VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
INFORMATIKOS KATEDRA

Optimizavimo metodai

I laboratorinis darbas - vienmatis optimizavimas

Atliko: 3 kurso 2 grupės studentė
Gabrielė Rinkevičiūtė

Vilnius
2024

Turinys

Įvadas

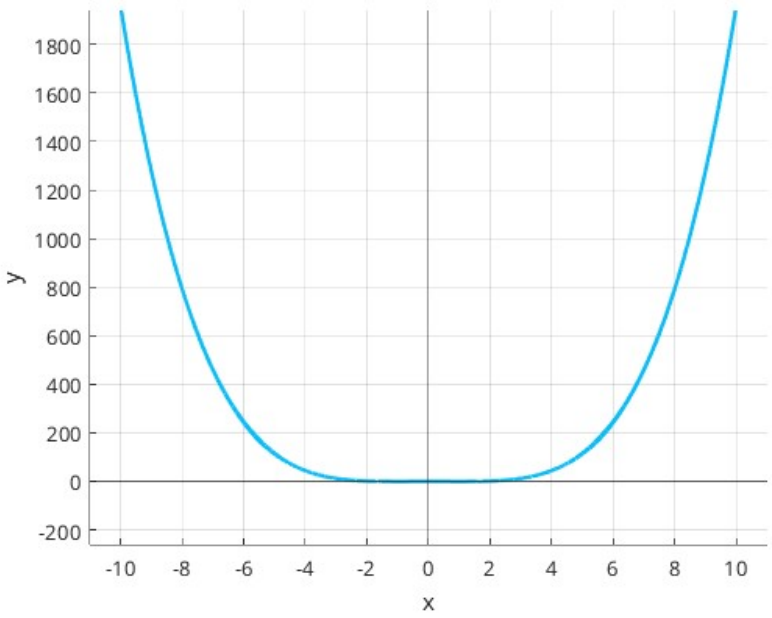
Šiame laboratoriniame darbe yra analizuojami trys funkcijos minimizavimo metodai: dalijimo pusiau metodas, auksinio pjūvio metodas, Niutono metodas. Šie metodai yra tiriami, juos pritaikant funkcijai

$$f(x) = \frac{(x^2 - a)^2}{b} - 1 \tag{1}$$

Šiuo atveju $a = 1, b = 5$, todėl

$$f(x) = \frac{(x^2 - 1)^2}{5} - 1 \tag{2}$$

Žemiau galima pamatyti šios funkcijos grafinį atvaizdavimą [1].



1 pav. Funkcijos $f(x)$ grafikas

Metodai yra lyginami iteracijų skaičiaus, tikslo funkcijos iškvietimo dažnio atžvilgiais. Metodų analizavimui buvo pasirinkta MATLAB programavimo kalba.

1. Dalijimo pusiau metodas

Žemiau pateiktas algoritmo pseudokodas iliustruoja dalijimo pusiau metodo veikimo principą. Šio algoritmo įgyvendinimą MATLAB kalba galima rasti čia.

Algorithm 1 Dalijimo pusiau metodas funkcijai $f(x)$

Require: $f(x), l \in D_f, r \in D_f, l < r$
Ensure: $x_{min}, \min_{x \in [l;r]} f$
  $l \leftarrow 0$
  $r \leftarrow 10$
  $L \leftarrow r - l$
  $x_m \leftarrow \frac{l+r}{2}$
  while $L > 10^{-4}$ **do**
    $x_1 \leftarrow l + \frac{L}{4}$
    $x_2 \leftarrow r - \frac{L}{4}$
    $y_1 \leftarrow f(x_1)$
    $y_2 \leftarrow f(x_2)$
    $y_m \leftarrow f(x_m)$
    if $y_1 < y_m$ **then**
      $r \leftarrow x_m$
      $x_m \leftarrow x_1$
    else if $y_2 < y_m$ **then**
      $l \leftarrow x_m$
      $x_m \leftarrow x_2$
    else
      $l \leftarrow x_1$
      $r \leftarrow x_2$
    end if
    $L \leftarrow r - l$
  end while
  return $x_m, f(x_m)$

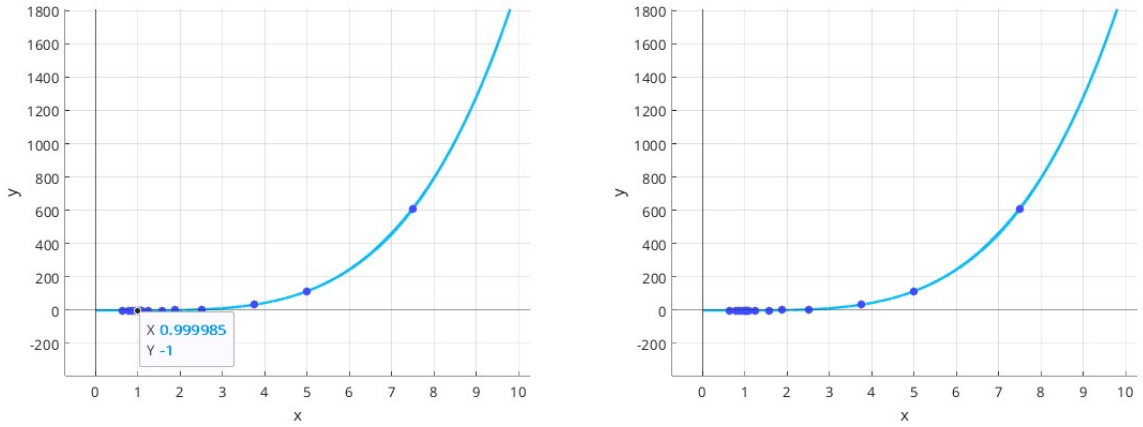
Pritaikius šį metodą aukščiau minėtai funkcijai, algoritmas sustoja po 17 iteracijų. Galutinis rezultatas yra paskutinis apskaičiuotas x_m. Šiuo atveju tai būtų $x_m = 0.99998$.

Kiekvienos iteracijos tarpinius duomenis galima rasti žemiau pateiktoje lentelėje [1].

	x_1	x_2	x_m
Iteracija nr. 1	2.50000	7.50000	5.00000
Iteracija nr. 2	1.25000	3.75000	2.50000
Iteracija nr. 3	0.62500	1.87500	1.25000
Iteracija nr. 4	0.93750	1.56250	1.25000
Iteracija nr. 5	0.78125	1.09375	0.93750
Iteracija nr. 6	0.85938	1.01563	0.93750
Iteracija nr. 7	0.97656	1.05469	1.01563
Iteracija nr. 8	0.99609	1.03516	1.01563
Iteracija nr. 9	0.98633	1.00586	0.99609
Iteracija nr. 10	0.99121	1.00098	0.99609
Iteracija nr. 11	0.99854	1.00342	1.00098
Iteracija nr. 12	0.99976	1.00220	1.00098
Iteracija nr. 13	0.99915	1.00037	0.99976
Iteracija nr. 14	0.99945	1.00006	0.99976
Iteracija nr. 15	0.99991	1.00021	1.00006
Iteracija nr. 16	0.99998	1.00014	1.00006
Iteracija nr. 17	0.99995	1.00002	0.99998
-	-	-	0.99998

1 lentelė. Dalijimo pusiau metodo iteracijos

Taip pat grafinis pasirinktų taškų atvaizdavimas matomas žemiau pateiktuose paveikslėliuose [2].



2 pav. Dalijimo pusiau metodo pasirinktų taškų grafikai

Neskaičiuojant iškvietimų, reikalingų grafikui pateikti, dalijimo pusiau metodo algoritmui bendrai prireikė 51 tikslo funkcijos iškvietimo.

2. Auksinio pjūvio metodas

Žemiau pateiktas algoritmo pseudokodas iliustruoja auksinio pjūvio metodo veikimo principą. Šio algoritmo įgyvendinimą MATLAB kalba galima rasti čia.

Algorithm 2 Auksinio pjūvio metodas funkcijai $f(x)$

Require: $f(x), l \in D_f, r \in D_f, l < r$
Ensure: $x_{min}, \min_{x\in[l;r]} f$
  $L \leftarrow r - l$
  $x_1 \leftarrow r - \tau \cdot L$ ▷ Čia ir toliau τ - Fibonačio skaičius ($\tau = 0{,}61803\ldots$).
  $x_2 \leftarrow l + \tau \cdot L$
  while $L > 10^{-4}$ **do**
    $x_1 \leftarrow l + \frac{L}{4}$
    $x_2 \leftarrow r - \frac{L}{4}$
    $y_1 \leftarrow f(x_1)$
    $y_2 \leftarrow f(x_2)$
    if $y_2 < y_1$ **then**
      $l \leftarrow x_1$
      $L \leftarrow r - l$
      $x_1 \leftarrow x_2$
      $x_2 \leftarrow l + \tau \cdot L$
    else
      $r \leftarrow x_2$
      $L \leftarrow r - l$
      $x_2 \leftarrow x_1$
      $x_1 \leftarrow r - \tau \cdot L$
    end if
  end while
  $x_m \leftarrow \frac{x_1+x_2}{2}$
  return $x_m, f(x_m)$

Šiam algoritmui prireikia daugiau - 26 iteracijų. Galutinis rezultatas skaičiuojamas naudojant vidurio taško formulę. Šiuo atveju tai būtų

$$\frac{x_1 + x_2}{2} = \frac{1.00002 + 0.99998}{2} = 1 \tag{3}$$

Čia x_1 ir x_2 yra paskutinės iteracijos atitinkamos reikšmės.

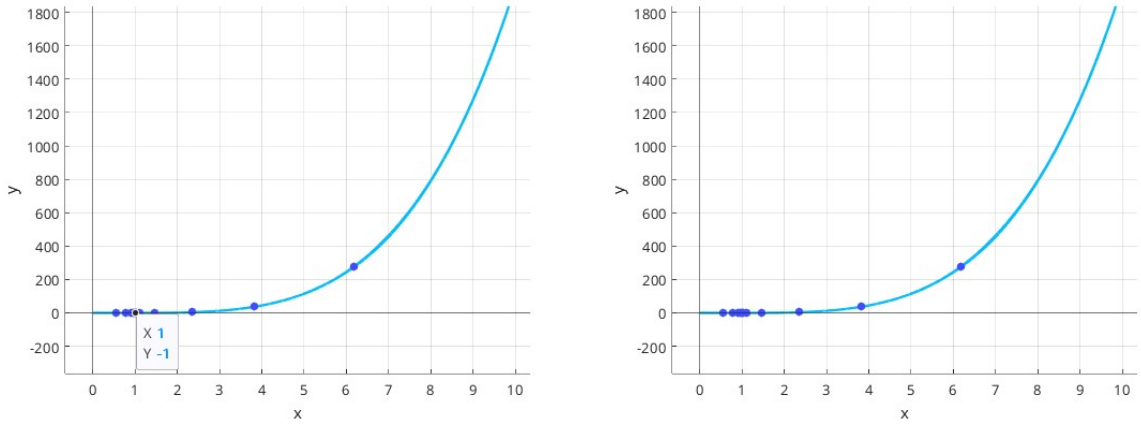
Kiekvienos iteracijos tarpinius duomenis galima rasti žemiau pateiktose lentelėse [2].

	x_1	x_2
Iteracija nr. 1	5.00000	2.50000
Iteracija nr. 2	7.50000	2.50000
Iteracija nr. 3	1.25000	3.75000
Iteracija nr. 4	1.25000	0.62500
Iteracija nr. 5	1.87500	1.25000
Iteracija nr. 6	0.93750	1.56250
Iteracija nr. 7	0.93750	0.78125
Iteracija nr. 8	1.09375	0.93750
Iteracija nr. 9	0.85938	1.01563
Iteracija nr. 10	1.01563	0.97656
Iteracija nr. 11	1.05469	1.01563
Iteracija nr. 12	0.99609	1.03516
Iteracija nr. 13	0.99609	0.98633

	x_1	x_2
Iteracija nr. 14	1.00586	0.99609
Iteracija nr. 15	0.99121	1.00098
Iteracija nr. 16	1.00098	0.99854
Iteracija nr. 17	1.00342	1.00098
Iteracija nr. 18	0.99976	1.00220
Iteracija nr. 19	0.99976	0.99915
Iteracija nr. 20	1.00037	0.99976
Iteracija nr. 21	0.99945	1.00006
Iteracija nr. 22	1.00006	0.99991
Iteracija nr. 23	1.00021	1.00006
Iteracija nr. 24	0.99998	1.00014
Iteracija nr. 25	0.99998	0.99995
Iteracija nr. 26	1.00002	0.99998

2 lentelė. Auksinio pjūvio metodo iteracijos

Taip pat grafinis pasirinktų taškų atvaizdavimas matomas žemiau pateiktuose paveikslėliuose [3].



3 pav. Auksinio pjūvio metodo pasirinktų taškų grafikai

Neskaičiuojant iškvietimų, reikalingų grafikui pateikti, auksinio pjūvio metodo algoritmui bendrai prireikė 48 tikslo funkcijos iškvietimų.

3. Niutono metodas

Šiame skyriuje pateikiamas metodas skiriasi nuo aukščiau paminėtų metodų tuo, kad jis veikia ne intervalų atmetimo principu, bet iki norimo tikslumo iteratyviai pritaikant formulę

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)} \tag{4}$$

Žemiau pateiktas algoritmo pseudokodas iliustruoja Niutono metodo veikimo principą. Šio algoritmo įgyvendinimą MATLAB kalba galima rasti čia.

Algorithm 3 Niutono metodas $f(x)$

Require: $x_0 \in R$
Ensure: $x_{min}, \min f$
  $p \leftarrow x_0$ ▷ Čia p ir n atitinka formulės x_i ir x_{i+1} reikšmes.
  $n \leftarrow p - \frac{f'(p)}{f''(p)}$
  while $\mid p - n \mid > 10^{-4}$ **do**
    $p \leftarrow n$
    $n \leftarrow p - \frac{f'(p)}{f''(p)}$
  end while
  return $n, f(n)$

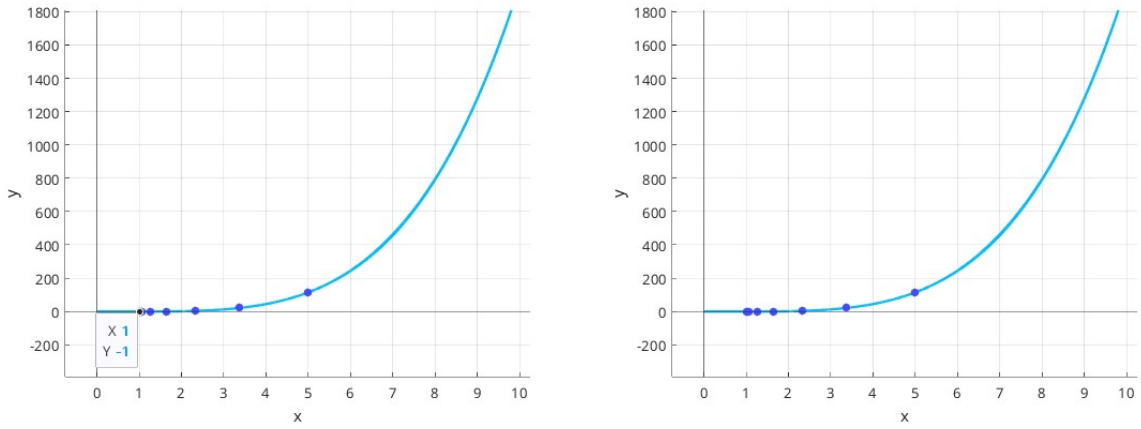
Šio metodo algoritmui prireikė mažiausiai - tik 9 iteracijų. Galutinis rezultatas yra paskutinis suskaičiuotas x_{i+1}.

Kiekvienos iteracijos tarpinius duomenis galima rasti žemiau pateiktoje lentelėje [3].

	x_{i+1}
Iteracija nr. 1	5.00000
Iteracija nr. 2	3.37838
Iteracija nr. 3	2.32001
Iteracija nr. 4	1.64878
Iteracija nr. 5	1.25280
Iteracija nr. 6	1.06041
Iteracija nr. 7	1.00480
Iteracija nr. 8	1.00003
Iteracija nr. 9	1.00000

3 lentelė. Niutono metodo iteracijos

Taip pat grafinis pasirinktų taškų atvaizdavimas matomas žemiau pateiktuose paveikslėliuose [4].



4 pav. Niutono metodo pasirinktų taškų grafikai

Neskaičiuojant iškvietimų, reikalingų grafikui pateikti, Niutono metodo algoritmui bendrai prireikė 16 tikslo funkcijos iškvietimų.

Išvados

Žemiau pateiktoje lentelėje [4] galima pamatyti, kaip skiriasi metodų efektyvumas, bei jų gauti rezultatai.

Metodas	Dalijimo pusiau	Auksinio pjūvio	Niutono
Iteracijų sk.	17	26	10
Tikslo funkcijos kvietimų sk.	51	48	16
Gautas minimumo taškas	0.999985	1	1
Gautas minimumas	-1	-1	-1

4 lentelė. Metodų palyginimo lentelė

Lyginant tiek pagal tikslo funkcijos iškvietimo skaičių, tiek pagal iteracijų kiekį, efektyviausiai su užduotimi susitvarko Niutono metodas.

Kalbant apie mažiausiai efektyvų metodą, nėra vienareikšmiško atsakymo. Lyginant pagal iteracijų kiekį, auksinio pjūvio metodas veikia prasčiausiai, tačiau palyginus dalijimo pusiau ir auksinio pjūvio metodus tikslo funkcijos kvietimų skaičiaus atžvilgiu, pirmasis metodas vis dėl to nusileidžia efektyvumu pirmajam.

Taip pat verta atkreipti dėmesį į gautus rezultatus - auksinio pjūvio ir Niutono metodų gauti minimo taškai yra tikslesni, negu dalijimo pusiau metodu gautas rezultatas.