

# Sprawozdanie

Bartosz Zasieczny

17 listopada 2013

## Spis treści

<b>1</b>	<b>Zadanie</b>	<b>1</b>
<b>2</b>	<b>Aparat matematyczny</b>	<b>1</b>
2.1	Metoda Newtona . . . . .	1
2.1.1	Uwagi . . . . .	2
2.2	Metoda bisekcji . . . . .	2
2.3	<i>Regula falsi</i> . . . . .	2
2.3.1	Wzory . . . . .	3
2.4	Metoda <i>złotego podziału</i> . . . . .	3
2.4.1	Algorytm . . . . .	3
<b>3</b>	<b>Badanie funkcji</b>	<b>4</b>
3.1	Funkcja 0 . . . . .	4
<b>4</b>	<b>Kompilacja i obsługa programu</b>	<b>4</b>
4.1	Wymagania . . . . .	4
4.2	Kompilacja . . . . .	5
4.3	Obsługa programu . . . . .	5

## 1 Zadanie

Korzystając z omówionych na wykładzie iteracyjnych metod aproksymacji pierwiastków, zaproponować sposób wyznaczania *ekstremum lokalnego* funkcji  $f \in C^1[a, b]$ . Wykonać eksperymenty m. in. dla:

0.  $f(x) = \sin(2\pi x)$ ,  $x \in [0, 1]$ ;
1.  $f(x) = e^{-x^2}$ ,  $x \in [-1, 1]$ ;
2.  $f(x) = \frac{x}{1+x^2}$ ,  $x \in [0, 10]$ ;
3.  $f(x) = x^2 + x - 1$ ,  $x \in [-1, 2]$ .

## 2 Aparat matematyczny

W poszukiwaniu ekstremów funkcji będziemy używać poniższych metod. Niektóre z nich pozwalają na znalezienie ekstremum wprost, inne będą skupiać się na poszukiwaniu miejsca zerowego pierwszej pochodnej funkcji tam gdzie to możliwe.

### 2.1 Metoda Newtona

**Metoda Newtona** polega na iteracyjnym wyznaczaniu kolejnych przybliżeń pierwiastka  $f(x)$  poprzez:

- znalezienie stycznej do jej wykresu w punkcie  $x_i$  (zaczynając od punktu startowego  $x_0$ );
- biorąc wartość dziedziny w punkcie przecięcia stycznej z osią  $X$  za  $i + 1$ -sze przybliżenie pierwiastka (czyli  $x_{i+1}$ ).

Kroki powtarzamy aż do otrzymania wymaganej precyzji. Kolejne przybliżenia  $x_{i+1}$  wyznaczamy za pomocą wzoru:

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

#### 2.1.1 Uwagi

- Charakterystyka tego zadania uniemożliwia użycie samej metody Newtona - dla pewnych danych może ona wskazać przybliżenia pierwiastka  $f(x)$  spoza pożądanego przedziału. Problemem też jest dobór odpowiedniego punktu startowego - dlatego w przypadku tego zadania należy stosować tę metodę tylko po wstępnym przybliżeniu pierwiastka funkcji przez inne metody iteracyjne.
- W przypadku tego zadania każda badana funkcja musi posiadać co najmniej dwie pochodne.

### 2.2 Metoda bisekcji

Dla funkcji  $f(x)$  ciągłej w przedziale  $[a, b]$  i przyjmującej na jego końcach wartości o różnych znakach ( $f(a)f(b) < 0$ ) należy wykonać następujące kroki:

1. sprawdzić, czy środek przedziału jest pierwiastkiem funkcji (sprawdzić czy  $f(x)$  dla wartości dziedziny  $x_0 = \frac{a+b}{2}$  ma wartość  $f(x_0) = 0$ ;
2. jeśli tak, to zakończyć algorytm i zwrócić  $x_0$ ;
3. w p. p. sprawdzić który z przedziałów ( $[a, x_0]$  czy  $[x_0, b]$ ) spełnia własność  $f(a')f(b') < 0$  i zastosować do niego pierwszy krok algorytmu.

## 2.3 Regula falsi

Metoda *falszywej prostej* wyznacza przybliżenia pierwiastka  $f(x)$  spełniającej następujące założenia w przedziale  $[a, b]$ :

- $f(x)$  jest ciągła w przedziale  $[a, b]$ ;
- $f(x)$  w przedziale  $[a, b]$  ma **dokładnie jeden** pierwiastek;
- $f(x)$  na końcach przedziału  $[a, b]$  przyjmuje różne znaki wartości ( $f(a)f(b) < 0$ );
- $\forall_{x \in [a, b]} \exists f'(x) \wedge \exists f''(x)$ ;
- $\forall_{x', x'' \in [a, b]} \operatorname{sgn} f'(x') = \operatorname{sgn} f'(x'') \wedge \operatorname{sgn} f''(x') = \operatorname{sgn} f''(x'')$ .

Aby wyznaczyć przybliżenie pierwiastka należy wykonać następujące kroki:

1. przez punkty  $A = (a, f(a))$  i  $B = (b, f(b))$  przeprowadzana jest prosta;
2. punkt przecięcia  $x_i$  osi  $X$  jest przybliżeniem pierwiastka;
3. jeśli precyzja przybliżenia jest zadowalająca to kończymy algorytm;
4. w p. p. wybierany jeden z przedziałów ( $[a, x_i]$  czy  $[x_i, b]$ ) taki, który spełnia własność  $f(a')f(b') < 0$  i stosujemy do niego pierwszy krok algorytmu.

### 2.3.1 Wzory

$$x_0 = \frac{af(b) - bf(a)}{f(b) - f(a)}$$
$$x_{i+1} = \begin{cases} \frac{x_i f(a) - af(x_i)}{f(a) - f(x_i)} & \text{gdy } f(a)f(x_i) \leq 0 \\ \frac{x_i f(b) - bf(x_i)}{f(b) - f(x_i)} & \text{gdy } f(b)f(x_i) < 0 \end{cases}$$

dla  $i = 1, 2, \dots$

## 2.4 Metoda złotego podziału

Ta metoda w odróżnieniu od poprzednich pozwala szukać lokalnego ekstremum wprost, bez konieczności odwoływania się do pochodnych danej funkcji i poszukiwania ich zer. Żeby funkcja  $f(x)$  mogła zostać zbadana za pomocą tej metody, musi być ona w przedziale  $[a, b]$ , w którym poszukujemy ekstremum, **unimodalna** – tzn. ciągła i posiadać w tym przedziale dokładnie jedno ekstremum.

### 2.4.1 Algorytm

Pierwszy krok algorytmu:

$$\begin{cases} x_L^{(0)} := b^{(0)} - (b^{(0)} - a^{(0)})k \\ x_R^{(0)} := a^{(0)} + (b^{(0)} - a^{(0)})k \end{cases}$$

Następnie iterujemy po przypadkach, aż do uzyskania zadowalającej precyzji:

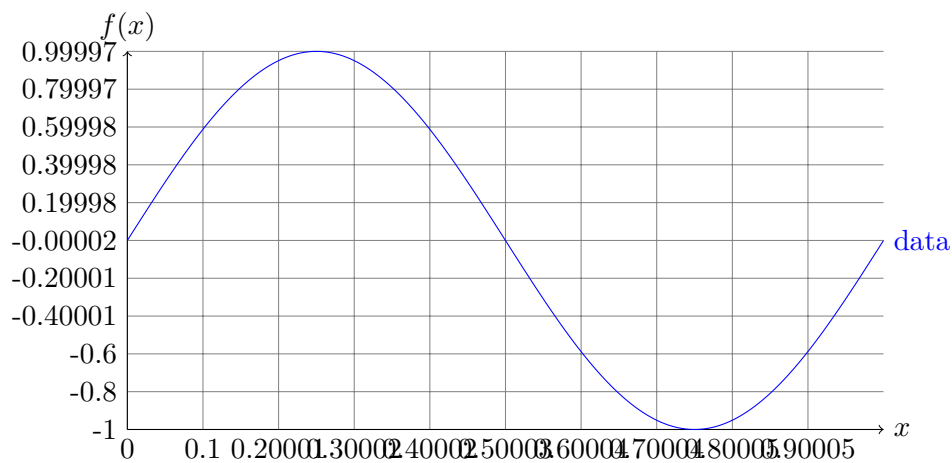
$$\begin{aligned} \bullet f(x_L^{(i)}) > f(x_R^{(i)}) &\Rightarrow \begin{cases} a^{(i+1)} := x_L^{(i)} \\ b^{(i+1)} := b^{(i)} \\ x_L^{(i+1)} := x_R^{(i)} \\ x_R^{(i+1)} := a^{(i+1)} + (b^{(i+1)} - a^{(i+1)})k \end{cases} \\ \bullet f(x_L^{(i)}) < f(x_R^{(i)}) &\Rightarrow \begin{cases} a^{(i+1)} := a^{(i)} \\ b^{(i+1)} := x_R^{(i)} \\ x_L^{(i+1)} := b^{(i+1)} - (b^{(i+1)} - a^{(i+1)})k \\ x_R^{(i+1)} := x_L^{(i)} \end{cases} \end{aligned}$$

Po zakończeniu iteracji, środek przedziału  $[a, b]$  jest brany jako przybliżenie lokalnego ekstremum funkcji.

## 3 Badanie funkcji

### 3.1 Funkcja 0

Wzór funkcji, pochodnych i badany przedział:



$$f(x) = \sin(2\pi x), x \in [0, 1]$$

$$f^{(1)}(x) = 2\pi \cdot \cos(2\pi x)$$

$$f^{(2)}(x) = -4\pi^2 \cdot \sin(2\pi x)$$

$$f^{(3)}(x) = -8\pi^3 \cdot \cos(2\pi x)$$

## 4 Kompilacja i obsługa programu

### 4.1 Wymagania

Aby skompilować program należy spełnić następujące wymagania dotyczące oprogramowania:

- kompilator *G++* w wersji 4.7 lub późniejszej - kompilator musi obsługiwać standard *C++11*,
- obecność narzędzia GNU Make

Powyższe wymagania powinny być automatycznie spełnione w każdej aktualnej dystrybucji GNU/Linux.

### 4.2 Kompilacja

Należy przejść do katalogu `prog` i wykonać polecenie `make` - kompilacja wykona się automatycznie. W pliku `Makefile` podane są polecenia, które należy wykonać aby skompilować program ręcznie.

### 4.3 Obsługa programu

Program uruchamiamy za pomocą pliku `main`, po jego nazwie podając ciąg będący kombinacją poniższych parametrów:

- `-f <nr_funkcji>` – za pomocą tego argumentu wybieramy jedną z dostępnych funkcji - liczba przyporządkowana funkcji to jej liczba porządkowa z treści zadania :3, dodanie 1 to pierwsza pochodna, dodanie 2 to druga pochodna,
- `-d <nr_funkcji>` – podobnie jak powyżej, tyle, że podajemy liczbę pochodnej,
- `-m <metoda>` – wybór jednej z metod:
  - `newton` – metoda newtona (obowiązkowe parametry wywołania to `-f -d -x`)
  - `regula_falsi` – *regula falsi* (obowiązkowe parametry to `-f -s`)
  - `bisection` – bisekcja (obowiązkowe parametry to `-f -s`)
  - `golden_section` – metoda *złotego podziału* (obowiązkowe parametry to `-f -s`)

– `plot` – "wykres" funkcji (punkty) (obowiązkowe parametry to `-f`  
`-s -step`)

- `-p <n>` – wypisz wyniki z precyzją  $n$  cyfr po przecinku (domyślnie 20),
- `-s <a> <b>` – określ badany przedział od  $a$  do  $b$ ,
- `-x <y>` –  $y$  jako punkt startowy,
- `-e (min|max)` – określ czy szukać lokalnego minimum czy maximum w przedziale (działa tylko z `-m golden_section`, domyślnie `min`),
- `-error <e>` – określ tolerancję błędu (domyślnie  $10^{-10}$ ),
- `-step <s>` – wielkość kroku przy obliczaniu punktów wykresu (działa tylko z `-m plot`, domyślnie 0.1),
- `-i <i>` – ilość iteracji (domyślnie 20),

Przykład: szukamy lokalnego minimum dla pierwszej funkcji z zadania, w podanym przedziale, za pomocą metody *złotego podziału*, z tolerancją błędu na poziomie  $10^{-12}$ , maksymalnie 30 iteracjami i precyzją 25 liczb po przecinku.

```
./main -f 0 -s 0 1 -m golden_section -e 10e-12 -i 30 -p 25
```