Szymon Ociepka *Data wykonania: 05.06.2024*

Metoda Złotego Podziału

oraz Ciąg Fibonacciego

Optymalizacja

Metody numeryczne

1. Wprowadzenie teoretyczne

1.1 Metoda złotego podziału

Metoda złotego podziału to technika optymalizacji jednowymiarowej, która jest często stosowana w algorytmach numerycznych. Jest to metoda bezgradientowa, co oznacza, że nie wymaga obliczania pochodnych funkcji.

Podstawą metody złotego podziału jest złoty podział, który jest unikalnym podziałem linii na dwie nierówne części, takie że stosunek długości całej linii (a+b) do dłuższej części (a) jest taki sam jak stosunek długości dłuższej części (a) do krótszej (b). Ten stosunek jest nazywany złotym podziałem i wynosi około 0.618033988749895.

$$\frac{a+b}{a}=\frac{a}{b}\simeq\varphi \qquad (1)$$

Cel optymalizacji jednowymiarowej:

- funkcja jest ciągła w przedziale [a, b],
- posiada co najwyżej jedno ekstremum w przedziale [a, b]

Głównym celem optymalizacji jednowymiarowej funkcji jest identyfikacja jej **minimum** w przedziale [a, b].

W kontekście optymalizacji, metoda złotego podziału polega na wyborze punktu podziału w taki sposób, aby podzielić obecny przedział niepewności na dwa podprzedziały, które są w złotym podziale. Następnie, funkcja celu jest obliczana w nowym punkcie podziału, a jeden z końców przedziału jest zastępowany nowym punktem podziału, tak aby utrzymać złoty podział. Proces ten jest powtarzany, aż przedział niepewności zostanie zredukowany do pożądanego poziomu.

1. Obliczamy wartości funkcji w dwóch punktach x_l, x_p takich, że $a<x_l<x_p<b$.
2. Jeżeli $f(x_l)>f(x_p)$, to szukane minimum znajduje się w przedziale $[x_l, b]$. (2)
3. Jeżeli $f(x_l)<f(x_p)$, to szukane minimum znajduje się w przedziale $[a, x_p]$. (3)
4. Zawężamy przedział w którym znajduje się minimum, aż do momentu gdy spełniony zostanie warunek: $(b-a)\leq\varepsilon$, gdzie ε jest przyjętą dokładnością obliczeń.

Złota liczba

$\varphi\simeq 0.618033988749895$

$$x_l=b-\varphi(b-a) \qquad (4)$$

$$x_p=b+\varphi(b-a) \qquad (5)$$

1.2 Ciąg Fibonacciego

Ciąg Fibonacciego to jeden z najbardziej znanych ciągów w matematyce, który został nazwany na cześć włoskiego matematyka Leonarda z Pizy, znanego również jako Fibonacci. Ciąg ten jest zdefiniowany rekurencyjnie, co oznacza, że każda kolejna liczba w ciągu jest sumą dwóch poprzednich liczb. Matematycznie, ciąg Fibonacciego jest zdefiniowany w następujący sposób:

$$f(n)=f(n-1)+f(n-2) \tag{6}$$

gdzie:

$f(0)=0$ oraz $f(1)=1$

Pierwsze kilka liczb w ciągu Fibonacciego to: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …

Ciąg Fibonacciego ma wiele zastosowań i pojawia się w wielu różnych dziedzinach, od biologii po informatykę. Na przykład, liczby Fibonacciego są obecne w naturze w takich strukturach jak układ liści na drzewie, układ nasion na kwiatostanie słonecznika, czy struktura muszli ślimaka. W informatyce, ciąg Fibonacciego jest często używany w algorytmach, takich jak algorytmy sortowania i wyszukiwania.

Ciąg Fibonacciego jest również interesujący z punktu widzenia teorii liczb. Na przykład, stosunek dwóch kolejnych liczb Fibonacciego dąży do złotego podziału, który jest irracjonalną liczbą często spotykaną w sztuce i architekturze.

Wreszcie, ciąg Fibonacciego jest przykładem ciągu, który jest zdefiniowany rekurencyjnie. Oznacza to, że aby obliczyć daną liczbę w ciągu, musimy najpierw obliczyć poprzednie liczby. To jest podstawowy koncept w informatyce i matematyce, który jest używany w wielu różnych kontekstach.

2. Opis implementacji numerycznej

2.1 Metoda złotego podziału

```
double f(double x) {
    return pow(x, 3) + pow(x, 2);
}
```

Fragment kodu 2.1: funkcja dla której obliczane jest minimum w danym przedziale

Funkcja ta musi spełniać założenia przytoczone we wstępie teoretycznym: funkcja jest ciągła w przedziale [a, b], posiada co najwyżej jedno ekstremum w przedziale [a, b].

```
while (b - a >= epsi) {
    if (f(c) < f(d)) {
        b = d;
    }
    else {
        a = c;
    }

    // Obliczamy nowe c i d
    c = b - phi * (b - a);
    d = a + phi * (b - a);
}
```

Fragment kodu 2.2: pętla w której obliczane jest minimum funkcji

Zmienne **a** i **b**, reprezentują początkowy zakres, w którym szukane jest minimum. Zmienna **epsi** to tolerancja błędu, która określa precyzję wyniku. Zmienna **phi** to wartość złotego podziału, która jest używana do wyznaczania nowych punktów **c** (x_l) i **d** (x_p).

Na początku obliczane są pierwsze dwa punkty podziału, **c** i **d** używając proporcji złotego podziału.

Następnie korzystamy z **pętli while**, która trwa, dopóki różnica między **a** i **b** jest większa lub równa **epsi**. W każdym obrocie pętli, funkcja **f** jest wywoływana dla punktów **c** i **d**. Jeżeli wartość **f(c)** jest mniejsza od **f(d)**, wzór (3), to **b** jest zastępowane przez **d**. W przeciwnym razie **a** jest zastępowane przez **c**.

Po każdym obrocie pętli, obliczane są nowe punkty podziału **c** i **d**.

Po zakończeniu pętli, funkcja wyświetla średnią z **a** i **b**, która jest przybliżonym miejscem minimum funkcji.

2.2 Ciąg Fibonacciego

```
fib[0] = 1;
fib[1] = 1;

for (int i = 2; i < n; i++){
    fib[i] = fib[i - 1] + fib[i - 2];
}
```

Fragment kodu 2.3: obliczanie wyrazów ciągu Fibonacciego

Zmienna **fib** jest to zadeklarowana tablica typu long double, która przechowuje wyrazy ciągu Fibonacciego. Zmienna **n** jest to liczba indeksu wyrazu ciągu ponadawana przez użytkownika programu. Następnie, dla każdej liczby od **2** do **n-1**, program oblicza kolejny wyraz ciągu jako sumę dwóch poprzednich wyrazów za pomocą pętli **for** i **wzoru (6)**.

3. Testy kodu numerycznego

3.1 Metoda złotego podziału

Test I

Dane:

$$f(x)=\log(\sin(x)+\cos(x)+2) x\in[2,6]$$

Wynik Wolfram Alpha:

x ≈ *3.9270*

Wynik zaimplementowanego programu:

```
Minimum funkcji wynosi: 3.92699
```

Fragment konsoli 3.1: wynik programu dla testu I

Test II

Dane:

$$f(x)=(x^3-2\cdot x+1)\cdot\cos(x^2)+\sin(x^3+3\cdot x), \quad x\in[-1.8,-1.2]$$

Wynik Wolfram Alpha:

x ≈ *-1.48598*

Wynik zaimplementowanego programu:

```
Minimum funkcji wynosi: -1.48598
```

Fragment konsoli 3.2: wynik programu dla testu II

Test III

Dane:

$$f(x)=e^{\tan(x)}+\sin(e^x), \quad x\in[2.2,2.5]$$

Wynik Wolfram Alpha:

x ≈ *2.39134*

Wynik zaimplementowanego programu:

```
Minimum funkcji wynosi: 2.39134
```

Fragment konsoli 3.3: wynik programu dla testu III

Test IV

Dane:

$$f(x)=\frac{e^{x^2}}{\log(x+2)}, \quad x\in[0.1,1.6]$$

Wynik Wolfram Alpha:

x ≈ *0.2689605*

Wynik zaimplementowanego programu:

```
Minimum funkcji wynosi: 0.268959
```

Fragment konsoli 3.4: wynik programu dla testu IV

3.2 Ciąg Fibonacciego

Test I

wartość dla **66** wyrazu ciągu

Oczekiwana wartość:

27 777 890 035 288

Wynik zaimplementowanego programu:

```
wyraz nr 66 = 27777890035288
```

Fragment konsoli 3.5: wynik programu dla testu I

Test II

wartość dla **125** wyrazu ciągu

Oczekiwana wartość:

59 425 114 757 512 643 212 875 125

Wynik zaimplementowanego programu:

```
wyraz nr 125 = 59425114757512653803880448
```

Fragment konsoli 3.6: wynik programu dla testu II

Test III

wartość dla **273** wyrazu ciągu

Oczekiwana wartość:

505 988 662 735 923 140
767 969 869 749 836 918
999 964 413 630 219 877
218

Wynik zaimplementowanego programu:

```
wyraz nr 273 = 505988662735923004528643092371885
557133246204936565293056
```

Fragment konsoli 3.7: wynik programu dla testu III

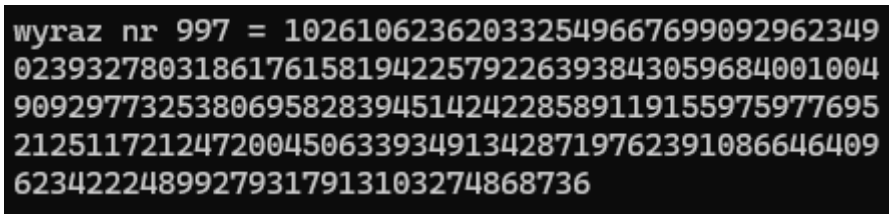
Test IV

wartość dla **997** wyrazu ciągu

Oczekiwana wartość:

10 261 062 362 033 262 336 604
926 729 245 222 132 668 558
120 602 124 277 764 622 905
699 407 982 546 711 488 272
859 468 887 457 959 087 733
119 242 564 077 850 743 657
661 180 827 326 798 539 177
758 919 828 135 114 407 499
369 796 465 649 524 266 755
391 104 990 099 120 377

Wynik zaimplementowanego programu:



Fragment konsoli 3.8: wynik programu dla testu IV

4. Analiza wyników

W celu głębszej analizy powstała tabela 1, w której porównane zostały wyniki zaimplementowanych metod z wynikami programu „Wolfram Alpha" na podstawie powyższych testów. W tabeli 2 porównane zostały wartości wyrazów Fibonacciego z Wikipedii z wynikami obliczonymi za pomocą zaimplementowanego programu.

Tabela 1: wartości x minimum funkcji dla Testów I-IV dla Złotego Podziału

Testy Złotego Podziału	**Wolfram Alpha**	**Program**
I	3.927	3.92699
II	-1.48598	-1.48598
III	2.39134	2.39134
IV	0.2689605	0.268959

Tabela 2: wartości danych wyrazów ciągu Fibonacciego

Numer Wyrazu Fibonacciego	**Wikipedia**	**Program**
66	27 777 890 035 288	27 777 890 035 288
125	59 425 114 757 512 643 212 875 125	59 425 114 757 512 653 803 880 448
273	505 988 662 735 923 140 767 969 869 749 836 918 999 964 413 630 219 877 218	505 988 662 735 923 004 528 643 092 371 885 557 133 246 204 936 565 293 056

997	10 261 062 362 033 262 336 604 926 729 245 222 132 668 558 120 602 124 277 764 622 905 699 407 982 546 711 488 272 859 468 887 457 959 087 733 119 242 564 077 850 743 657 661 180 827 326 798 539 177 758 919 828 135 114 407 499 369 796 465 649 524 266 755 391 104 990 099 120 377	10 261 062 362 033 254 966 769 909 296 234 902 393 278 031 861 761 581 942 257 922 639 384 305 968 400 100 490 929 773 253 806 958 283 945 142 422 858 911 915 597 597 769 521 251 172 124 720 045 063 393 491 342 871 976 239 108 664 640 962 342 224 899 279 317 913 103 274 868 736

Dzięki analizie wyników można stwierdzić, że przedstawiona metoda została zaimplementowana poprawnie. Wyniki uzyskane za pomocą zaimplementowanego programu są praktycznie identyczne względem wyników uzyskanych za pomocą programu Wolfram Alpha. Jednakże, występuje parę przypadków gdzie zauważono minimalne rozbieżności, które mogą wynikać z precyzji elementów - liczby zmiennoprzecinkowe o dużej liczbie miejsc po przecinku, zainicjowana stała Eulera używana w programie.
Mogą wystąpić błędy zaokrąglenia wynikające z ograniczonej precyzji liczb zmiennoprzecinkowych. Te błędy mogą się kumulować w trakcie obliczeń, prowadząc do zaokrąglania wyników. Wartości w drugiej tablicy dla małych elementów ciągu Fibonacciego są identyczne. Natomiast dla większych elementów ciągu wartości się różnią końcowymi cyframi w danej liczbie. Wynika to przede wszystkim stąd, iż tablica fib, z której korzystamy podczas implementacji programu jest typu long double, co pozwala na przechowywanie bardzo dużych liczb z dużą precyzją. Jednakże, ze względu na ograniczenia pamięci, program może nie być w stanie poprawnie obliczyć wyrazów ciągu dla bardzo dużych wartości n.

5. Wnioski

W tym sprawozdaniu przedstawiono oraz omówiono metodę *Złotego Podziału.* Pokazano oraz wytłumaczono jej prostą implementację w języku c++. Z analizy przeprowadzonych testów wynika że metoda *Złotego Podziału* jest skutecznym narzędziem do znajdowania ekstremum. Testy wykazały, że zastosowana implementacja jest szczególnie efektywna pod względem uzyskanych wyników, natomiast mogą wystąpić minimalne błędy zaokrąglenia wynikające z ograniczonej precyzji liczb zmiennoprzecinkowych.

Metoda złotego podziału jest atrakcyjna ze względu na swoją prostotę i efektywność, ale ma też pewne ograniczenia. Na przykład, jest mniej efektywna dla funkcji, które są silnie zakrzywione lub mają wiele lokalnych ekstremów. Ponadto, jako metoda bezgradientowa, może być mniej efektywna niż metody gradientowe dla funkcji, które są dobrze opisane przez swoje gradienty.

Omówiony został również ciąg Fibonacciego oraz jego prosta implementacja. Wyniki pokazały że dla małych wartości program radzi sobie bezproblemowo, jednakże istnieją pewne ograniczenia. Ze względu na ograniczenia pamięci, program może nie być w stanie

poprawnie obliczyć wyrazów ciągu dla bardzo dużych wartości n, co pokazują zrobione testy. Podsumowując, ciąg Fibonacciego to potężne narzędzie, które ma wiele zastosowań w różnych dziedzinach. Jego prostota i elegancja czynią go idealnym tematem do nauki zarówno dla początkujących, jak i doświadczonych matematyków i programistów.

6. Źródła

- prezentacja z zajęć „lab 13.pdf”
- Beata Pańczyk, Edyta Łukasik, Jan Sikora, Teresa Guziak, Metody numeryczne w przykładach, Politechnika Lubelska, 2012
- www.wolframalpha.com
- pl.wikibooks.org/wiki/Ci%C4%85g_Fibonacciego