Kauno technologijos universitetas

Informatikos fakultetas

Netiesinių lygčių sprendimo ataskaita

Laboratorinio užduotis

Aistis Jakutonis

Studentas

Čalnerytė Dalia
Kriščiūnas Andrius

Dėstytojai

Kaunas, 2025

Turinys

1. Pirma dalis

Užduoties variantas:

Metodai: Stygų, Niutono (liestinių).

Funkcija f(x):

$$f(x) = -1.29x^4 + 5.08x^3 - 2.76x^2 - 6.31x + 4.1$$

1 funkcija f(x)

Funkcija g(x):

$$g(x) = x(\cos(x))^2 - \left(\frac{x}{2}\right)^2 ;\ -10 \leq x \leq 10$$

2 funkcija g(x)

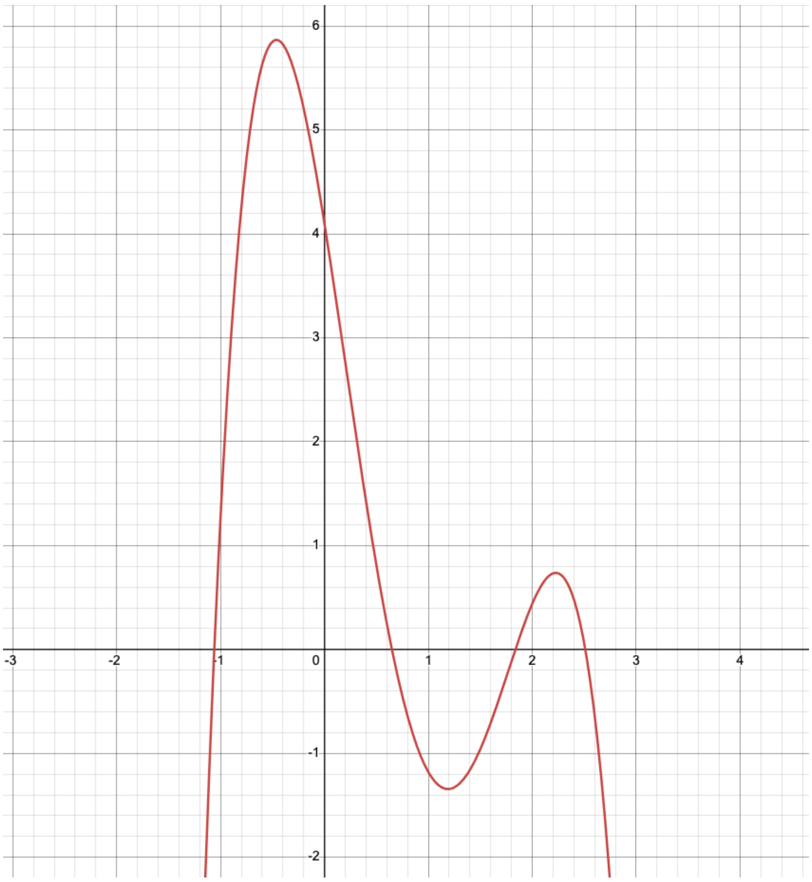
Funkcija h(x):

$$h(x) = 29\sin(x) - 5\cos(4x)\,; 0 \leq x \leq 4$$

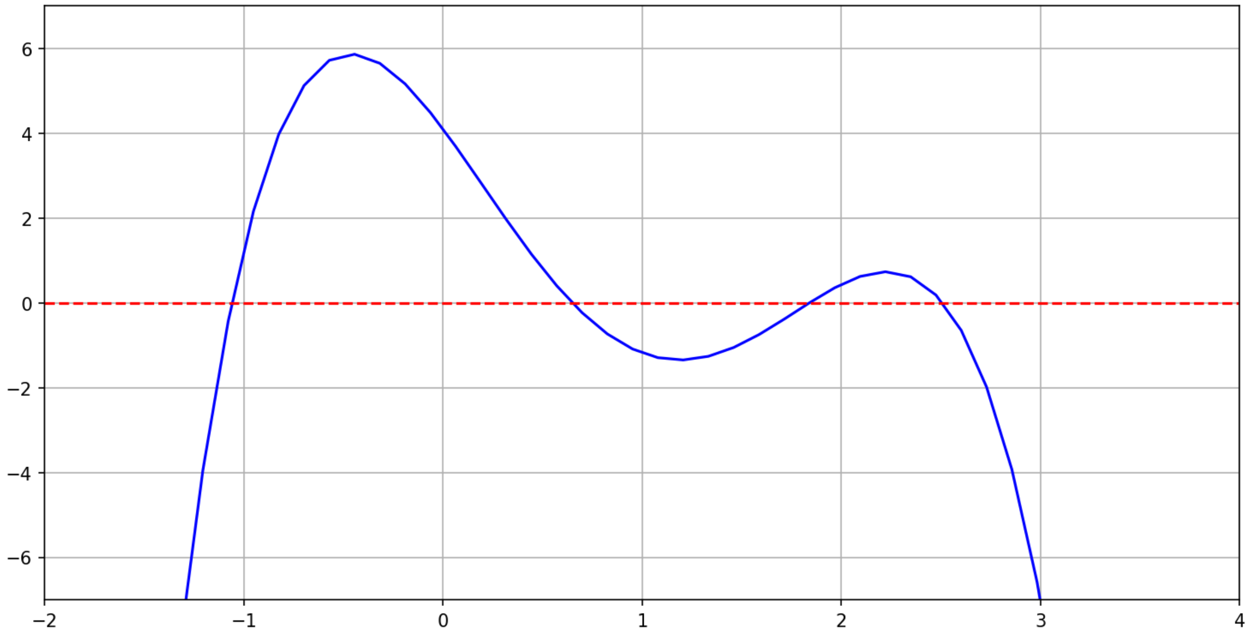
3 funkcija h(x)

1.1. Funkcijos f(x) šakų intervalas taikant grubų ir tikslesnį įverčius

Funkcijos f(x) grafikas:

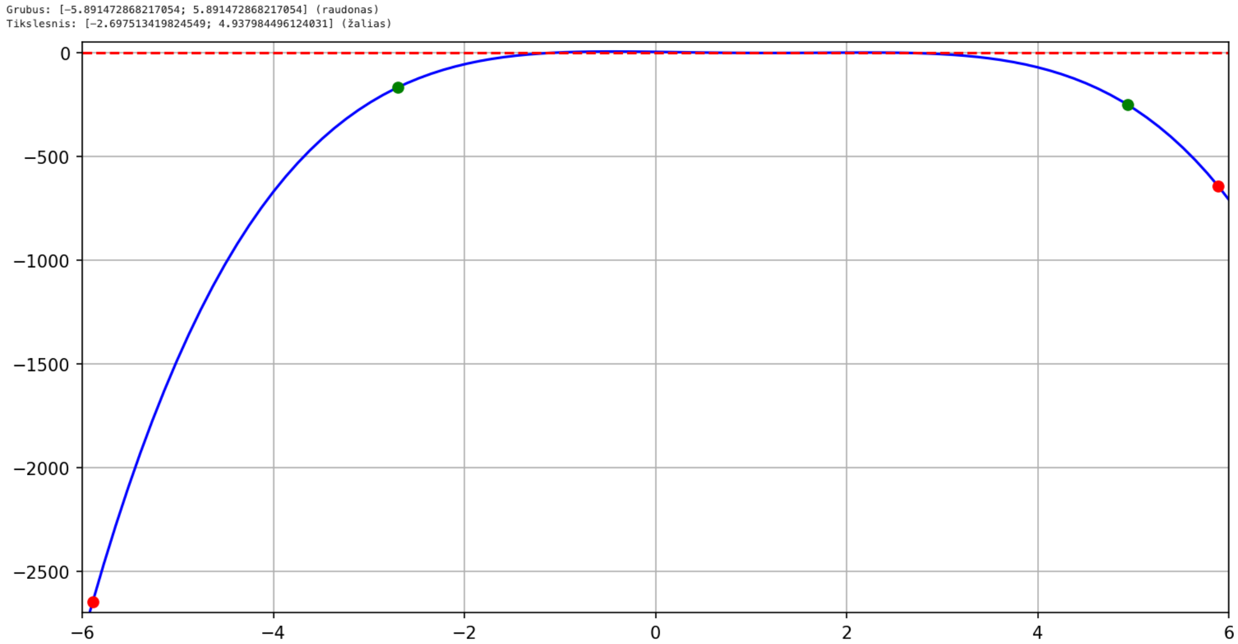


1 pav. Pilnas funkcijos f(x) grafikas



2 pav. Funkcijos f(x) grafikas atvaizduotas programine įranga

Apskaičiavus grubų (raudonas) ir tikslesnį (žalias) įverčius:



3 pav. Grafikas atvaizduotas taip, kad matytųsi ir grubus ir tikslesnis įverčiai

Akivaizdžiai matyti, kad tikslesnis įvertis parodo žymiai geresnį šaknų intervalą nei grubus įvertis. Todėl ateinančiose užduotyse naudosime tikslesnio įverčio rezultatus.

Grubus įvertis buvo apskaičiuotas pagal:

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

4 funkcija Daugianaris

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

5 funkcija Grubaus įverčio apskaičiavimo formulė

Tikslesnis įvertis buvo apskaičiuotas pagal:

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

6 funkcija Daugianaris

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

7 funkcija Tikslesnio įverčio apskaičiavimo formulė teigiamoms šaknims

Neigiamų šaknų įverčiai gaunami, vietoje daugianario f(x) imant f(-x).

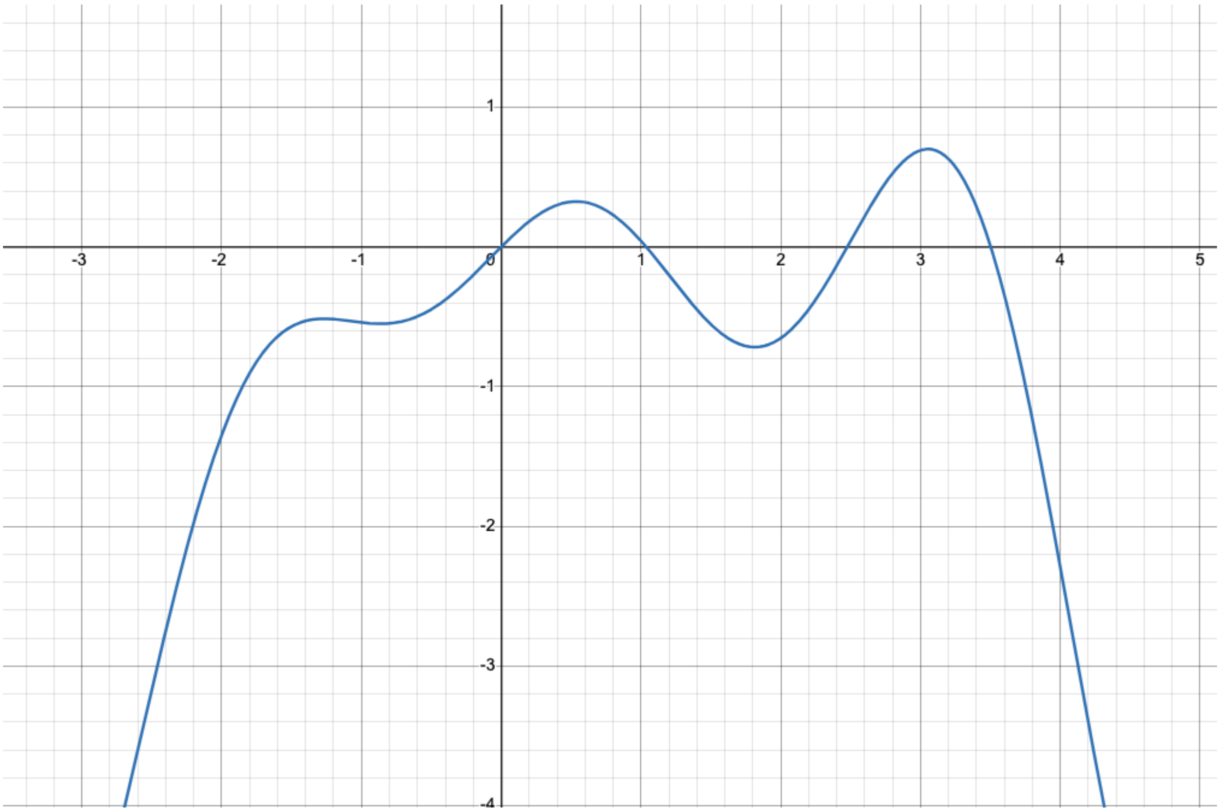
Algoritmo kodo fragmentas:

1 lentelė Grubaus ir tikslesnio įverčių kodo fragmentas

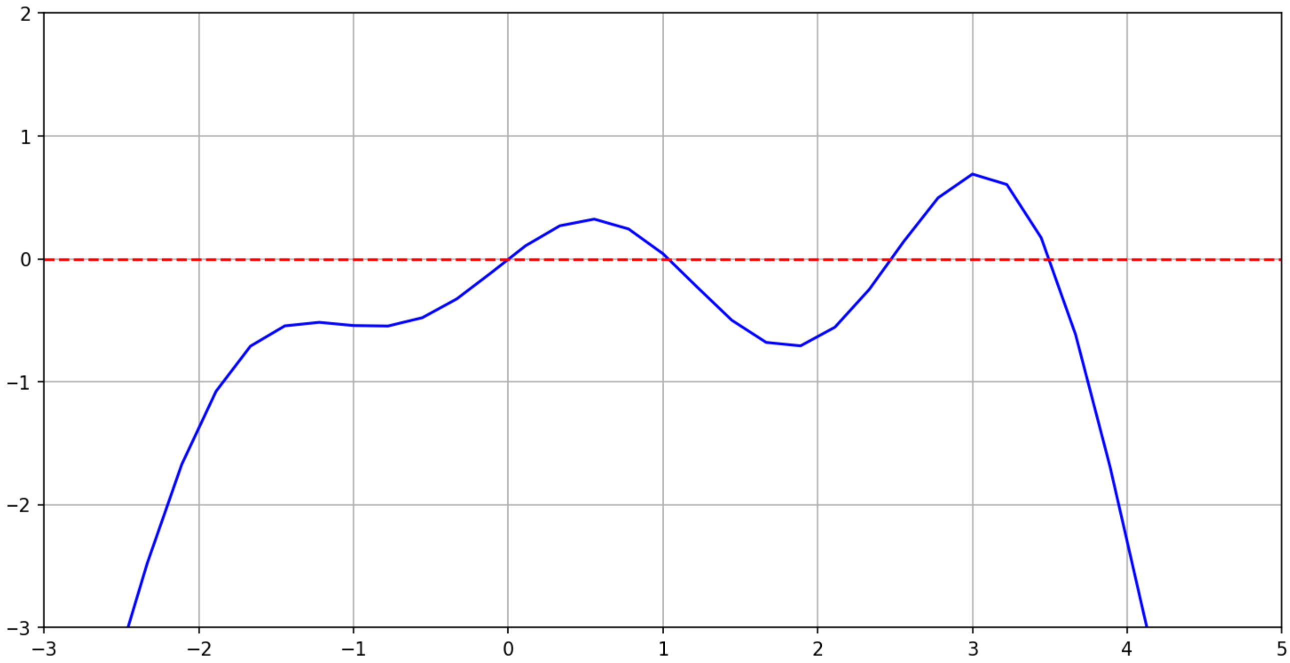
```
def func(x):
    return -1.29 * x**4 + 5.08 * x**3 - 2.76 * x**2 - 6.31 * x + 4.10

grubusRteig = 1 + 6.31 / 1.29
grubusRneig = -1 * grubusRteig

Bteig = 5.08
kteig = 4 - 3
tikslesnisRteig = 1 + (Bteig / 1.29)**(1/kteig)

Bneig = 6.31
kneig = 4 - 1
tikslesnisRneig = -1 * (1 + (Bneig / 1.29)**(1/kneig))
```

1.2. Funkcijos g(x) atvaizdavimas nurodytame intervale

Funkcijos g(x) grafikas:

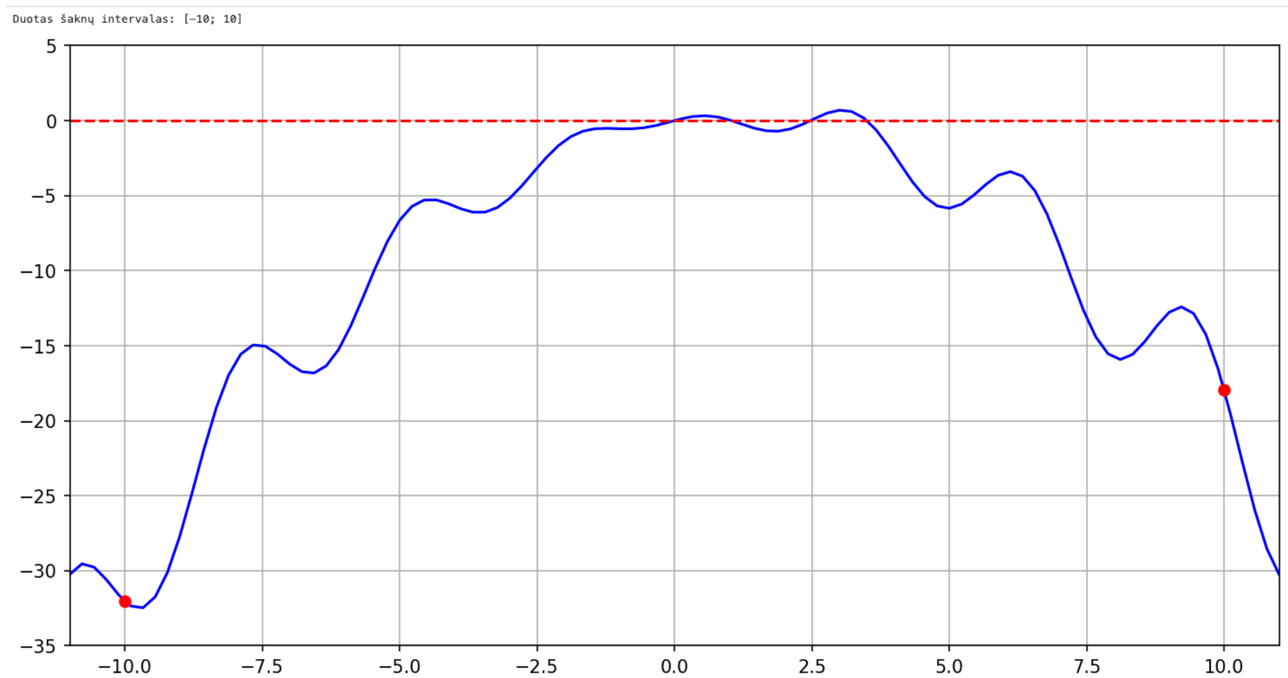


4 pav. Pilnas funkcijos g(x) grafikas



5 pav. Funkcija g(x) atvaizduota programine įranga

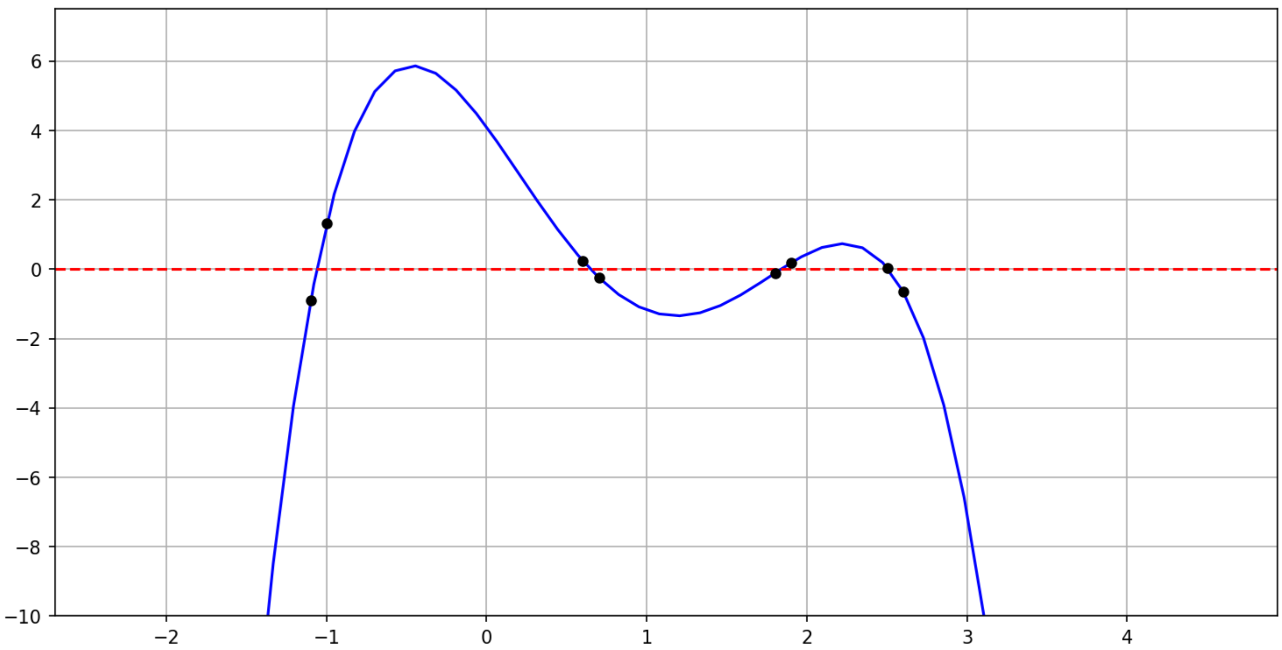
Funkcija g(x) su intervalų jos žymėmis:



6 pav. Funkcija g(x) su intervalų žymėmis

1.3. Funkcijos f(x) šaknų atskyrimo intervalų radimas naudojant skenavimo algoritmą

Šaknų intervalai atlikus skenavimo algoritmą tarp pirmoje užduotyje gauto tikslesnio įverčio reikšmių (intervalai grafike pažymėti tarp juodų taškų):



7 pav. Funkcijos f(x) šaknų intervalai atlikus skenavimo algoritmą

```
Intervalas: [-2.697513419824549; 4.937984496124031]

Rasti šaknų intervalai:
1) [-1.097513; -0.997513]
2) [0.602487; 0.702487]
3) [1.802487; 1.902487]
4) [2.502487; 2.602487]
```

8 pav. Gauti programinės įrangos rezultatai atlikus skenavimo algoritmą

Taigi, matome, jog atlikus skenavimo algoritmą buvo surastos keturios šaknys, kurių intervalus galime matyti 8 paveikslėlyje.

Skenavimas buvo atliekamas taip: Buvo pasirinktas tam tikras žingsnis h, kurį naudojant buvo einama per funkcijos taškus. Tikrinant ar pirmesnio žingsnio funkcijos rezultatas skiriasi ženklu nuo po jo einančio žingsnio, jei ne einama toliau.

Algoritmo kodo fragmentas:

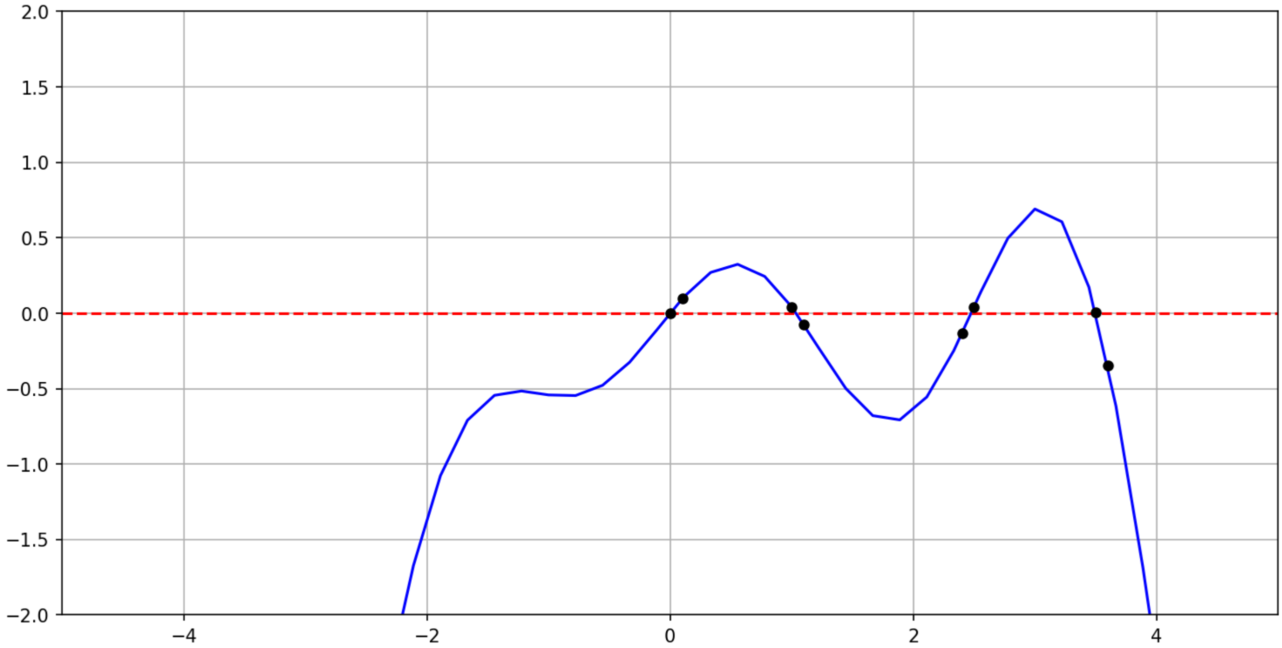
2 lentelė Šaknų atskyrimo intervalo radimo algoritmo kodo fragmentas

```
#skenavimo zingsnis
#saknu intervalu sarasas
h = 0.1
intervalai = []
iteracijos = []
steps = 0

xleft = start
```

```
yleft = func(xleft)

while xleft < end - 1e-12:
    #desine tikrinamo intervalo dalis (pridejus zingsni)
    xright = min(xleft + h, end)
    yright = func(xright)
    steps += 1

    #zenklo tikrinimas
    if yleft * yright < 0:
        #jei skiriasi zenklai tada rastas saknies intervalas
        intervalai.append((xleft, xright))
        iteracijos.append(steps)
        plt.plot([xleft, xright], [func(xleft), func(xright)], 'ko',
markersize=5)
        steps = 0

    #judame toliau ir ieskome kitu saknu
    xleft, yleft = xright, yright
```

1.4. Funkcijos g(x) šaknų atskyrimo intervalų radimas naudojant skenavimo algoritmą

Šakų intervalai atlikus skenavimo algoritmą nurodytame intervale (šaknų intervalai pažymėti tarp juodų taškų):



9 pav. Funkcijos g(x) šaknų intervalai atlikus skenavimo algoritmą

```
Intervalas: [-10; 10]

Rasti šaknų intervalai:
1) [-0.000000; 0.100000]
2) [1.000000; 1.100000]
3) [2.400000; 2.500000]
4) [3.500000; 3.600000]
```

10 pav. Programinės įrangos rezultatai atlikus skenavimo algoritmą

Matome, jog funkcijoje g(x) taip pat buvo rasti keturių šaknų intervalai atlikus skenavimo algoritmą.

1.5. Funkcijos f(x) šaknų tikslinimas naudojant stygų ir Niutono (liestinių) metodus

Stygų metodas: paimami intervalo galų taškai ant x ašies ir tada randami funkcijos taškai įsistatant x į funkcijos formulę. Tuomet yra brėžiama tiesė tarp rastų taškų ir ten, kur ši tiesė kerta x ašį yra gaunamas naujas taškas. Prieš pasirenkant, kurią intervalo pusę perkelti į naujai rastą tašką vėl yra patikrinama tarp kurių funkcijos taškų skiriasi ženklai.

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

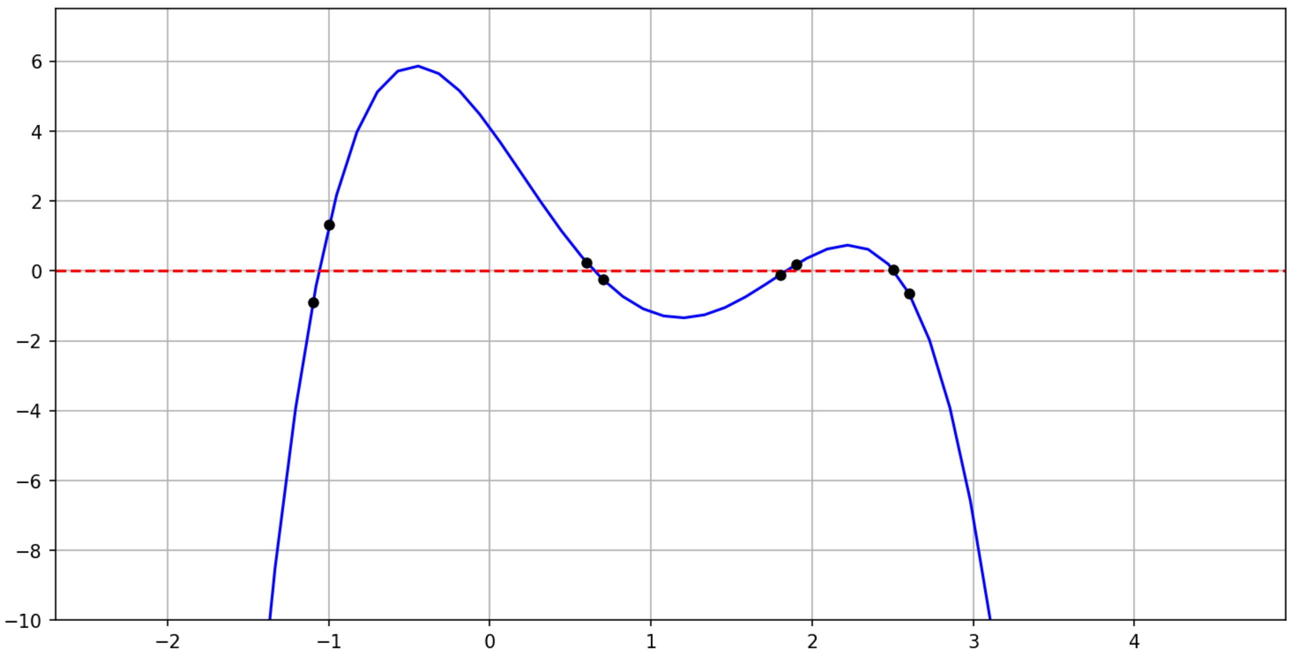
8 funkcija X ašies kirtimo apskaičiavimo formulė

Niutono (liestinių) metodas: pasirenkamas taškas esanti šaknies intervale. Tada apskaičiuojama funkcijos reikšmė tame taške. Tuomet yra randama liestinė rastam funkcijos taškui. Ir ten, kur liestinė kerta x ašį yra gaunamas naujas taškas nuo, kurio vėl bus kartojami šie veiksmai kol bus rastas šaknies taškas reikiamu tikslumu.

$$f(x^0) + \frac{df}{dx}|_{x0}\,(x - x^0) = 0$$

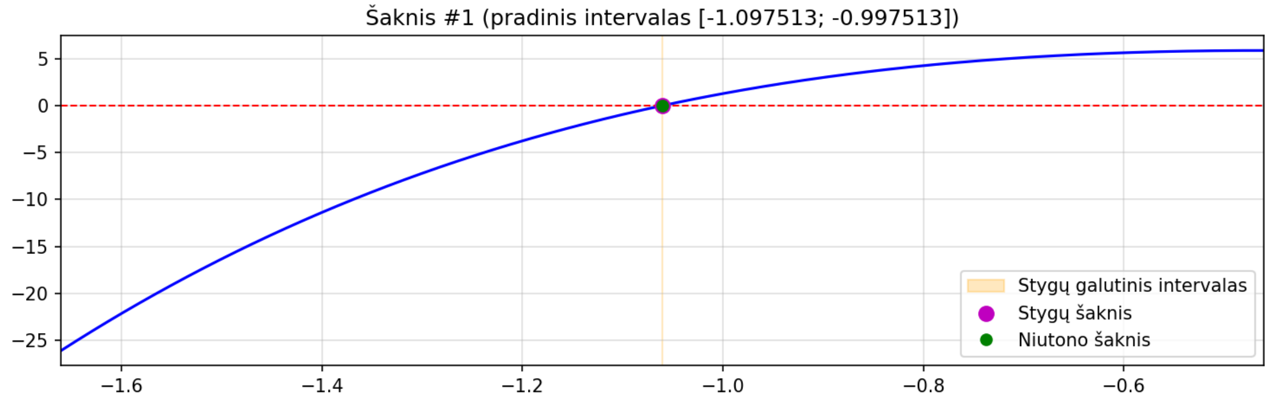
9 funkcija Naujo x radimo formulė

Funkcijos f(x) šaknų tikslinimą pradėjome pasižymėdami rastus šaknų intervalus:

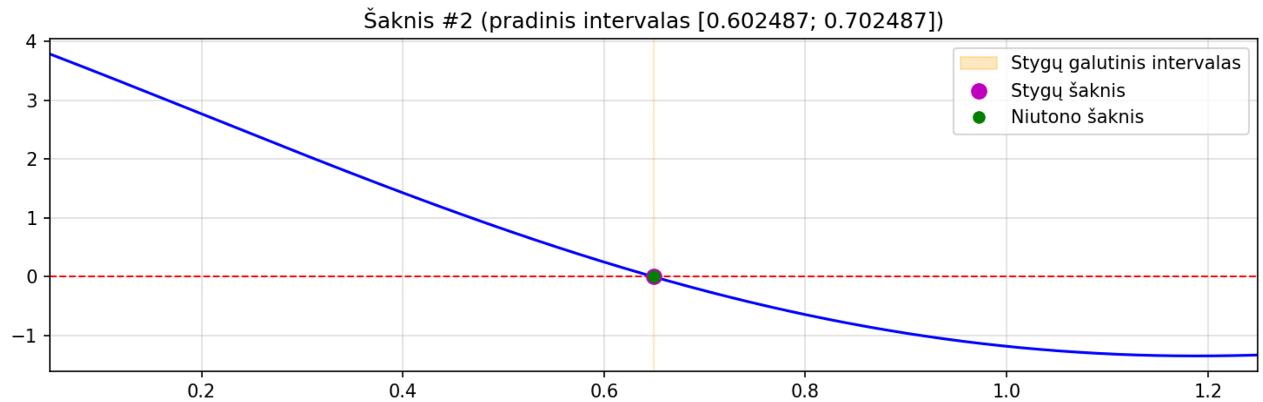


11 pav. Funkcijos f(x) šaknų intervalai

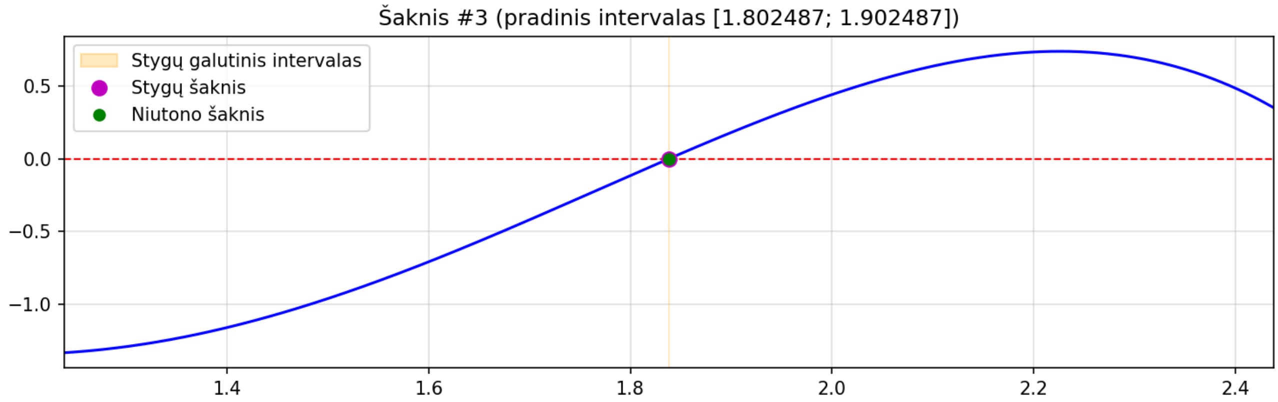
Pasinaudojus stygų ir Niutono metodais buvo rastos šaknys, kurias pademonstravome grafikais:



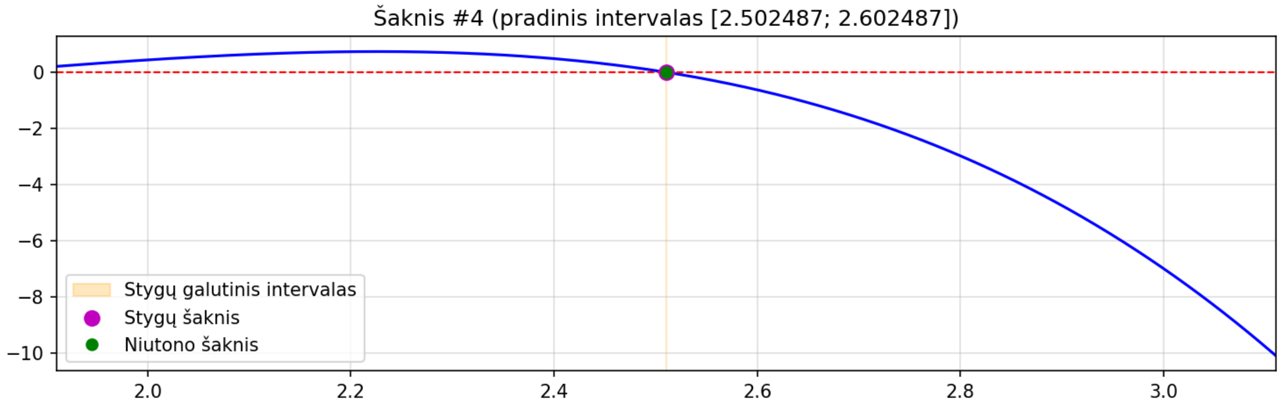
12 pav. Pirmos šaknies tikslinimas stygų ir Niutono metodais



13 pav. Antros šaknies tikslinimas stygų ir Niutono metodais



14 pav. Trečios šaknies tikslinimas stygų ir Niutono metodais



15 pav. Ketvirtos šaknies tikslinimas stygų ir Niutono metodais

Toliau galime matyti programinės įrangos išvestus rezultatus apie rastas šaknų vertes, iteracijų skaičių bei nustatytą tikslumą:

```
Intervalas: [-2.697513419824549; 4.937984496124031]

1) [-1.097513; -0.997513]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   -1.0603727932711937
Niutono rez: -1.0603727933017841
Stygų funkcijos reikšmė ties šaknimi:   4.525881891481731e-10
Niutono funkcijos reikšmė ties šaknimi: -2.4582380575566276e-10
Tikslumas: 1e-09
Iteracijų skaičius stygų:   7
Iteracijų skaičius niutono: 3

2) [0.602487; 0.702487]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   0.6494305349810486
Niutono rez: 0.6494305348708012
Stygų funkcijos reikšmė ties šaknimi:   -5.237676958813609e-10
Niutono funkcijos reikšmė ties šaknimi: 1.4303225270850817e-11
Tikslumas: 1e-09
Iteracijų skaičius stygų:   6
Iteracijų skaičius niutono: 6

3) [1.802487; 1.902487]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   1.8383683864541727
Niutono rez: 1.8383683864491867
Stygų funkcijos reikšmė ties šaknimi:   1.49018575257287e-11
Niutono funkcijos reikšmė ties šaknimi: 0.0
Tikslumas: 1e-09
Iteracijų skaičius stygų:   5
Iteracijų skaičius niutono: 9

4) [2.502487; 2.602487]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   2.510558368029964
Niutono rez: 2.5105583680921373
Stygų funkcijos reikšmė ties šaknimi:   3.582609764407607e-10
Niutono funkcijos reikšmė ties šaknimi: -4.085620730620576e-14
Tikslumas: 1e-09
Iteracijų skaičius stygų:   11
Iteracijų skaičius niutono: 12
```

16 pav. Programinės įrangos rezultatai

3 lentelė Funkcijos f(x) šaknų radimo rezultatai pritaikius stygos ir Niutono metodus

Metodas	Atskyrimo intervalas	Rezultatas	Funkcijos reikšmė	Tikslumas	Iteracijų skaičius
Stygų	-1,097513; -0,997513	-1,06037	4,52588189e-10	1e-9	7

Niutono (liestinių)	-1,097513; -0,997513	-1,06037	-2,45823805e-10	1e-9	3
Stygų	0,602487; 0,702487	0,649431	-5,23767695e-10	1e-9	6
Niutono (liestinių)	0,602487; 0,702487	0,649431	1,430322527e-11	1e-9	6
Stygų	1,802487; 1,902487	1,83837	1,490185752e-11	1e-9	5
Niutono (liestinių)	1,802487; 1,902487	1,83837	0	1e-9	9
Stygų	2,502487; 2,602487	2,51056	3,582609764e-10	1e-9	11
Niutono (liestinių)	2,502487; 2,602487	2,51056	-4,085620730e-14	1e-9	12

Algoritmo kodo fragmentas:

4 lentelė Šaknų tikslinimas naudojant stygų metodą

```
#stygu metodas
        fa, fb = func(a), func(b)
        #jei saknis yra intervalo pradzia arba pabaiga
        if fa == 0:
            rez1 = a
            steps1 = 0
        elif fb == 0:
            rez1 = b
            steps1 = 0
        else:
            steps1 = 0

            #pirminis plotis/tikslumas
            prev_width = abs(b - a)
            #blogejimo parametrai
            no_improve = 0
            max_stall = 5
            epsw = 1e-15

            #kol nepasiektas reikiamas tikslumas
            while abs(b - a) > accuracy:
                #c tai taskas kur styga kerta x asi
                c = a - fa * (b - a) / (fb - fa)
                fc = func(c)
                steps1 += 1

                if abs(fc) <= accuracy:
                    a = b = c
                    break

                #tikrina tarp kuriu tasku yra saknis
                if fa * fc < 0:
                    b, fb = c, fc
                else:
                    a, fa = c, fc
```

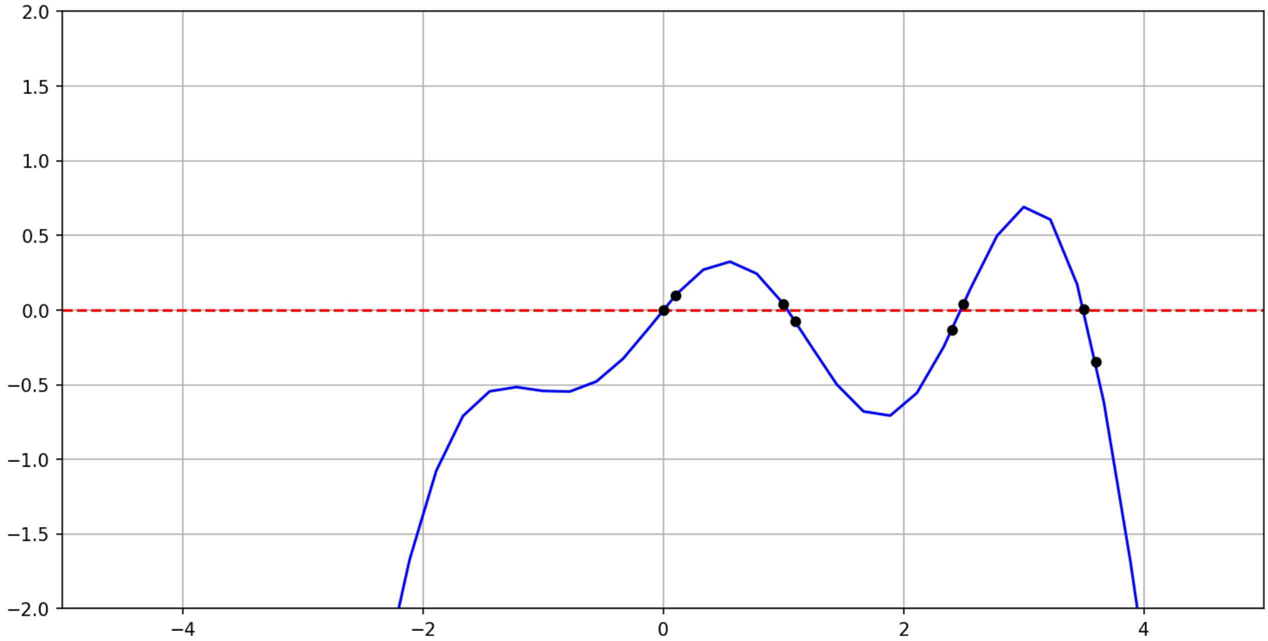
```
                width = abs(b - a)

                #tikrina ar neblogeja
                if width >= prev_width - epsw:
                    no_improve += 1
                else:
                    no_improve = 0

                prev_width = width

                #kai uzstringam arba negerejam - neprogresuojam (plotis
nemazeja arba mazeja labai letai)
                #kai toks dalykas trunka max_stall zingsniu skaiciu
                #tada paimam intervalo viduri ir skatiname konvergacija
                if no_improve >= max_stall:
                    m = 0.5 * (a + b)
                    fm = func(m)
                    steps1 += 1

                    if fa * fm < 0:
                        b, fb = m, fm
                    else:
                        a, fa = m, fm

                    no_improve = 0

                #apsisaugojimas nuo dalybos is nulio
                if fb == fa:
                    break

            # rez1 = a if abs(func(a)) < abs(func(b)) else b
            rez1 = (a + b) / 2
```

5 lentelė Šaknų tikslinimas naudojant Niutono (liestinių) metodą

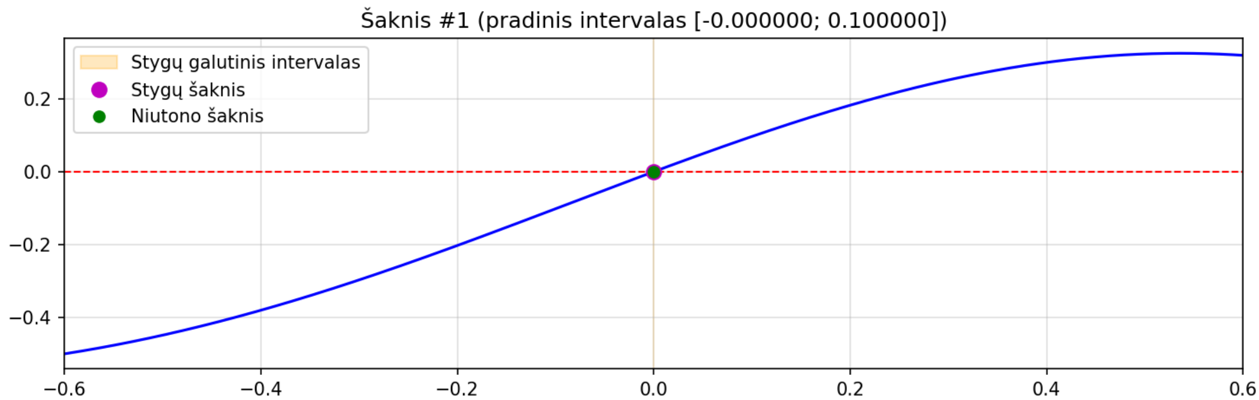
```
#pirmasis spejimo taskas
        #maximalus iteraciju skaicius
        rez2 = a1
        max_iter = 1000

        #niutono liestiniu metodas
        while steps2 < max_iter:
            #funkcijos reiksme ties spejimo tasku
            fx = func(rez2)

            #jei spejimas yra pakankamai tikslus radome rezultata
            if abs(fx) <= accuracy:
                break

            #funkcijos isvestine - niutono formulei reikalinga
            dfx = -5.16 * rez2**3 + 15.24 * rez2**2 - 5.52 * rez2 - 6.31

            #negalim testi nes bus dalyba is nulio
            if dfx == 0:
                break

            #gaunamas naujas spejimo taskas ten kur liestine kirto x asi
```

```
                xnew = rez2 - fx / dfx

                #apsauga divergacijos atveju
                #jei spejimas iseina is intervalo ribu ar nauja reiksme padidino
klaida
                #vietoje gauto spejimo tasko imamas intervalo vidurio taskas
                if (xnew < a1 or xnew > b1) or (abs(func(xnew)) > abs(fx)):
                    xnew = 0.5 * (a1 + b1)

                #atnaujinamas spejimas ir iteraciju skaicius
                rez2 = xnew
                steps2 += 1
```

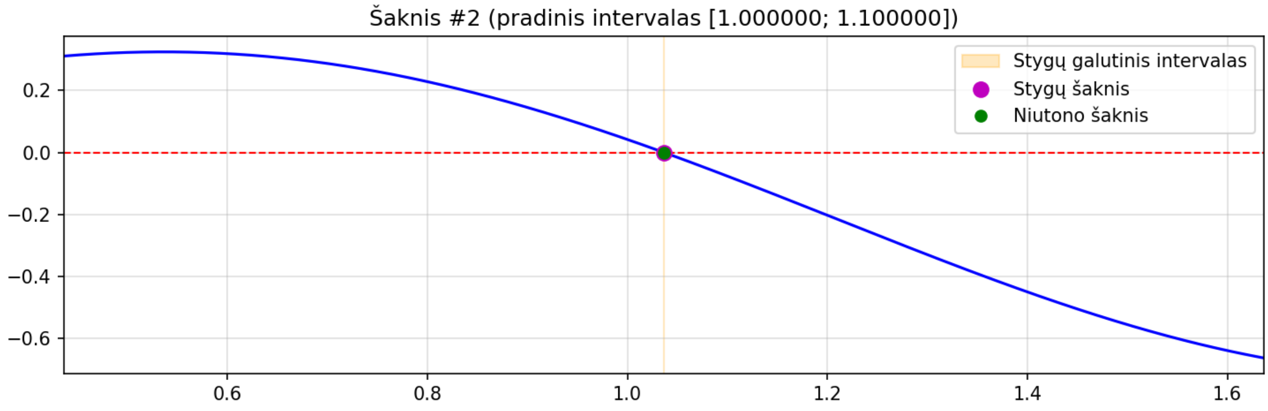
1.6. Funkcijos g(x) šaknų tikslinimas naudojant stygų ir Niutono (liestinių) metodus



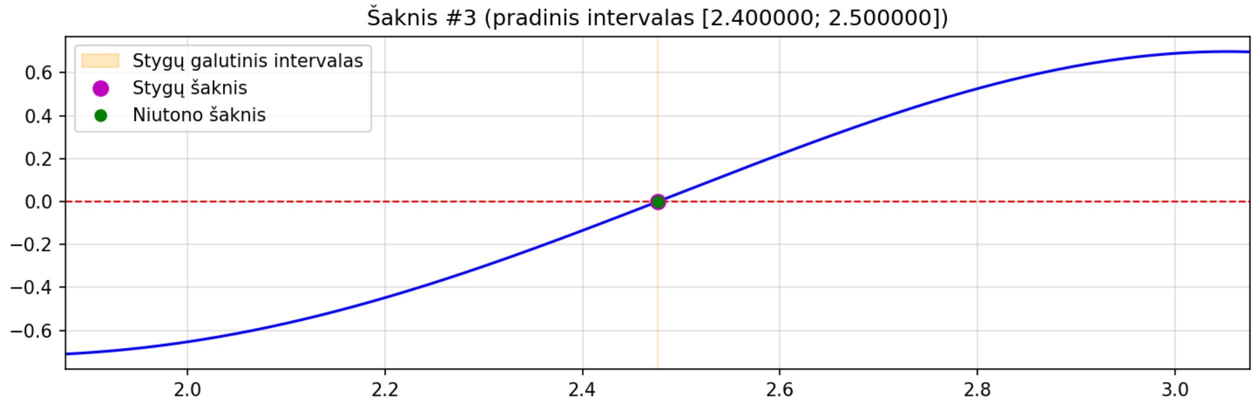
17 pav. Funkcijos g(x) šaknų intervalai



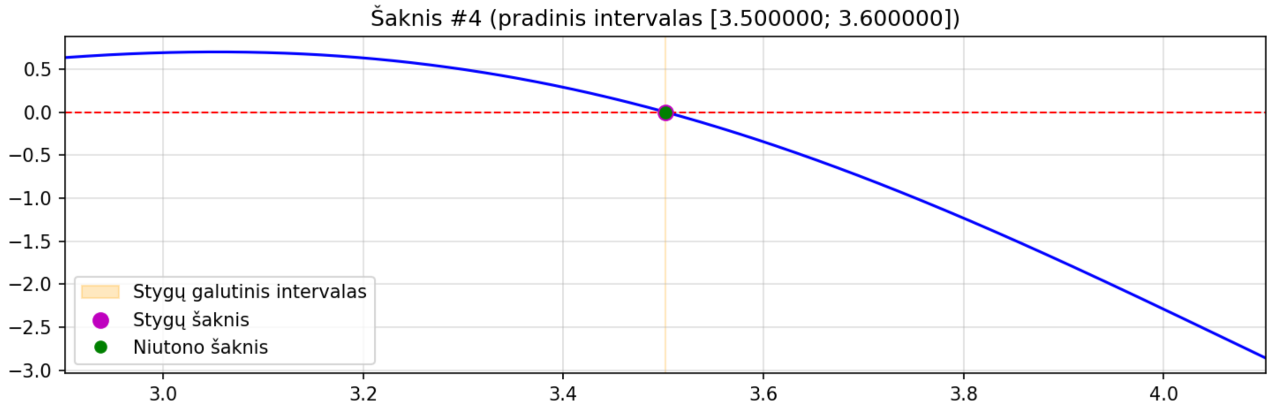
18 pav. Pirmos šaknies tikslinimas stygų ir Niutono metodais



19 pav. Antros šaknies tikslinimas stygų ir Niutono metodais



20 pav. Trečios šaknies tikslinimas stygų ir Niutono metodais



21 pav. Ketvirtos šaknies tikslinimas stygų ir Niutono metodais

```
Intervalas: [-10; 10]

1) [-0.000000; 0.100000]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   6.808499307439747e-16
Niutono rez: -1.8790524691780774e-14
Stygų funkcijos reikšmė ties šaknimi:   6.808499307439746e-16
Niutono funkcijos reikšmė ties šaknimi: -1.8790524691780863e-14
Tikslumas: 1e-09
Iteracijų skaičius stygų:   1
Iteracijų skaičius niutono: 0

2) [1.000000; 1.100000]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   1.0366738759885363
Niutono rez: 1.0366738760140357
Stygų funkcijos reikšmė ties šaknimi:   2.968175705220233e-11
Niutono funkcijos reikšmė ties šaknimi: -9.303668946358812e-14
Tikslumas: 1e-09
Iteracijų skaičius stygų:   6
Iteracijų skaičius niutono: 3

3) [2.400000; 2.500000]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   2.47646804700038
Niutono rez: 2.4764680473081127
Stygų funkcijos reikšmė ties šaknimi:   -5.496223476342266e-10
Niutono funkcijos reikšmė ties šaknimi: 2.6645352591003757e-15
Tikslumas: 1e-09
Iteracijų skaičius stygų:   3
Iteracijų skaičius niutono: 6

4) [3.500000; 3.600000]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   3.502147391054282
Niutono rez: 3.50214739124302
Stygų funkcijos reikšmė ties šaknimi:   5.076952191984674e-10
Niutono funkcijos reikšmė ties šaknimi: -9.39484046114103e-11
Tikslumas: 1e-09
Iteracijų skaičius stygų:   7
Iteracijų skaičius niutono: 8
```

22 pav. Programinės įrangos rezultatai

6 lentelė Funkcijos g(x) šaknų radimo rezultatai pritaikius stygos ir Niutono metodus

Metodas	Atskyrimo intervalas	Rezultatas	Funkcijos reikšmė	Tikslumas	Iteracijų skaičius

Stygų	-0; -0,1	6,8085e-16	6,8084993e-16	1e-9	1
Niutono (liestinių)	-0; -0,1	-1,8791e-14	-1,8790524e-14	1e-9	0
Stygų	1; 1,1	1,036674	2,96817570e-11	1e-9	6
Niutono (liestinių)	1; 1,1	1,036674	-9,3036689e-14	1e-9	3
Stygų	2,4; 2,5	2,476468	-5,4962234e-10	1e-9	3
Niutono (liestinių)	2,4; 2,5	2,476468	2,66453525e-15	1e-9	6
Stygų	3,5; 3,6	3,502147	5,07695219e-10	1e-9	7
Niutono (liestinių)	3,5; 3,6	3,502147	-9,3948404e-11	1e-9	8

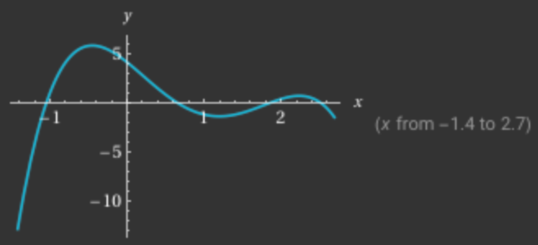
1.7. Gautų funkcijos f(x) reikšmių tikrinimas naudojant išorinius išteklius

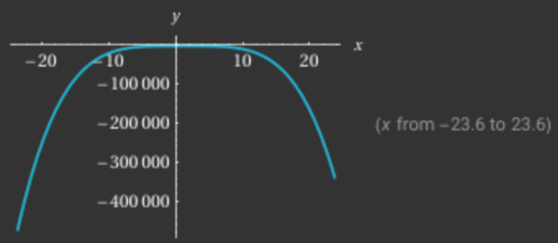
Gautas šaknis tikriname su programinėje įrangoje įdiegtomis funkcijomis. Gauti rezultatai yra pavaizduoti žemiau esančiame paveikslėlyje:

Input

$-1.29x^4 + 5.08x^3 - 2.76x^2 - 6.31x + 4.1$

Plot





Alternate form

$-1.29(x - 2.51056)(x - 1.83837)(x - 0.649431)(x + 1.06037)$

$x(x(x(5.08 - 1.29x) - 2.76) - 6.31) + 4.1$

$\frac{1}{100}(-129x^4 + 508x^3 - 276x^2 - 631x + 410.)$

Roots

Step-by-step solution

$x \approx -1.06037$

$x \approx 0.649431$

$x \approx 1.83837$

$x \approx 2.51056$

23 pav. WolframAlpha įrankio rezultatai

```
1) [-1.097513; -0.997513]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   -1.0603727932711937
Niutono rez: -1.0603727933017841

2) [0.602487; 0.702487]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   0.6494305349810486
Niutono rez: 0.6494305348708012

3) [1.802487; 1.902487]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   1.8383683864541727
Niutono rez: 1.8383683864491867

4) [2.502487; 2.602487]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   2.510558368029964
Niutono rez: 2.5105583680921373

[-1.06037279  2.51055837  1.83836839  0.64943053]
```

24 pav. Šaknų reikšmių lyginimo rezultatai

Rezultatuose matyti, kad gautų šaknų rezultatai beveik visiškai sutampa su rezultatais gautais naudojant stygų ar Niutono (liestinių) metodus. Taigi, matant šiuos rezultatus galime teigti, kad gauti rezultatai buvo gana tikslūs ir teisingi.

1.8. Gautų funkcijos g(x) reikšmių tikrinimas naudojant išorinius išteklius

Gautas šaknis tikriname su programinėje įrangoje įdiegtomis funkcijomis. Gauti rezultatai yra pavaizduoti žemiau esančiame paveikslėlyje:

Numerical roots — More digits

$x \approx 1.03667387601396\ldots$

$x \approx 2.47646804730811\ldots$

$x \approx 3.50214739121355\ldots$

Integer root

$x = 0$

25 pav. WolframAlpha įrankio rezultatai

```
1) [-0.000000; 0.100000]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   6.808499307439747e-16
Niutono rez: -1.8790524691780774e-14
Root: -1.8790524691780774e-14

2) [1.000000; 1.100000]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   1.0366738759885363
Niutono rez: 1.0366738760140357
Root: 1.0366738760139507

3) [2.400000; 2.500000]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   2.47646804700038
Niutono rez: 2.4764680473081127
Root: 2.476468047309617

4) [3.500000; 3.600000]
Šaknų tikslinimas stygų ir niutono (liestinių) metodu:
Stygų rez:   3.502147391054282
Niutono rez: 3.50214739124302
Root: 3.5021473911905634
```

26 pav. Šaknų reikšmių lyginimo rezultatai

Pagal gautus rezultatus matome, jog apskaičiuotos funkcijos g(x) šaknys buvo pakankamai tikslios.

2. Antra dalis

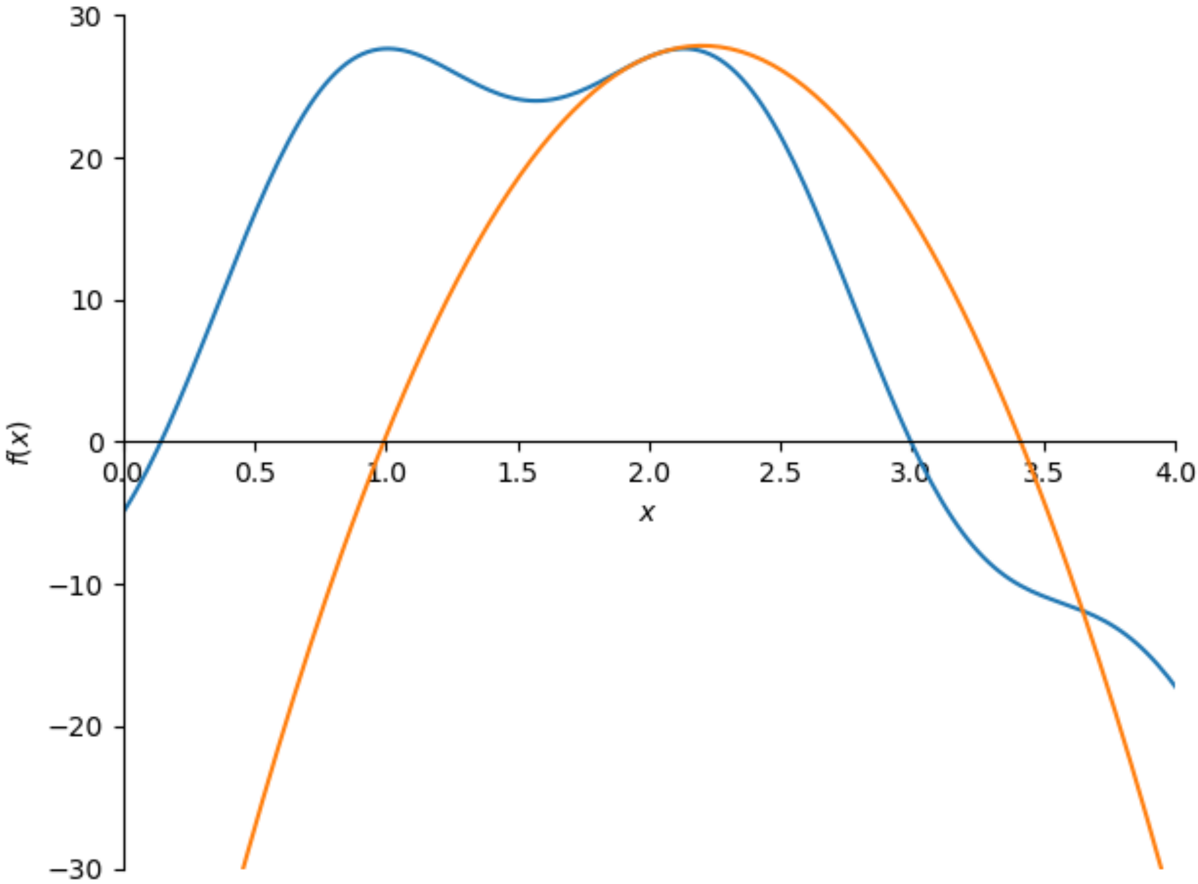
2.1. TE formulė

$$f(x) = f(x_0) + (x - x_0) * \frac{f'(x_0)}{1!} + (x - x_0)^2 * \frac{f''(x_0)}{2!} + \cdots$$

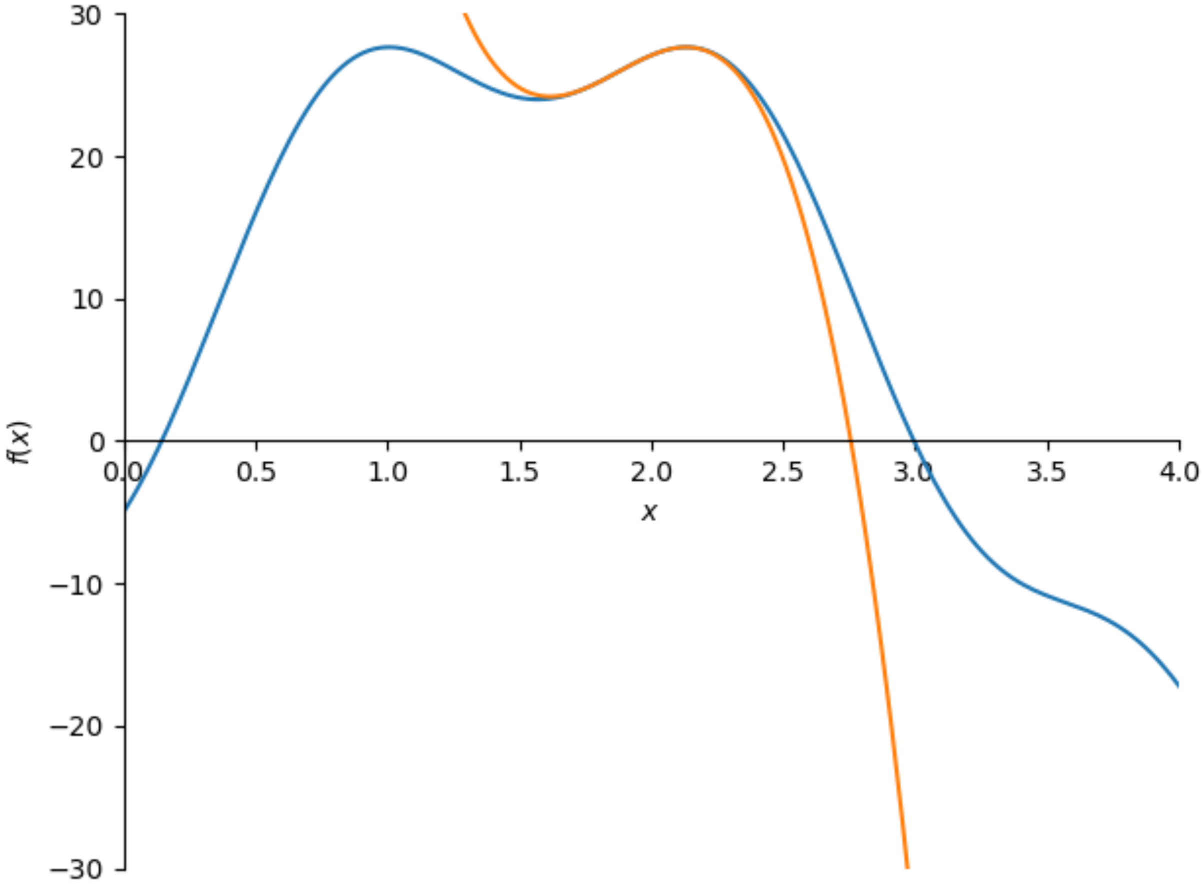
10 funkcija TE formulė

2.2. Atvaizduojama funkcija h(x) su TE, kai TE narių skaičius lygus 3, 4 ir 5

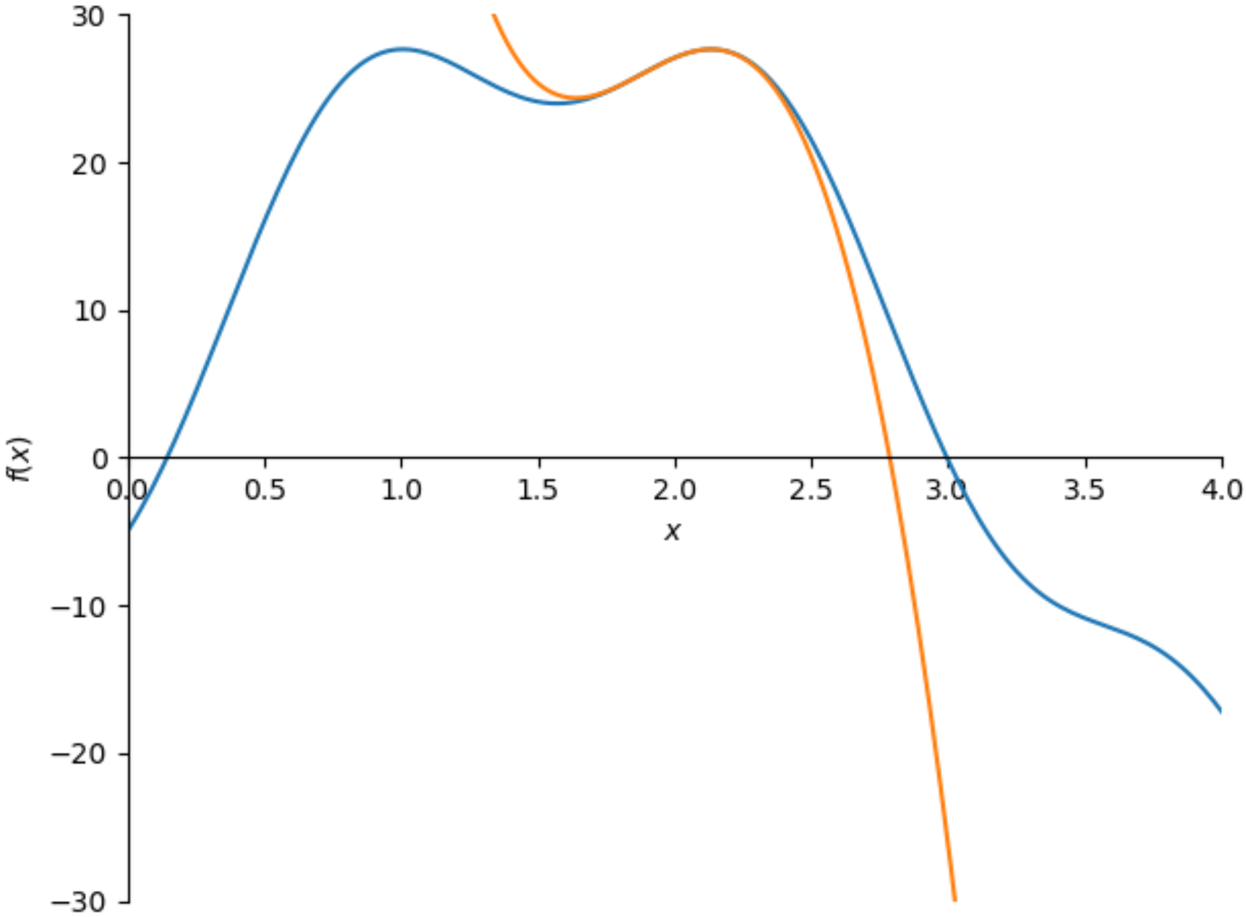
Mėlynai atvaizduojama funkcija h(x), o TE oranžine.



27 pav. Funkcijos, kai TE narių skaičius yra 3



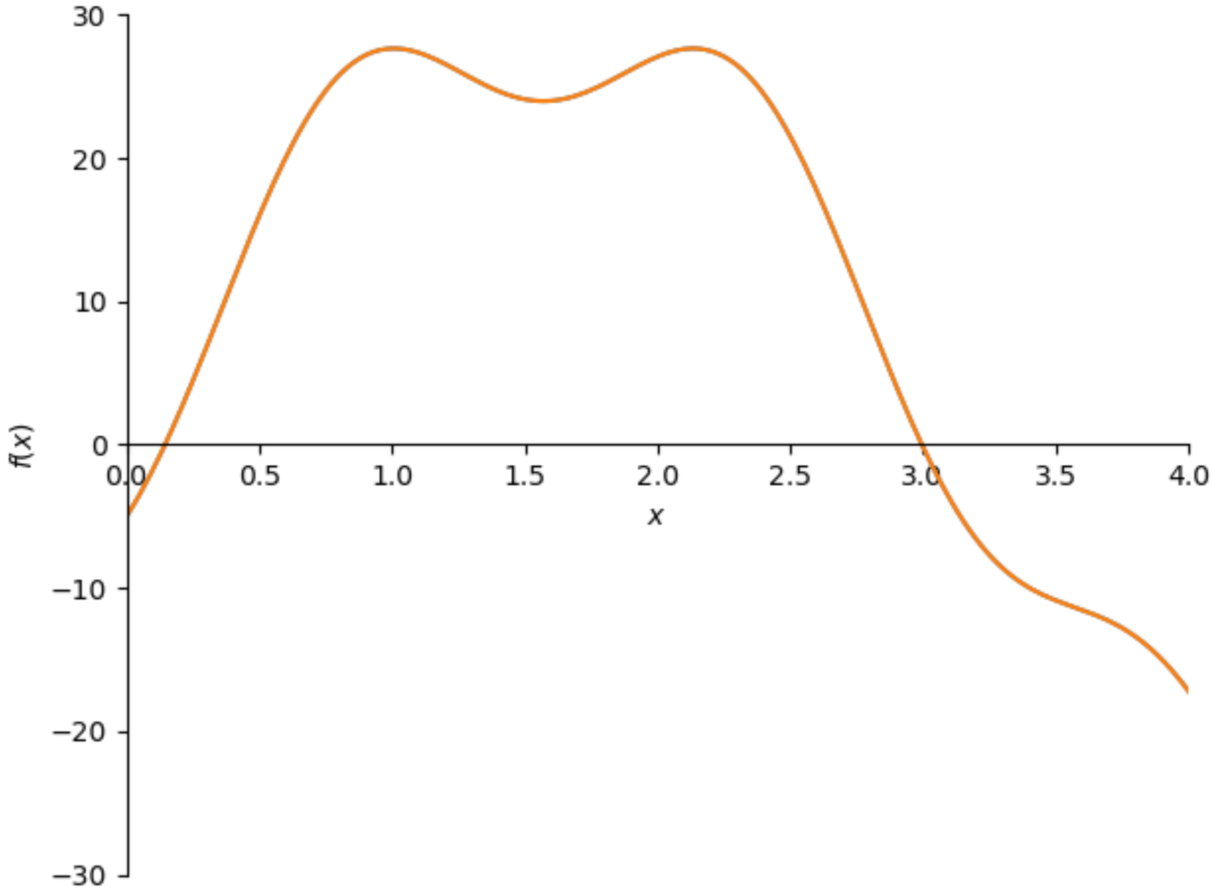
28 pav. Funkcijos, kai TE narių skaičius yra 4



29 pav. Funkcijos, kai TE narių skaičius yra 5

2.3. Grafikas su h(x) funkcija ir reikiamą tikslumą užtikrinanti TE

Grafike praktiškai nėra matoma h(x) funkcija, nes pasiektas reikiamas tikslumas buvo tik tuomet, kai TE yra beveik visiškai tokia pati kaip ir h(x):



30 pav. Reikiamą tikslumą atitinkantis TE daugianaris nubraižytas ant h(x) funkcijos

Algoritmo kodo fragmentas:

7 lentelė Algoritmo kodo fragmentas su reikiamą tikslumą užtikrinančios TE radimu

```
#simboliniai kintamieji
x, f, fp, df = sympy.symbols(('x', 'f', 'fp', 'df'))
#funkcija
fa = 29 * sympy.sin(x) - 5 * sympy.cos(4 * x)

#paimamas intervalo vidurys
#leistina paklaida lyginant saknis
#nurodomas maximalus TE eilutes nariu skaicius
x0 = 2
compare = 1e-4
maxN = 30

#isrusiuotos niutono saknys
niutono_sorted = sorted(niutono)

bestN = None
bestfp = None
bestte_roots = None

#einama per laipsnius ir ieskoma pirmo kurio saknu paklaidos telpa i leistina
intervala
for N in range (1, maxN + 1):
    #dabartine isvestine - startuoja nuo pacios funkcijos
```

```python
    fN = fa
    #teiloro eilutes konstantinis narys
    fp = fN.subs(x, x0)

    #kiekvienam zingsnyje pereinama prie kitos isvestines
    for i in range(1, N + 1):
        fN = fN.diff(x)
        #pridedamas teiloro narys
        fp = fp + fN.subs(x, x0) / math.factorial(i) * (x - x0)**i

    #pavertimas i polinomo objekta pagal x
    P = sympy.Poly(fp, x)
    #grazina koeficientus
    kf = np.array(P.all_coeffs(), dtype=float)

    #randa visas polinomo saknis (realias ir kompleksines)
    saknys = np.roots(kf)
    #atfiltruoja kad liktu tik realios saknys
    saknys = saknys[np.isreal(saknys)]
    saknys = np.real(saknys)
    #paliekamos tik tos saknys kurios patenka i norima intervala
    saknys = [val for val in saknys if start - 1e-12 <= val <= end + 1e-12]
    #isrikiuojamos saknys
    saknys_sorted = sorted(saknys)

    #netinka N jeigu nesutampa rastu saknu skaicius su rastu saknu skaiciumi
is niutono metodo
    if (len(saknys_sorted) != len(niutono_sorted)):
        continue

    #sudeda saknis poromis ir apskaiciuoja skirtuma tarp saknu poru
    diffs = [abs(a - b) for a, b in zip(saknys_sorted, niutono_sorted)]

    #jei visi skirtumai tarp saknu poru patenka i paklaidos intervala tada
issaugome N ir uzbaigiame cikla
    if all(d <= compare for d in diffs):
        bestN = N
        bestfp = fp
        bestte_roots = saknys_sorted
        break
```

2.4. Reikiamą tikslumą užtikrinančios TE analitinė išraiška daugianario pavidalu

Reikiamą tikslumą atitinkanti TE yra sudaryta iš 22 narių, o jos analitinė išraiška atrodo taip:

```
TE: (x − 2)**23*(−134217728*sin(8)/9861761756471625 − 29*cos(2)/258520167388849766400000) + (x − 2)**22*(33554432*cos(8)/428772250281375 − 29*sin(2)/1124000727777607680000) + (x − 2)**21*(29*cos(2)/51090942171709440000 + 16777216*sin(8)/38979295480125) + (x − 2)**20*(2
9*sin(2)/2432902008176640000 − 4194304*cos(8)/1856156927625) + (x − 2)**19*(−4194304*sin(8)/371231385525 − 29*cos(2)/121645100408832000) + (x − 2)**18*(1048576*cos(8)/19538493975 − 29*sin(2)/6402373705728000) + (x − 2)**17*(29*cos(2)/355687428096000 + 524288*sin(8)/21
70943775) + (x − 2)**16*(29*sin(2)/20922789888000 − 131072*cos(8)/127702575) + (x − 2)**15*(−524288*sin(8)/127702575 − 29*cos(2)/1307674368000) + (x − 2)**14*(131072*cos(8)/8513505 − 29*sin(2)/87178291200) + (x − 2)**13*(29*cos(2)/6227020800 + 65536*sin(8)/1216215) +
(x − 2)**12*(29*sin(2)/479001600 − 16384*cos(8)/93555) + (x − 2)**11*(−16384*sin(8)/31185 − 29*cos(2)/39916800) + (x − 2)**10*(4096*cos(8)/2835 − 29*sin(2)/3628800) + (x − 2)**9*(29*cos(2)/362880 + 2048*sin(8)/567) + (x − 2)**8*(29*sin(2)/40320 − 512*cos(8)/63) + (x −
2)**7*(−1024*sin(8)/63 − 29*cos(2)/5040) + (x − 2)**6*(256*cos(8)/9 − 29*sin(2)/720) + (x − 2)**5*(29*cos(2)/120 + 128*sin(8)/3) + (x − 2)**4*(29*sin(2)/24 − 160*cos(8)/3) + (x − 2)**3*(−160*sin(8)/3 − 29*cos(2)/6) + (x − 2)**2*(−29*sin(2)/2 + 40*cos(8)) + (x − 2)*(29
*cos(2) + 20*sin(8)) − 5*cos(8) + 29*sin(2)
```

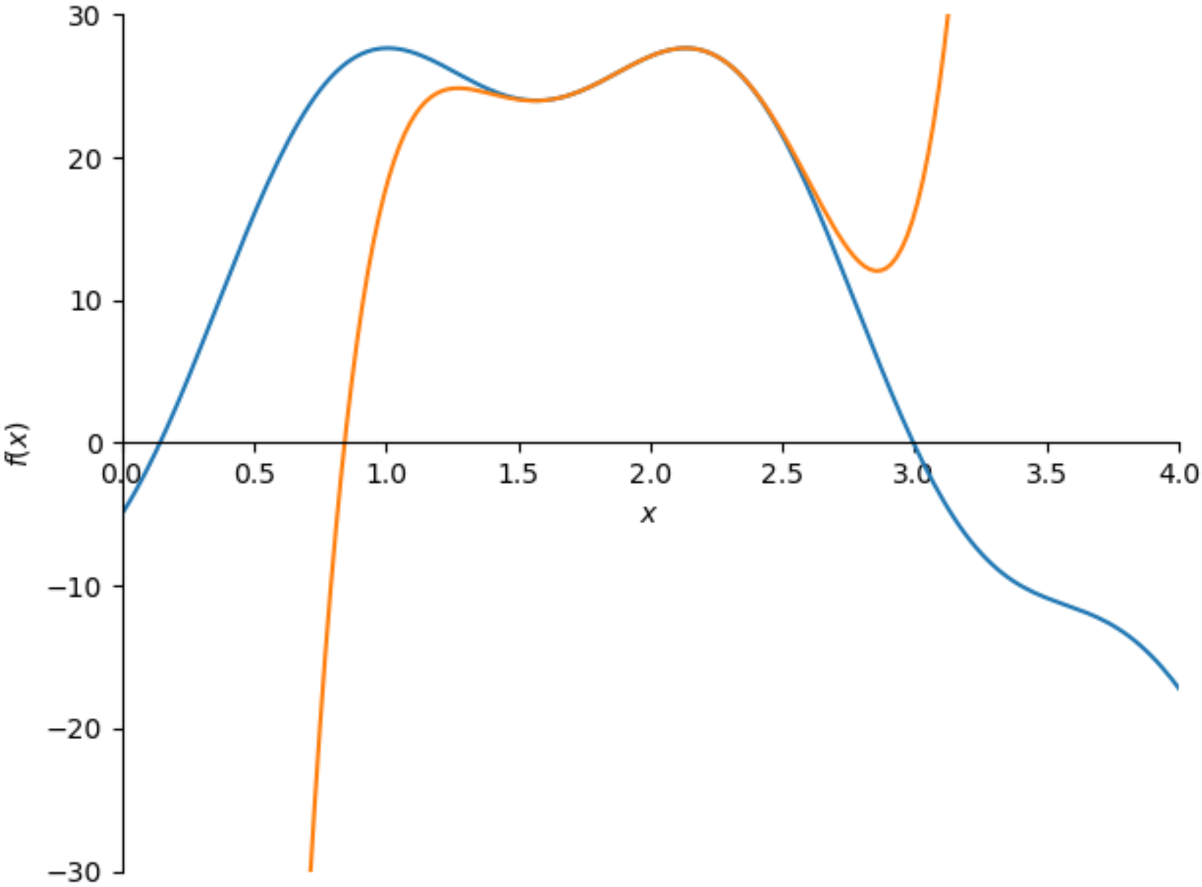
31 pav. TE analitinė išraiška daugianario pavidalu

TE:

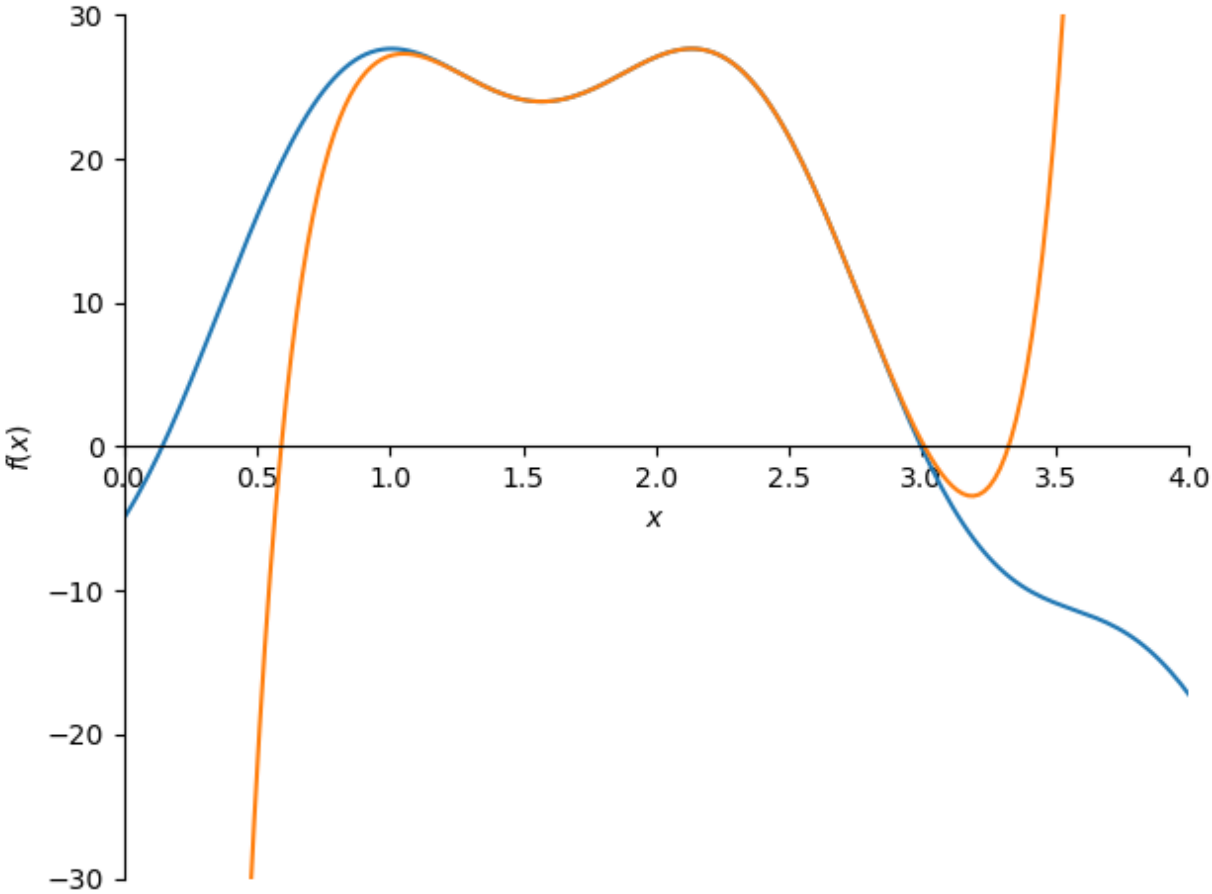
```
(x − 2)**23*(−134217728*sin(8)/9861761756471625 − 29*cos(2)/258520167388849766400
000) + (x − 2)**22*(33554432*cos(8)/428772250281375 − 29*sin(2)/1124000727777607
680000) + (x − 2)**21*(29*cos(2)/51090942171709440000 + 16777216*sin(8)/38979295
480125) + (x − 2)**20*(29*sin(2)/2432902008176640000 − 4194304*cos(8)/1856156927
625) + (x − 2)**19*(−4194304*sin(8)/371231385525 − 29*cos(2)/121645100408832000)
+ (x − 2)**18*(1048576*cos(8)/19538493975 − 29*sin(2)/6402373705728000) + (x − 2
```

```
)**17*(29*cos(2)/355687428096000 + 524288*sin(8)/2170943775) + (x - 2)**16*(29*s
in(2)/20922789888000 - 131072*cos(8)/127702575) + (x - 2)**15*(-524288*sin(8)/12
7702575 - 29*cos(2)/1307674368000) + (x - 2)**14*(131072*cos(8)/8513505 - 29*sin
(2)/87178291200) + (x - 2)**13*(29*cos(2)/6227020800 + 65536*sin(8)/1216215) + (
x - 2)**12*(29*sin(2)/479001600 - 16384*cos(8)/93555) + (x - 2)**11*(-16384*sin(
8)/31185 - 29*cos(2)/39916800) + (x - 2)**10*(4096*cos(8)/2835 - 29*sin(2)/36288
00) + (x - 2)**9*(29*cos(2)/362880 + 2048*sin(8)/567) + (x - 2)**8*(29*sin(2)/40
320 - 512*cos(8)/63) + (x - 2)**7*(-1024*sin(8)/63 - 29*cos(2)/5040) + (x - 2)**
6*(256*cos(8)/9 - 29*sin(2)/720) + (x - 2)**5*(29*cos(2)/120 + 128*sin(8)/3) + (
x - 2)**4*(29*sin(2)/24 - 160*cos(8)/3) + (x - 2)**3*(-160*sin(8)/3 - 29*cos(2)/
6) + (x - 2)**2*(-29*sin(2)/2 + 40*cos(8)) + (x - 2)*(29*cos(2) + 20*sin(8)) - 5
*cos(8) + 29*sin(2)
```

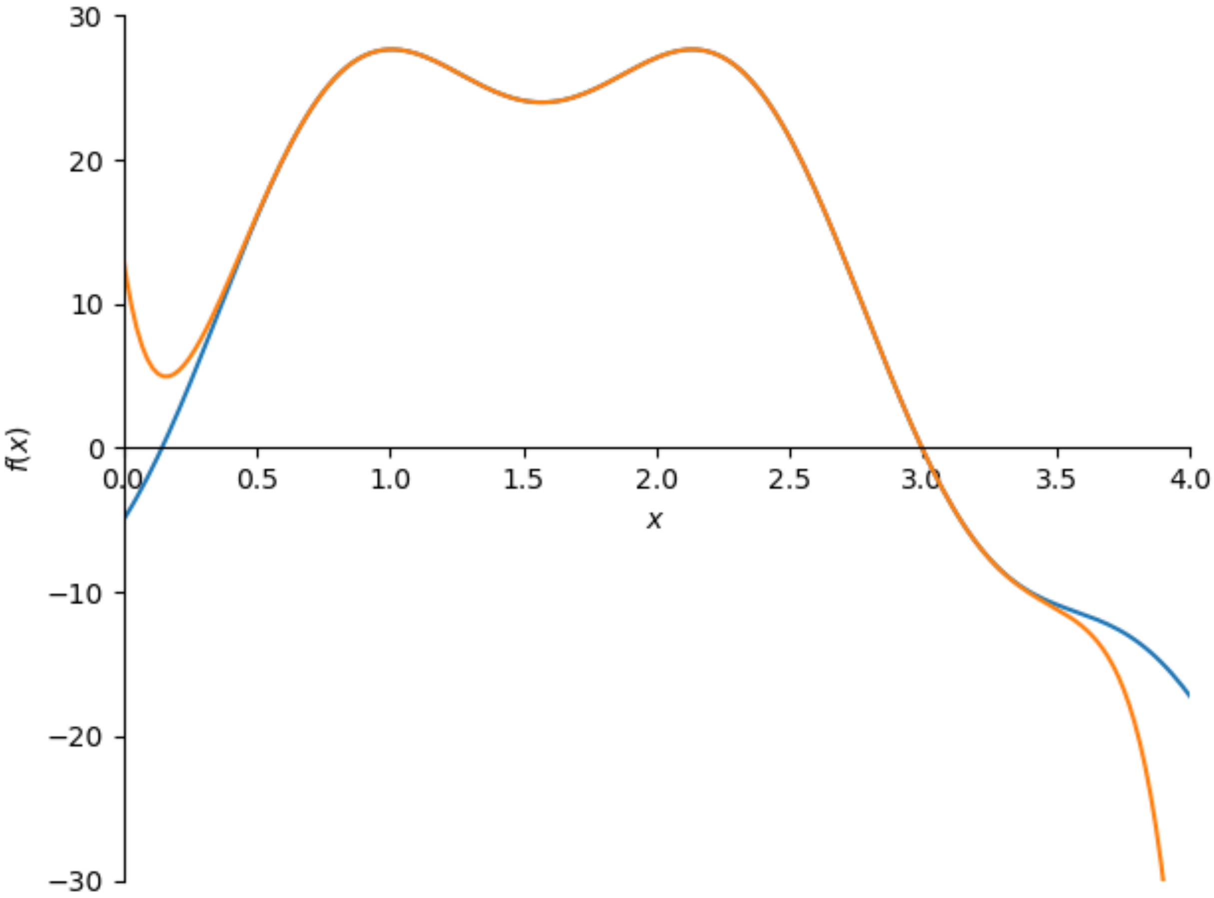
2.5. Grafikai parodantys kaip gerėjo sprendinys priklausomai nuo TE narių skaičiaus



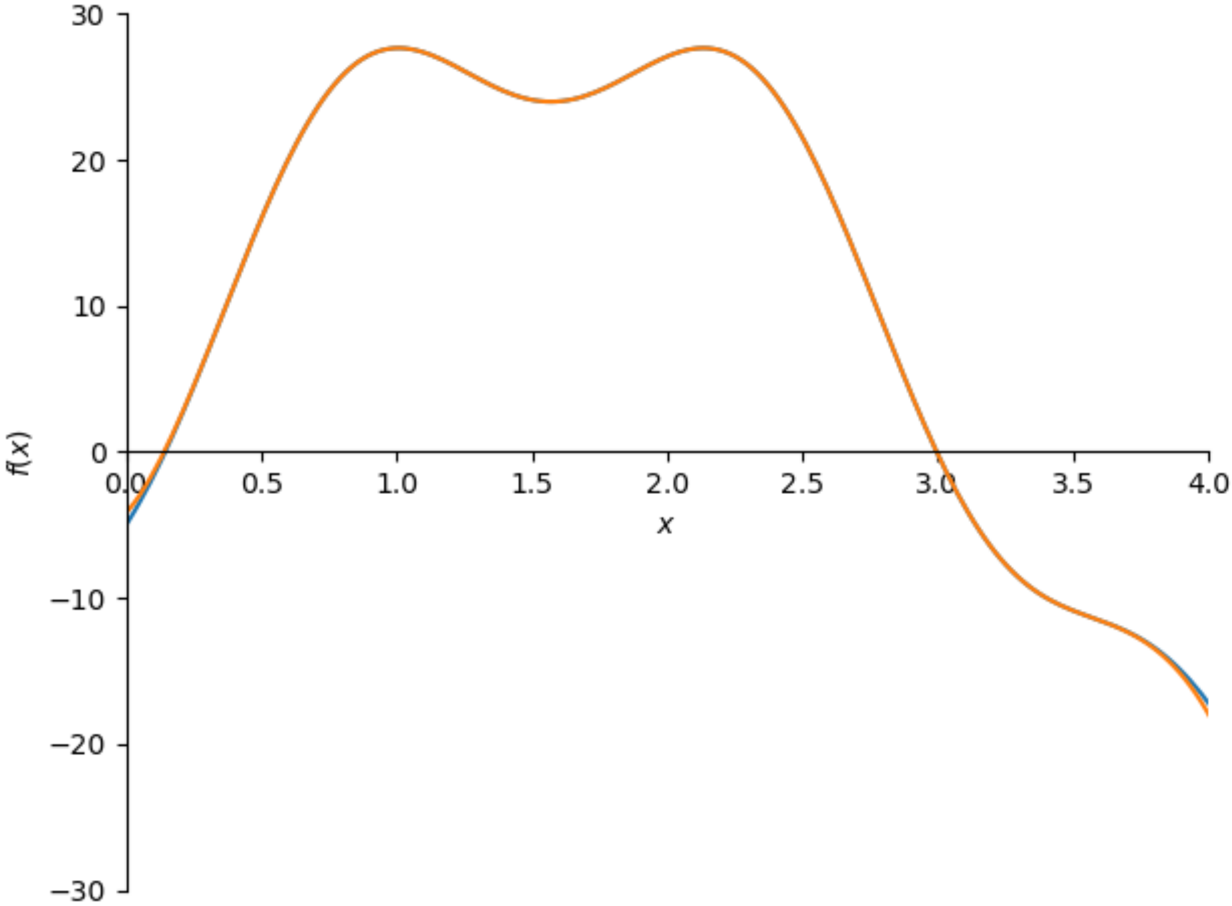
32 pav. Narių skaičius lygus 4



33 pav. Narių skaičius lygus 9

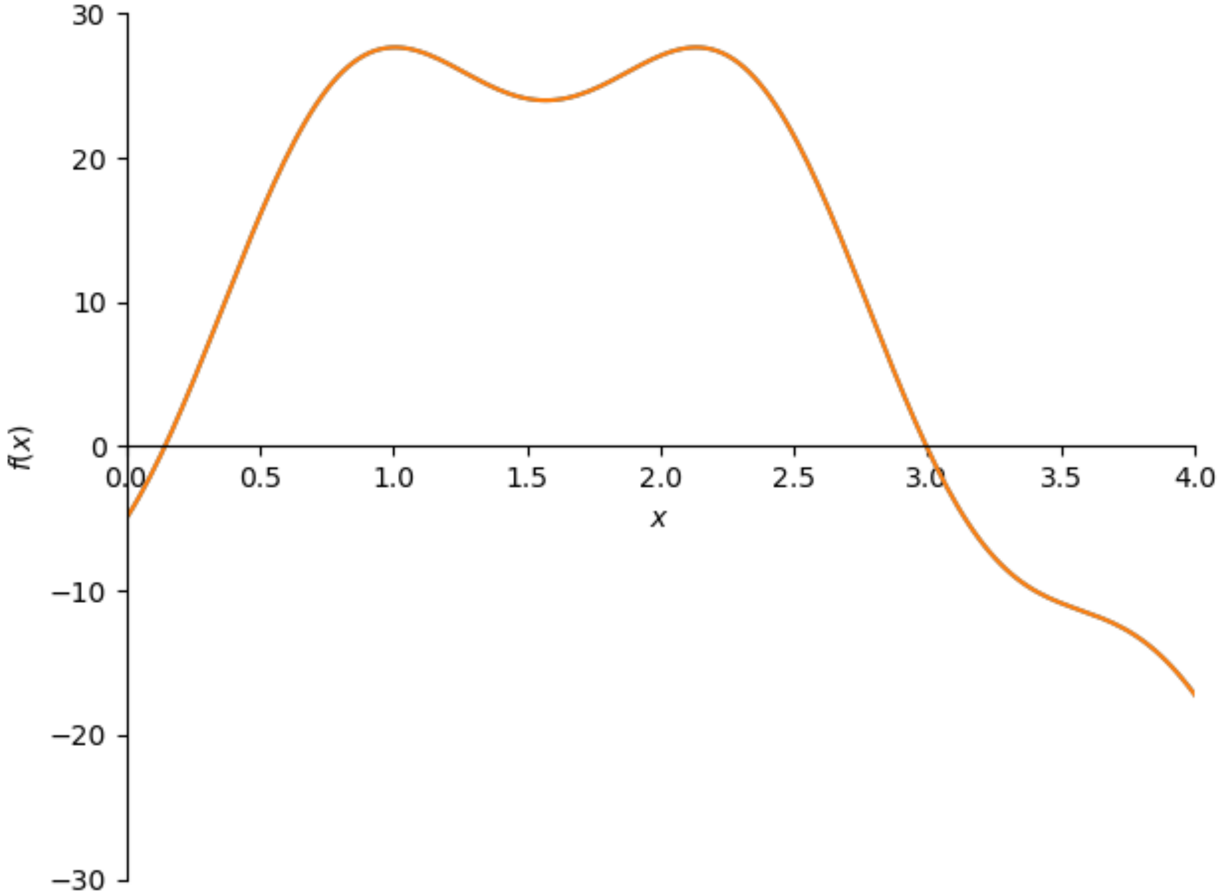


34 pav. Narių skaičius lygus 14



35 pav. Narių skaičius lygus 19

2.5.1. Grafikas, kuris nurodo visą randamų šaknų skaičių nagrinėjame intervale



36 pav. Pakankamo tikslumo grafikas, kuriame yra visos randamos šaknys reikiamu tikslumu

```
1) [0.100000; 0.200000]
Saknis: 0.14479906071536203

2) [2.900000; 3.000000]
Saknis: 2.99679359288042
```

37 pav. Rastos šaknys naudojantis Niutono (liestinių) metodu

```
0.144777222006
2.996793593106
```

38 pav. Gautos šaknys pasinaudojus roots funkcija

Taigi, matome, jog gautos šaknys yra praktiškai identiškos kaip ir apskaičiuotos programine įranga, tai galime teigti, kad gauti rezultatai buvo pakankamai tikslūs ir teisingi.

3. Šaltiniai

1. https://www.wolframalpha.com/
2. JupiterLabs