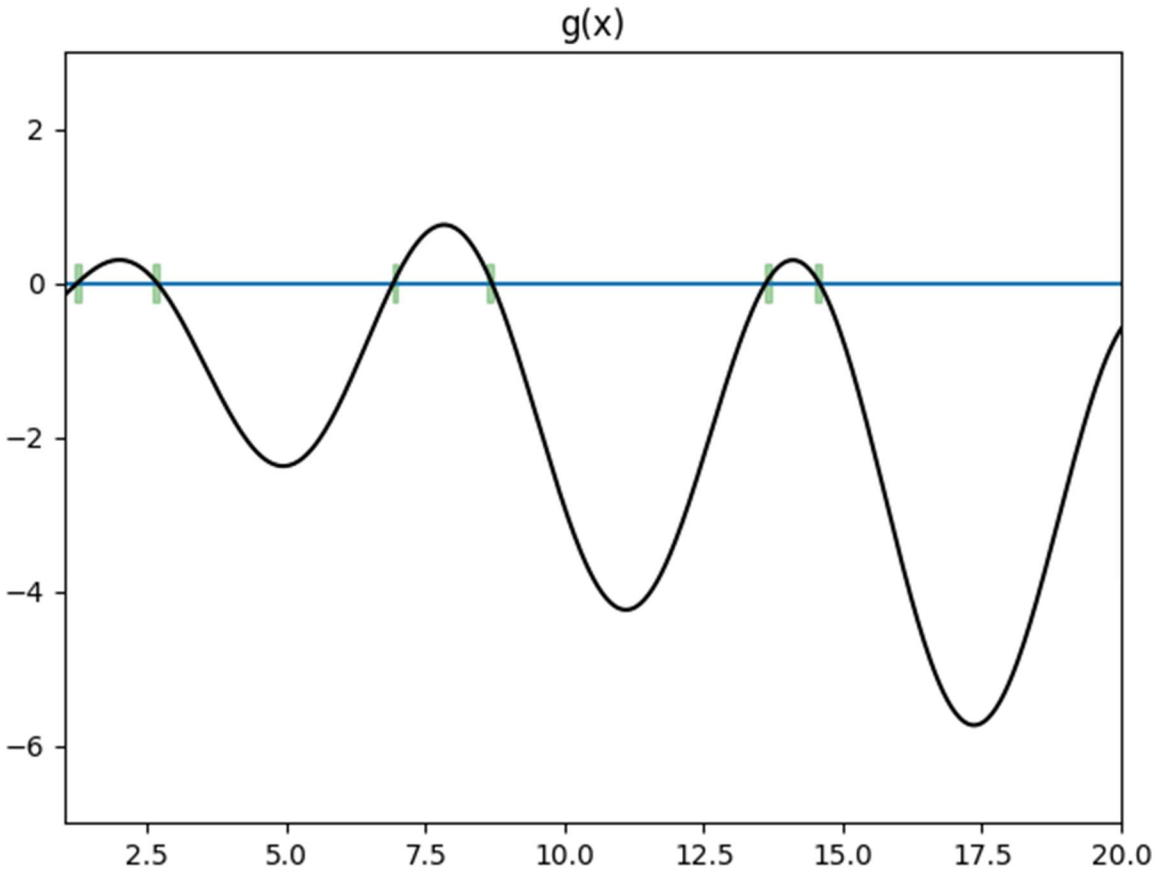
$$(8.6, 8.7),$$
$$(13.6, 13.7),$$
$$(14.5, 14.6)]$$

wolframalpha.com gautos šaknys:

$$[1.24483722360717,$$
$$2.67133437780012,$$
$$6.92207470019942,$$
$$8.69072939602094,$$
$$13.6199907156415,$$
$$14.5729067030846]$$

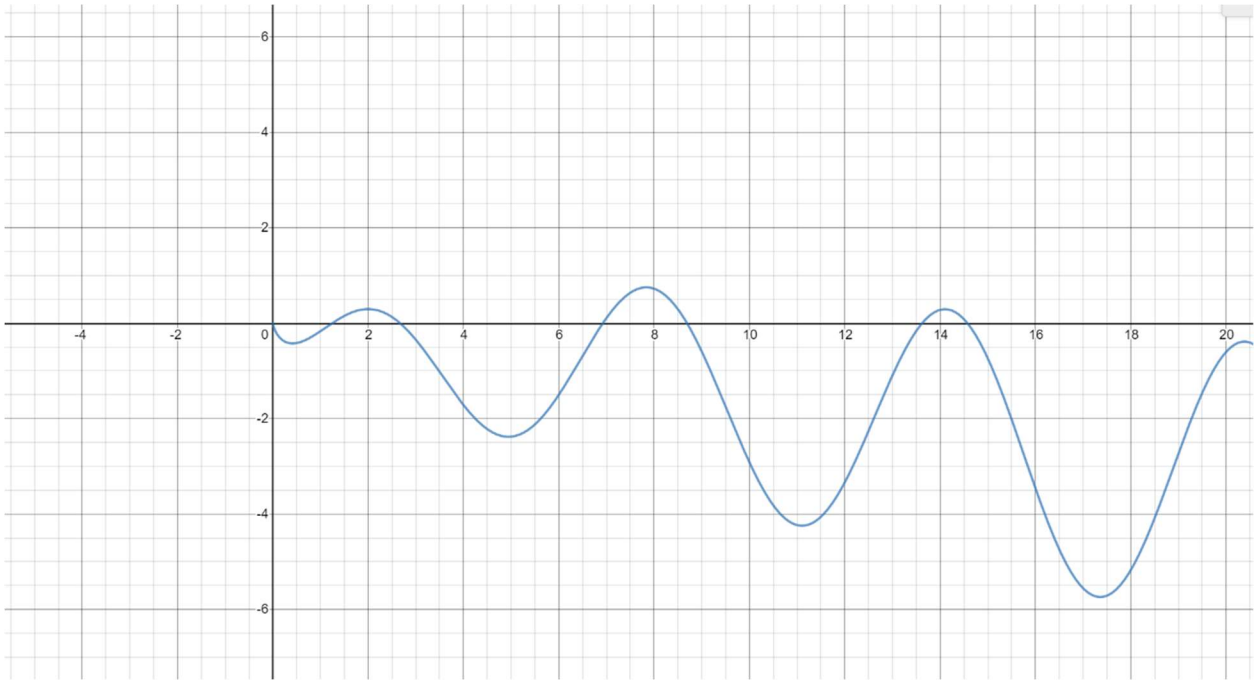
Toliau gauti intervalai yra pateikiami tikslinimo algoritmams:

Metodas	Artinys/Intervalas	Šaknis	Tikslumas	Iteracijos
Stygų	$(1.2, 1.3)$	1.2448372724904877	0.0000001	3
Niutono	1.2	1.2448371935430196	0.0000001	3
Skenavimo	$(1.2, 1.3)$	1.2448371887207033	0.0000001	21
Stygų	$(2.6, 2.7)$	2.671334368237046	0.0000001	4
Niutono	2.6	2.6713343778709864	0.0000001	4
Skenavimo	$(2.6, 2.7)$	2.6713342666625985	0.0000001	27
Stygų	$(6.9, 6.9)$	6.922074727277498	0.0000001	3
Niutono	6.9	6.922074693253911	0.0000001	3
Skenavimo	$(6.9, 6.9)$	6.922074699401848	0.0000001	20
Stygų	$(8.6, 8.7)$	8.690729387604913	0.0000001	3
Niutono	8.6	8.690729396068326	0.0000001	4
Skenavimo	$(8.6, 8.7)$	8.690729331970196	0.0000001	22
Stygų	$(13.6, 13.7)$	13.619990725101998	0.0000001	4
Niutono	13.6	13.619990715641462	0.0000001	4
Skenavimo	$(13.6, 13.7)$	13.619990539550752	0.0000001	22
Stygų	$(14.5, 14.6)$	14.572906663898022	0.0000001	4
Niutono	14.5	14.572906704760806	0.0000001	4
Skenavimo	$(14.5, 14.6)$	14.572906684875454	0.0000001	21



pav. 8 MatplotLib gautas g(x) grafikas

Žali kvadratai yra gauti šaknų intervalai naudojantis skenavimo su nekintančiu žingsniu.



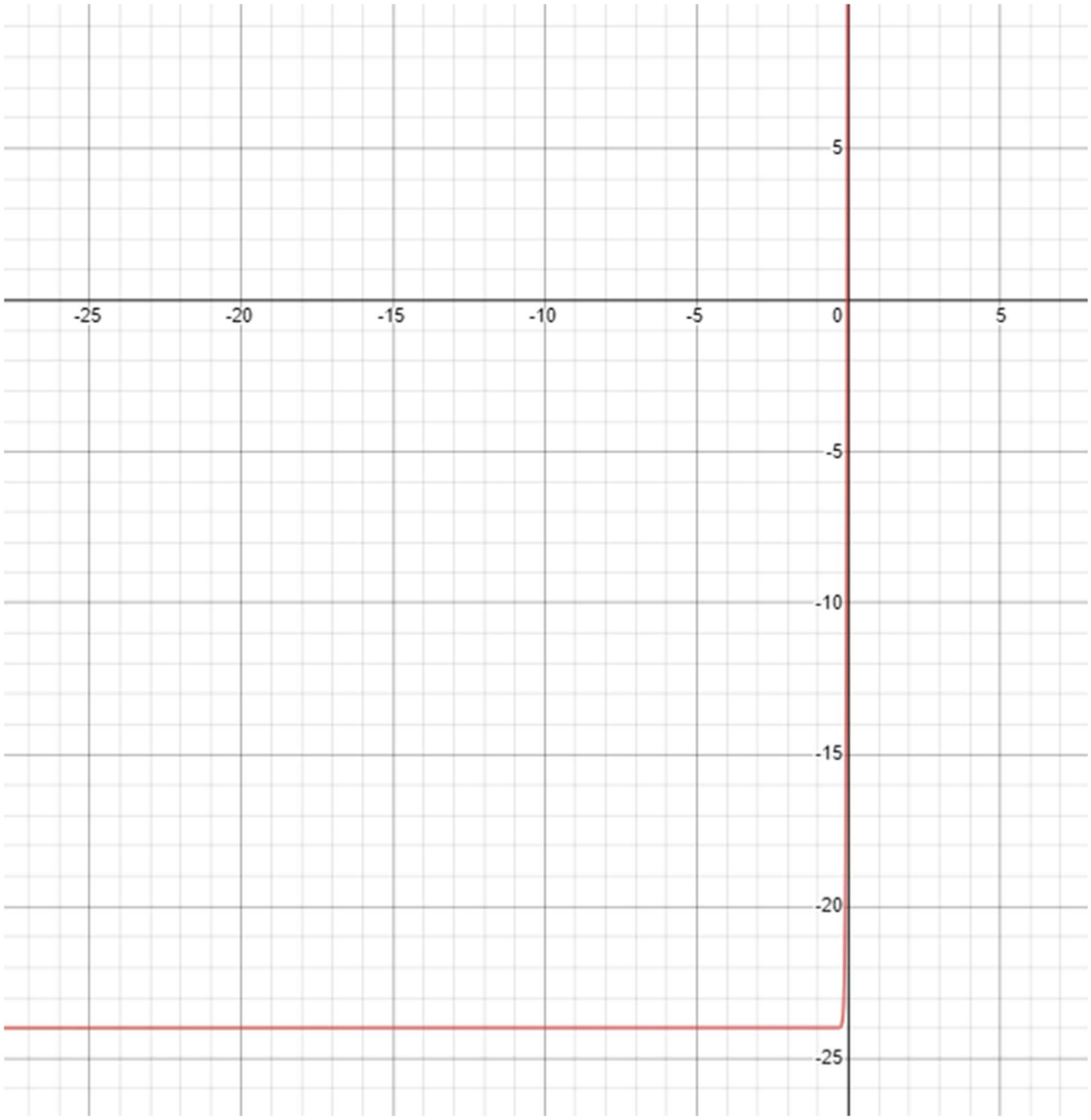
pav. 9 desmos.com gautas grafikas g(x)

Niutono ir stygų metodai yra beveik identiško greičio šiuo atveju, skenavimo metodas gali būti greitesnis optimizuojant žingsnio mažėjimą.

3. Antra užduotis

Sprendžiant antrąją užduotį ranka apsiskaičiavau funkcijos išraišką, kuri gavosi:

$$80e^{30k} - 24$$



pav. 10 desmos.com gautas grafikas

Iš grafiko matosi, kad sprendinys yra tik vienas, todėl kode aš skaičiavau pirmą intervalą (kad vienas galima rasti apsiskaičiavus ranka).

Toliau naudodamasis pirmoje užduotyje naudotu stygų su nekintančiu dydžiu aptikau sprendimo intervalą. O patikslinimui naudojausi Niutono metodą, nes jis yra panašaus greičio kaip ir stygų metodas.

Gautas sprendinys: -0.040132426810864534 po 10 iteracijos, skaičiavimas atliktas su 1e-8 tikslumu, realus sprendinys gautas išsprendus ranka gaunasi $k = \frac{\ln\left(\frac{24}{80}\right)}{30} \approx -0.04013$