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

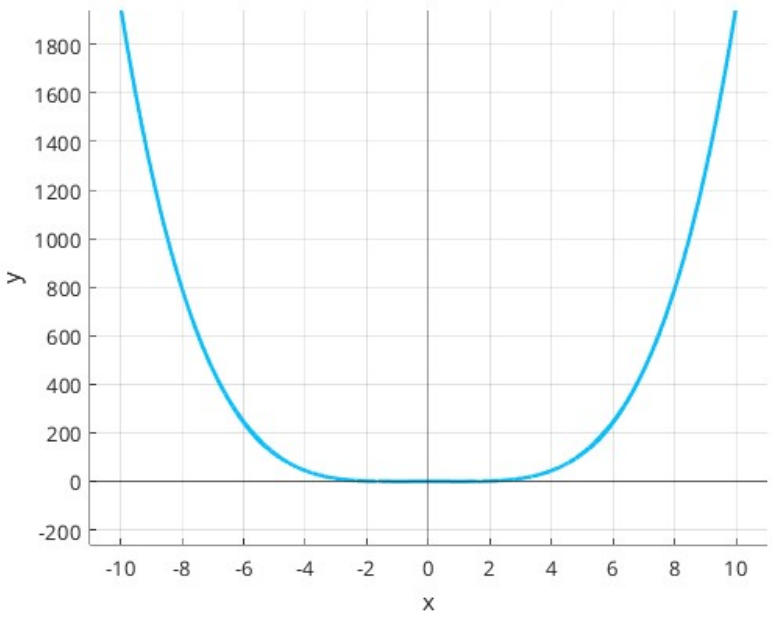
Šiame laboratoriniame darbe yra analizuojami trys funkcijos minimizavimo metodai: dalijimo pusiau metodas, auksinio pjūvio metodas, Niutono metodas. Šie metodai yra tiriami, juos pritaikant funkcijai

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

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 pagal tai, kiek jiems prireikė iteracijų bei funkcijų skaičiavimų atlikti, kad būtų gautas norimo tikslumo rezultatas.

Metodai buvo įgyvendinti Matlab programavimo kalba.

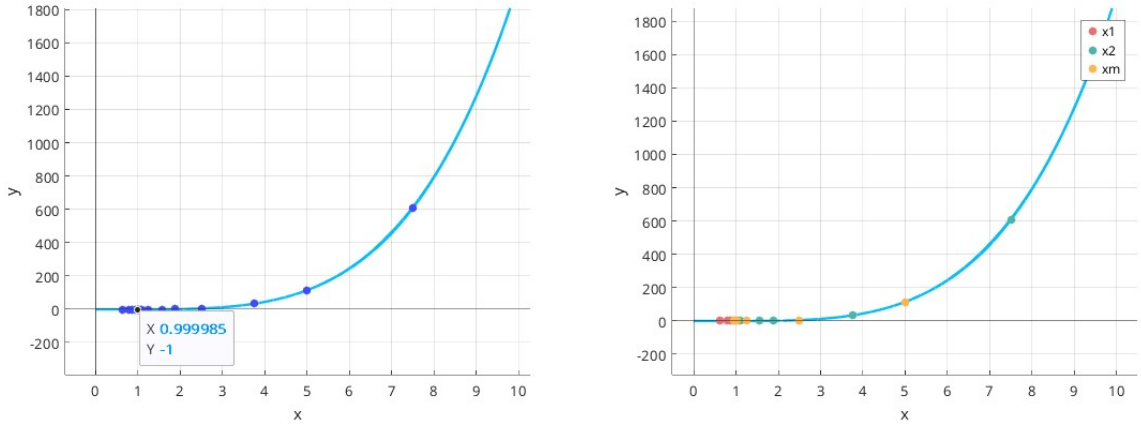
1. Dalijimo pusiau metodas

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 $x_m = 0.99998$. 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 35 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.

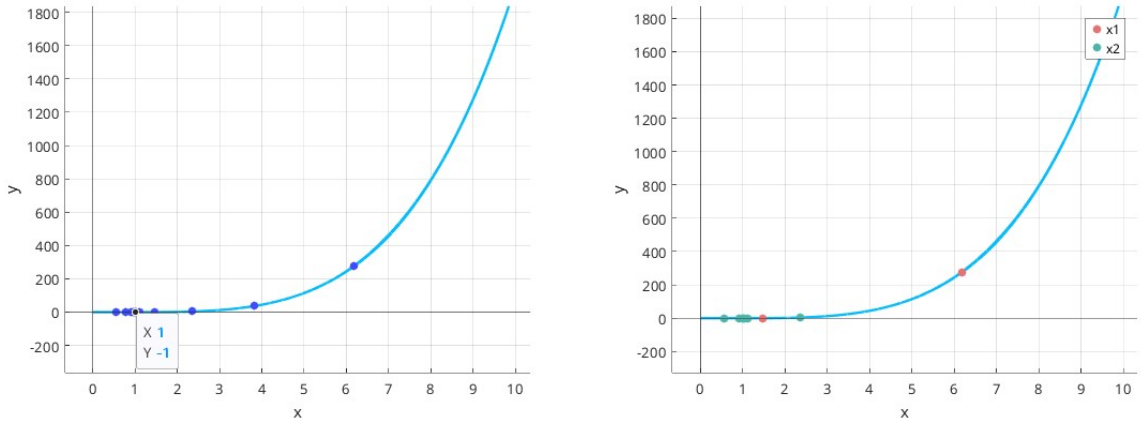
Algorithm 2 Auksinio pjūvio metodas funkcijai $f(x)$

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 - 24 iteracijų. Galutinis rezultatas skaičiuojamas naudojant vidurio taško formulę. Šiuo atveju tai būtų

$$\frac{x_1 + x_2}{2} \approx \frac{1.0000228882 + 0.9999847412}{2} \approx 1.0000038146 \tag{3}$$

Čia x_1 ir x_2 yra paskutinės iteracijos atitinkamos reikšmės. 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 26 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

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

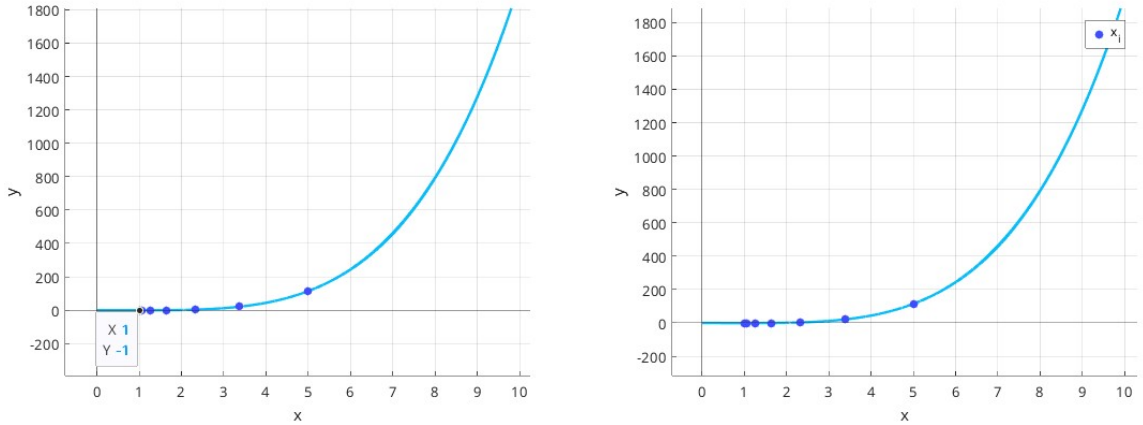
Šio metodo kontekste funkcijos iškvietimu laikysime išvestinių funkcijų iškvietimus.

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 8 iteracijų. Galutinis rezultatas yra paskutinis suskaičiuotas x_{i+1}. 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 funkcijos skaičiavimų.

Išvados

Žemiau pateiktoje lentelėje [1] galima pamatyti, kaip skiriasi metodų efektyvumas, bei jų gauti rezultatai.

Metodas	Dalijimo pusiau	Auksinio pjūvio	Niutono
Iteracijų sk.	17	24	8
Funkcijų skaičiavimo sk.	35	26	16
Gautas minimumo taškas	0.9999847412	1.0000038146	1.0000000018
Gautas minimumas	-0.9999999998	-0.9999999999	-1.0000000000

1 lentelė. Metodų palyginimo lentelė

Lyginant metodus pagal iteracijų skaičių, su duotos funkcijos minimumo radimu geriausiai susitvarko Niutono metodas. Dalijimo pusiau metodas yra antras pagal efektyvumą, atlikdamas darbą apytiksliai 2 kartus ilgiau. Paskutinėje vietoje pagal iteracijų skaičių yra auksinio pjūvio metodas, kuriam prireikia 3 kartus daugiau iteracijų negu Niutono metodui.

Lyginant metodus pagal funkcijų skaičiavimo kiekį, situacija yra kiek kitokia. Pirmoje vietoje taip pat yra Niutono metodas, tačiau vėlesnėse vietose rezultatai skiriasi nuo aukščiau aptartų. Antroje vietoje pagal funkcijų skaičiavimo kiekį yra auksinio pjūvio metodas, o trečioje - dalijimo pusiau metodas.

Kadangi funkcijų efektyvumas yra vertinamas pagal tikslo funkcijos iškvietimų skaičių, galima būtų teigti, kad Niutono metodas vienareikšmiškai yra efektyvesnis už kitus. Tačiau prisiminus formulę, naudojamą Niutono metode, matome, kad kviečiama ne pati tikslo funkcija, o jos pirmos ir antros eilės išvestinės. Taigi vienareikšmiškai vertinti šių metodų negalima, bet tarus, kad išvestinių skaičiavimas užima tiek pat laiko ir resursų, kiek ir tikslo funkcijos skaičiavimas, matome, kad šiai funkcijai skaičiuoti efektyviausias yra Niutono metodas.

Teigti, kad kažkuris metodas apskritai yra geresnis už kitus, negalima, nes metodai šio laboratorinio ribose metodai buvo pritaikyti tik vienai funkcijai ir rezultatai gali skirtis pritaikius kitoms.

Taip pat vertėtų atkreipti dėmesį ir į funkcijų tikslumą. Dalijimo pusiau ir auksinio pjūvio metodams buvo pateikta sąlyga, kad metodas turi dirbti iki tol, kol intervalo dydis yra didesnis 10^{-4}. Niutono metodui sąlyga buvo veikti iki tol, kol žingsnio skirtumas nebus mažesnis už 10^{-4}. Matome, kad Niutono metodu gauti rezultatai yra artimiausi teoriškai apskaičiuotam rezultatui. Tolimiausi rezultatai, tuo tarpu, yra gaunami panaudojus dalijimo pusiau metodą.

Apibendrinus, galima teigti, kad atsižvelgus į tam tikras prielaidas, efektyviausias metodas pateiktai funkcijai minimizuoti yra Niutono metodas - tiek iteracijų, tiek funkcijų skaičiavimo kiekio atžvilgiais.