VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

Vienmatis optimizavimas

Laboratorinio darbo ataskaita

Atliko: Monika Mirbakaitė

VU el. p.: monika.mirbakaite@mif.stud.vu.lt,

Vertino: Vyr. M. Darbuot., Dr. (HP), Julius Žilinskas

Vilnius

2024

TURINYS

ĮVADAS

Šiame laboratoriniame darbe yra realizuoti optimizavimo intervalo dalijimo pusiau, auksinio pjūvio ir Niutono metodo algoritmai. Tikslo funkciją $f(x) = (x2-a)2/b-1$, čia a ir b – studento knygelės numerio “1*1**ab” skaitmenys. Ši funkcija minimizuota minėtais algoritmais intervale [0,10] iki tikslumo $10-4$ bei Niutono metodu nuo $x0 = 5$.

REZULTATŲ PALYGINIMAS

1.1. Rezultatų lentelė

1 lentelė. pateikiami rezultatai rodo, kad Niutono metodas rado sprendimą per mažiausiai žingsnių (7) ir tikslo funkcijos kvietimų (14) iš trijų išbandytų metodų. Tačiau Niutono metodo sprendinys (2.0000000000000018) ir funkcijos minimumo įvertis (-1.0) labiausiai skiriasi nuo kitų minimizavimo rezultatų. Dalijimo pusiau metodas ir aukso pjūvio metodas surado tikslesnius sprendinius (atitinkamai 2.0000076293945312 ir 2.0000124592124724) ir funkcijos minimumo įverčius (atitinkamai -0.9999999998137348 ir -0.9999999995032546).

1 lentelė. Rezultatai.

	Dalijimo pusiau metodas	**Auksinio pjūvio metodas**	**Niutono metodas**
Sprendinys	2.0000076293945312	2.0000124592124724	2.0000000000000018
Funkcijos minimumo įvertis	-0.9999999998137348	-0.9999999995032546	-1.0
Atliktų žingsnių sk.	17	24	7
Funkcijos skaičiavimų sk.	35	26	14

1.2. Rezultatai

1 pav. (dalijimo pusiau metodas), 2 pav. (aukso pjūvio metodas), 3 pav. (niutono metodas) vaizduojami duomenų šaltiniai rezultatų lentelei sudaryti.

1.2.1. Rezultatai naudojant dalijimo pusiau metodą

```
--------DALIJIMO PUSIAU METODAS--------
Dalijimo pusiau metodo iteraciju sk.:  17
Tikslo funkcijos kvietimu sk.:         35
Sprendinys (xm):                       2.0000076293945312
F-jos minimumo ivertis f(xm):         -0.9999999998137348
```

1 pav. rezultatai. Dalijimo pusiau metodas.

1.2.2. Rezultatai naudojant auksinio pjūvio metodą

```
--------AUKSINIO PJUVIO METODAS--------
Auksinio pjuvio metodo iteraciju sk.:  24
Tikslo funkcijos kvietimu sk.:         26
Sprendinys (xm):                       2.0000124592124724
F-jos minimumo ivertis f(xm):         -0.9999999995032546
```

2 pav. rezultatai. Auksinio pjūvio metodas.

1.2.3. Rezultatai naudojant niutono metodą

```
------------NIUTONO METODAS------------
Niutono metodo iteraciju sk.            7
Tikslo funkcijos kvietimu sk.:         14
Sprendinys (xm):                       2.0000000000000018
F-jos minimumo ivertis f(xm):         -1.0
```

3 pav. rezultatai. Niutono metodas.

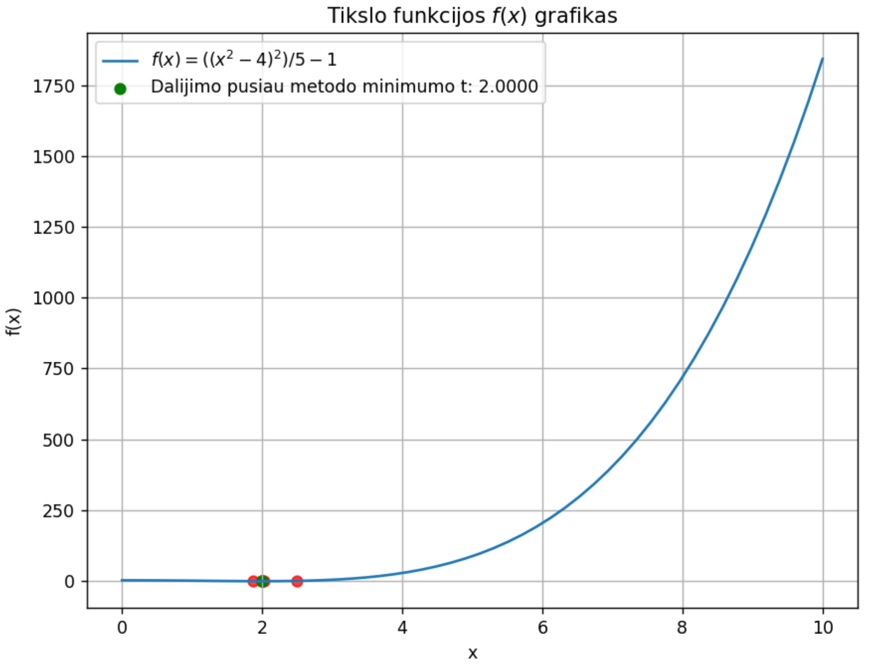
VIZUALIZUOTOS TIKSLO FUNKCIJOS IR JŲ BANDYMO TAŠKAI

4 pav. (dalijimas pusiau metodas), 6 pav. (auksinio pjūvio metodas), 8 pav. (niutono metodas) vaizduojami tikslo funkcijos ir jų bandymų taškų vizualizacijos intervale [0, 10]. 5 pav. (dalijimas pusiau metodas), 7 pav. (auksinio pjūvio metodas), 9 pav. (niutono metodas) vaizduojami tikslo funkcijos ir jų bandymų taškų vizualizacijos intervale [0, 3].

Raudona spalva žymimi bandymo taškai, o žalia – minimumo taškas.

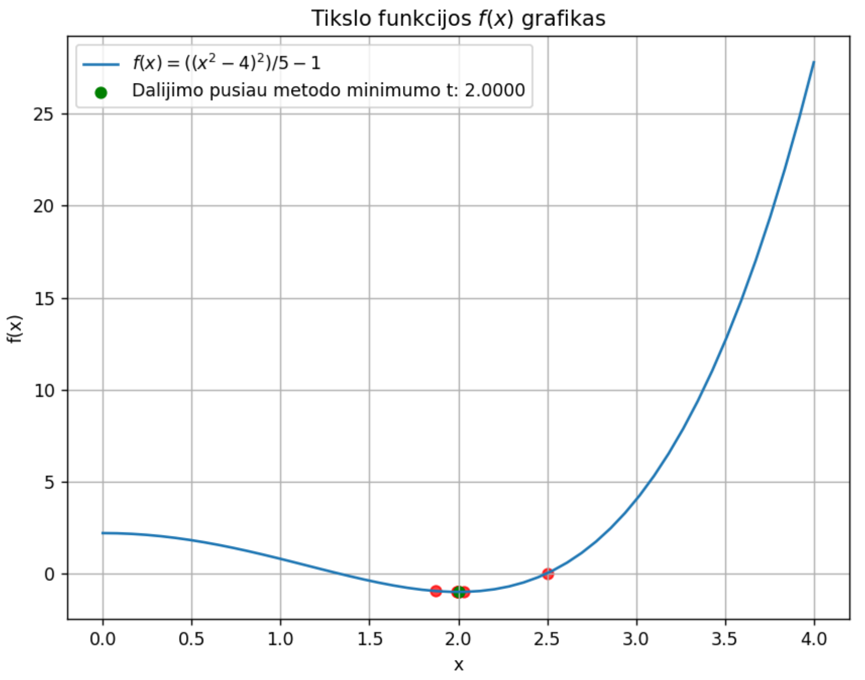
1.3. Dalijimo pusiau algoritmo tikslo funkcijos ir jos bandymo taškų vizualizacija

1.3.1. Dalijimo pusiau algoritmo grafikas (x intervale [0,10])



4 pav. grafikas. Dalijimo pusiau metodas (1).

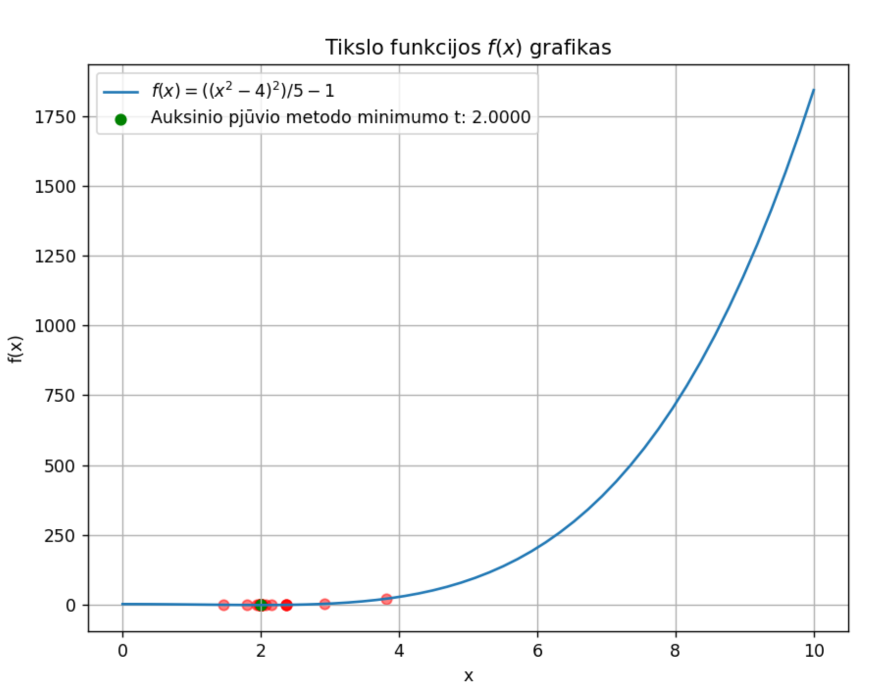
1.3.2. Dalijimo pusiau algoritmo grafikas (x intervale [0,4])



5 pav. grafikas. Dalijimo pusiau metodas (2).

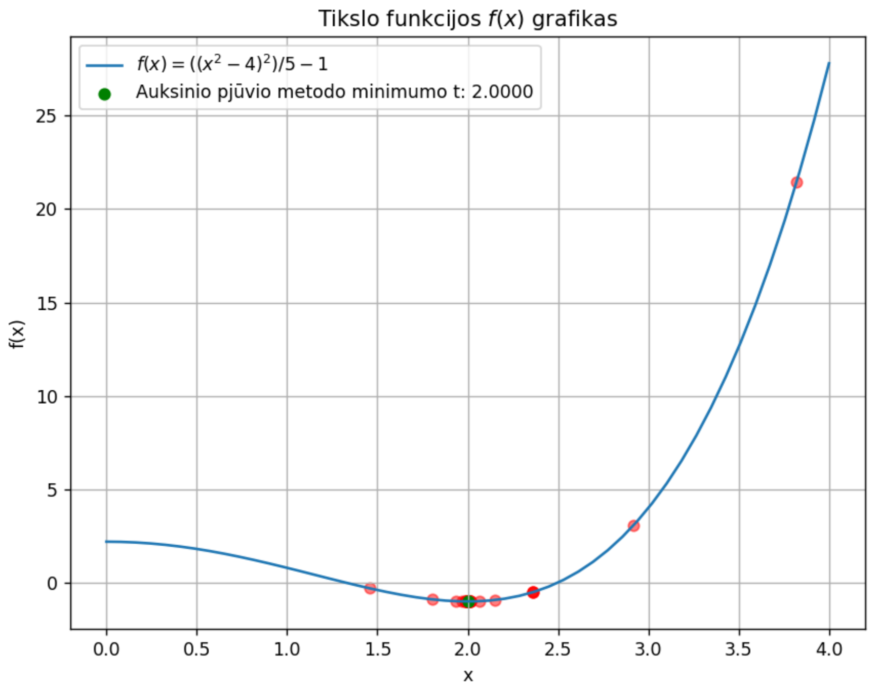
Auksinio pjūvio algoritmo tikslo funkcijos ir jos bandymo taškų vizualizacija

1.3.3. Auksinio pjūvio algoritmo grafikas (x intervale [0,10])



6 pav. grafikas. Auksinio pjūvio metodas (1).

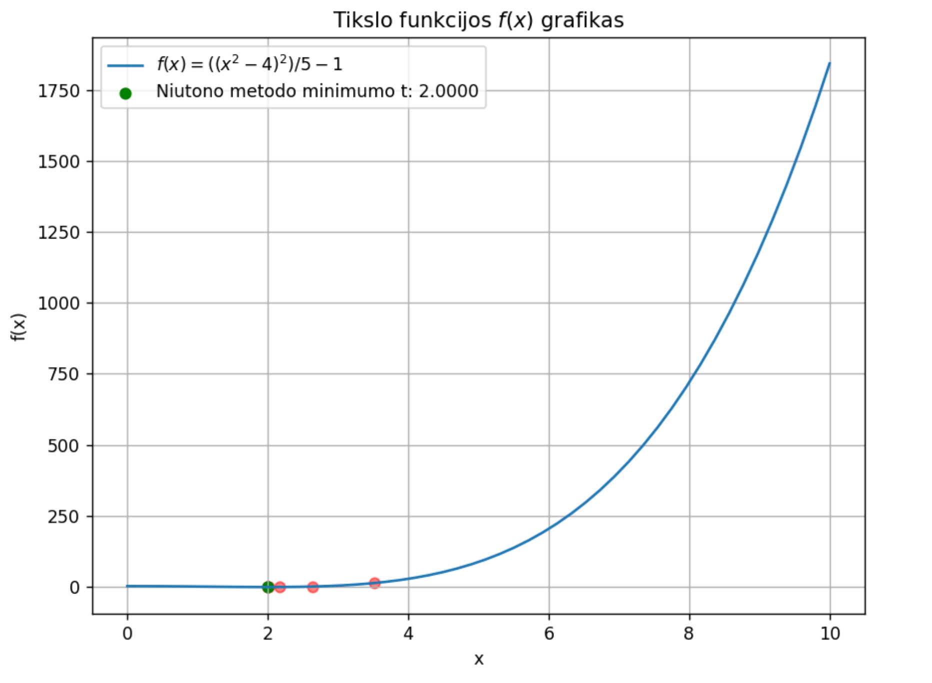
1.3.4. Auksinio pjūvio algoritmo grafikas (x intervale [0,4])



7 pav. grafikas. Auksinio pjūvio metodas (2).

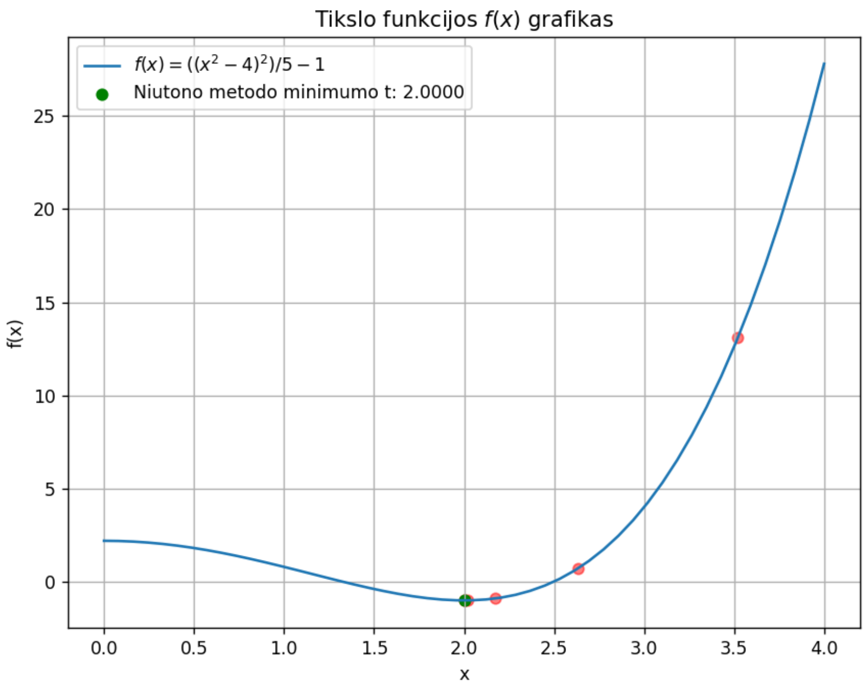
1.4. Niutono algoritmo tikslo funkcijos ir jos bandymo taškų vizualizacija

1.4.1. Niutono algoritmo grafikas (x intervale [0,10])



8 pav. grafikas. Niutono metodas (1).

1.4.2. Niutono algoritmo grafikas (x intervale [0,4])



9 pav. grafikas. Niutono metodas (2).

MINIMIZAVIMO ALGORITMŲ REALIZACIJOS

Toliau pateikiamas kodas, kuriuo buvo gauti aukščiau esantys rezultatai.

1.5. Dalijimo pusiau metodo realizacija

```
import numpy as np
import matplotlib.pyplot as plt

studento_nr = "1717745"
studento_numeris = [int(char) for char in studento_nr]
#studento_numeris = [1, 3, 1, 3 , 3 , 7 ,6]

a = studento_numeris[5]
if studento_numeris[6] == 0:
    digits_sum = sum(studento_numeris)
    while digits_sum > 9:
        digits_sum = sum(int(digit) for digit in str(digits_sum))
    b = digits_sum
else:
    b = studento_numeris[6]

# f(x) = ((x^2 - a)^2) / b - 1
def f(x):
    return ((x ** 2 - a) ** 2) / b - 1

def dalijimas_pusiau(func, interval, epsilon):
    l, r = interval
    L = r - l
    xm = (l + r) / 2
    fxm = func(xm)
    iter = 0
    fja = 1
    while True:
        iter+=1
```

```
        fja+=2
        x1 = l + L / 4
        x2 = r - L / 4
        fx1 = func(x1)
        fx2 = func(x2)
        if fx1 < fxm:
            r = xm
            xm = x1
            fxm = fx1
        elif fx2 < fxm:
            l = xm
            xm = x2
            fxm = fx2
        else:
            l = x1
            r = x2
        L = r - l
        if L < epsilon:
             break
        plt.scatter(xm, f(xm), color='red', alpha=0.5)
    print("--------DALIJIMO PUSIAU METODAS--------")
    print ("Dalijimo pusiau metodo iteraciju sk.: ", iter)
    print ("Tikslo funkcijos kvietimu sk.:        ", fja)
    print ("Sprendinys (xm):                      ", xm)
    print ("F-jos minimumo ivertis f(xm):        ", f(xm))
    return xm

# Sukurti x reikšmes nuo 0 iki 10
x_values = np.linspace(0, 10)
y_values = f(x_values)
plt.figure(figsize=(8, 6))
plt.plot(x_values, y_values, label=f'$f(x) = ((x^2 - {a})^2) / {b} - 1$')
plt.title('Tikslo funkcijos $f(x)$ grafikas')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.legend()

interval = [0, 10]
epsilon = 1e-4

dalijimas_pusiau_min = dalijimas_pusiau(f, interval, epsilon)
plt.scatter(dalijimas_pusiau_min, f(dalijimas_pusiau_min), color='green',
label=f'Dalijimo pusiau metodo minimumo t: {dalijimas_pusiau_min:.4f}')

plt.legend()
plt.show()
```

1.6. Auksinio pjūvio metodo realizacija

```
import numpy as np
import matplotlib.pyplot as plt

studento_nr = "1717745"
studento_numeris = [int(char) for char in studento_nr]

a = studento_numeris[5]
if studento_numeris[6] == 0:
    digits_sum = sum(studento_numeris)
    while digits_sum > 9:
        digits_sum = sum(int(digit) for digit in str(digits_sum))
    b = digits_sum
else:
    b = studento_numeris[6]

# f(x) = ((x^2 - a)^2) / b - 1
```

```
def f(x):
    return ((x ** 2 - a) ** 2) / b - 1

def auksinio_pjuvio(func, interval, epsilon):
  l, r = interval
  tau = (-1 + (5 ** 0.5)) / 2
  L = r - l
  x1 = r - tau * L
  x2 = l + tau * L
  fx1 = f(x1)
  fx2 = f(x2)
  iter = 0
  fja = 2
  tBandymo = []
  while True:
    iter+=1
    fja+=1
    if fx2 < fx1:
      l = x1
      L = r - l
      x1 = x2
      x2 = l + tau * L
      fx1 = fx2
      fx2 = f(x2)
    else:
      r = x2
      L = r - l
      x2 = x1
      x1 = r - tau * L
      fx2 = fx1
      fx1 = f(x1)
      if L<epsilon:
         break
    plt.scatter([x1, x2], [f(x1), f(x2)], color='red', alpha=0.5)
  print("--------AUKSINIO PJUVIO METODAS--------")
  print ("Auksinio pjuvio metodo iteraciju sk.: ", iter)
  print ("Tikslo funkcijos kvietimu sk.:        ", fja)
  print ("Sprendinys (xm):                      ", (x1 + x2) / 2)
  print ("F-jos minimumo ivertis f(xm):        ", f((x1 + x2) / 2))

  return (x1 + x2) / 2
x_values = np.linspace(0, 10)
y_values = f(x_values)

plt.figure(figsize=(8, 6))
plt.plot(x_values, y_values, label=f'$f(x) = ((x^2 - {a})^2) / {b} - 1$')
plt.title('Tikslo funkcijos $f(x)$ grafikas')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.legend()

interval = [0, 10]
epsilon = 1e-4

auksinio_pjuvio_min = auksinio_pjuvio(f, interval, epsilon)
plt.scatter(auksinio_pjuvio_min, f(auksinio_pjuvio_min), color='green',
label=f'Auksinio pjūvio metodo minimumo t: {auksinio_pjuvio_min:.4f}')

plt.legend()
plt.show()
```

1.7. Niutono metodo realizacija

```
import numpy as np
import matplotlib.pyplot as plt
```

```
studento_nr = "1717745"
studento_numeris = [int(char) for char in studento_nr]
#studento_numeris = [1, 3, 1, 3 , 3 , 7 ,6]

a = studento_numeris[5]
if studento_numeris[6] == 0:
    digits_sum = sum(studento_numeris)
    while digits_sum > 9:
        digits_sum = sum(int(digit) for digit in str(digits_sum))
    b = digits_sum
else:
    b = studento_numeris[6]

# f(x) = ((x^2 - a)^2) / b - 1
def f(x):
    return ((x ** 2 - a) ** 2) / b - 1

def df(x):
    return 4 * x * (x ** 2 - a) / b

def dff(x):
    return (12 * x ** 2 - 4*a)/b

def niutono_metodas(f, df, x0, epsilon):
    iter = 0
    fja = 0
    while True:
        iter+=1
        fja+=2
        x1 = x0 - df(x0) / dff(x0)
        if abs(x0 - x1) < epsilon:
            break
        x0 = x1
        plt.scatter(x0, f(x0), color='red', alpha=0.5)
    print("------------NIUTONO METODAS------------")
    print("Niutono metodo iteraciju sk.           ", iter)
    print ("Tikslo funkcijos kvietimu sk.:        ", fja)
    print ("Sprendinys (xm):                      ", x1)
    print ("F-jos minimumo ivertis f(xm):        ", f(x1))

    return x1

x_values = np.linspace(0, 10)
y_values = f(x_values)

plt.figure(figsize=(8, 6))
plt.plot(x_values, y_values, label=f'$f(x) = ((x^2 - {a})^2) / {b} - 1$')
plt.title('Tikslo funkcijos $f(x)$ grafikas')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.legend()

x0 = 5
interval = [0, 10]
epsilon = 1e-4

niutiono_metodas_min = niutono_metodas(f, df, x0, epsilon)
plt.scatter(niutiono_metodas_min, f(niutiono_metodas_min), color='green',
label=f'Niutono metodo minimumo t: {niutiono_metodas_min:.4f}')

plt.legend()
plt.show
```

IŠVADOS

Realizavau minėtus minimizavimo algoritmus ir gavau šias išvadas:

1. Palyginus visus 3 algoritmus, matome, kad Niutono metodas rado sprendimą per mažiausiai žingsnių, tikslo funkcijos kvietimų.
2. Palyginus visus 3 algoritmus, matome, kad Niutono metodo sprendinys ir funkcijos minimumo įvertis labiausiai skiriasi nuo kitų minimizavimo rezultatų.
3. Palyginus visus 3 algoritmus, matome, kad dalijimo pusiau metodas ir aukso pjūvio metodas surado tikslesnius sprendinius ir funkcijos minimumo įverčius.