Dokumentacja do zadania 2.5

Marcin Horoszko, Radosław Głombiowski, Jacek Dermont

25 listopada 2012

1 Zadanie 2.5

Dla równania $f(x) = 0$, gdzie $f(x) = lnx + x - 5$, wczytać a,b $\in R$ takie, by $0 <$ a $<$ b oraz $f(a) \cdot f(b) < 0$. Następnie, dopóki "użytkownik się nie znudzi", wczytywać wartość $0 < \epsilon < 1$ i metodą połowienia na [a,b] przybliżyć z dokładnością ϵ rozwiązanie tego równania. Rozwiązanie to przybliżyć również metodą Newtona z $x_0 =$ a, przy czym x_k będzie dobrym przybliżeniem, gdy $|x_k - x_{k-1}| \leqslant \epsilon$. Porównać ilość kroków wykonanych metodą połowienia i metodą Newtona.

2 Podstawowe pojęcia

2.1 (Metoda połowienia (równego podziału, bisekcji))**.** Jedna z metod rozwiązywania równań nieliniowych. Opiera się ona na twierdzeniu Bolzano-Cauchy'ego: *Jeżeli funkcja ciągła $f(x)$ ma na końcach przedziału domkniętego wartości różnych znaków, to wewnątrz tego przedziału, istnieje co najmniej jeden pierwiastek równania $f(x) = 0$.*. Przebieg metody:
Wybieramy przybliżenie ϵ (np. $\epsilon = 0.001$).
1. Sprawdzamy czy dla $c = \frac{a+b}{2}$ $f(c) = 0$. Jeśli tak, to kończymy, a c jest pierwiastkiem funkcji.
2. Jeżeli nie, dzielimy przedział na połowy $[a, c][c, b]$. Jeżeli $f(a) \cdot f(c) < 0$ to $b = c$, w przeciwnym przypadku $a = c$.
3. Powtarzamy proces, dopóki odległość między a i b nie będzie mniejsza lub równa przybliżeniu ϵ. Zarówno a i b będą pierwiastkami spełniającymi zadane przybliżenie.

2.2 (Metoda Newtona)**.** Iteracyjny algorytm wyznaczania przybliżonej wartości pierwiastka funkcji. Można ją zastosować, gdy w przedziale $[a, b]$ funkcja jest ciągła, $f(a) \cdot f(b) < 0$ oraz pierwsza i druga pochodna funkcji mają stały znak na tym przedziale. Przebieg metody:
Wybieramy przybliżenie ϵ (np. $\epsilon = 0.001$) oraz wyliczamy pochodną $f'(x)$.
1. $x_0 = a$
2. Kolejny k-ty x wyliczamy ze wzoru: $x_{k+1} = x_k$-$\frac{f(x_k)}{f'(x_k)}$.
3. Powtarzamy 2. punkt dopóki $|x_{k+1} - x_k| \leqslant \epsilon$. x_{k+1} jest szukanym pierwiastkiem spełniającym zadane przybliżenie.

Przy spełnionych założeniach błąd przybliżenia maleje kwadratowo. Jednakże, jeśli punkt startowy jest daleko od pierwiastka, metoda Newtona może być rozbieżna i w konsekwencji działać w “nieskończoność”. W tym wypadku należy się zabepieczyć, np. poprzez ustalenie maksymalnej ilości iteracji.

3 Metoda numeryczna

Posiadając a, b (0<a<b i $f(a) \cdot f(b)$ <0) oraz ϵ (0< ϵ <1) możemy przybliżyć wartość pierwiastka $f(x) = lnx + x - 5$ powyższymi metodami.
Dla metody połowienia*:

- $i = 0$ (ilość kroków)
- tworzymy pętlę **while**, której warunkiem działania jest $|a - b| > \epsilon$:
  - $i + +$
  - c = $\frac{a+b}{2}$
  - jeśli $f(a) \cdot f(c) < 0$, to b = c
  - w przeciwnym przypadku a = c
- zmienna c to zadane przybliżenie
- zmienna i to ilość wykonanych kroków

* ponieważ dla równania w zadaniu x jest niewymierne, omijamy sprawdzanie $f(c) == 0$

Dla metody Newtona:

- definiujemy pochodną $f'(x) = \frac{1}{x} + 1$
- $j = 0$ (ilość kroków)
- x_0 = a
- tworzymy nie kończącą się pętlę **while**:
  - $j + +$
  - $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$
  - jeśli $|x_0 - x_1| \leqslant \epsilon$, to przerywamy **pętlę**
  - jeśli $j == 100$, to przerywamy **pętlę**
  - $x_0 = x_1$
- jeśli $j < 100$, zmienna x_1 jest naszym szukanym przybliżeniem
- jeśli $j == 100$, metoda Newtona jest w tym przypadku rozbieżna; nie otrzymaliśmy założonego przybliżenia
- zmienna j to ilość wykonanych kroków

4 Opis programu

Program został w całości napisany w języku *C*.

Wejście

- a, liczba typu double, taka że $a > 0$; jeśli $a \leqslant 0$, to program zapyta ponownie
- b, liczba typu double, taka że $b > a$; jeśli $b \leqslant a$, to program zapyta ponownie jeśli $f(a) \cdot f(b) \geqslant 0$, to program ponownie zapyta o a i b
- ϵ, liczba typu double, taka że $0 < \epsilon < 1$, podawana w nieskończonej pętli dopóki użytkownik "się nie znudzi"; jeśli ϵ jest spoza przedziału $[0, 1]$, program zapyta ponownie

Wyjście

- c, liczba typu double, pierwiastek funkcji o przybliżeniu ϵ
- i, liczba całkowita, ilość kroków wykonanych za pomocą metody połowień
- x_1, liczba typu double, pierwiastek funkcji o przybliżeniu ϵ *
- j, liczba całkowita, ilość kroków wykonanych za pomocą metody Newtona *

* jeśli $j == 100$ to x_1 nie jest pierwiastkiem, bo metoda Newtona stała się rozbieżna

Program składa się tylko z jednego pliku *main.c*. Zawiera wszystkie potrzebne funkcje.

5 Kod źródłowy

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double f(double x) {
    return log(x)+x-5;
}
double fp(double x) {
    return (1/x)+1;
}

int main() {
    double A,B,c,d,e,x0,x1,tmp=0;
    printf("--- Program rozwiazujacy rownanie ln(x) + x - 5 = 0.
       ---\n\n");
    printf("Podaj liczbe a i b, takie ze 0 < a < b \n");
    do{
```

```
        if(tmp == 1) printf("Nie ma rozwiazania. Podaj inne a i b.\n");
        do{
          printf("Podaj a: ");
          scanf("%lf",&A);
        }while(A <= 0);
        do{
          printf("Podaj b: ");
          scanf("%lf",&B);
        }while(B <= A);
        tmp = 1;
      } while(f(A) * f(B) >= 0);

      double a,b;
      int i,j;
      while(1) {
          printf("Podaj epsilon, takie ze 0 < epsilon < 1\n");
          do{
            printf("Podaj epsilon: ");
            scanf("%lf",&e);
          } while(e <= 0 || e >= 1);

          // metdoa polowien
          a = A;
          b = B;
          i = 0;
          while(fabs(a-b) > e) {
              c = (a+b)/2;
              if (f(a)*f(c) < 0) b = c;
              else a = c;
              i++;
          }
          printf("\nMetoda polowien:\n");
          printf("%.10lf\n",c);
          printf("Ilosc krokow: %i.\n",i);

          // metoda newtona
          x0 = A;
          j = 0;
          while (1) {
              x1 = x0-f(x0)/fp(x0);
              j++;
              if (fabs(x0-x1) <= e && fabs(c-x1) < 1) break;
              if (j == 100) break;
              x0 = x1;
          }
          printf("\nMetoda Newtona:\n");
          if (j == 100) printf("(wynik nieprawidlowy, metoda Newtona rozbiezna!) ");
          printf("%.10lf\n",x1);
          printf("Ilosc krokow: %i.\n\n",j);
      }
      return 0;
}
```

6 Przykładowe uruchomienia

Przykład 1

```
--- Program rozwiazujacy rownanie ln(x) + x - 5 = 0. ---

Podaj liczbe a i b, takie ze 0 < a < b
Podaj a: 1
Podaj b: 10
Podaj epsilon, takie ze 0 < epsilon < 1
Podaj epsilon: 0.1

Metoda polowien:
3.7421875000
Ilosc krokow: 7.

Metoda Newtona:
3.6934325794
Ilosc krokow: 3.

Podaj epsilon, takie ze 0 < epsilon < 1
Podaj epsilon: 0.001

Metoda polowien:
3.6932983398
Ilosc krokow: 14.

Metoda Newtona:
3.6934413590
Ilosc krokow: 4.
```

Przykład 2

```
--- Program rozwiazujacy rownanie ln(x) + x - 5 = 0. ---

Podaj liczbe a i b, takie ze 0 < a < b
Podaj a: 10
Podaj b: 20
Nie ma rozwiaznia. Podaj inne a i b.
Podaj a: 3
Podaj b: 5
Podaj epsilon, takie ze 0 < epsilon < 1
Podaj epsilon: 0.001

Metoda polowien:
3.6943359375
Ilosc krokow: 11.

Metoda Newtona:
3.6934413590
Ilosc krokow: 3.
```

Przykład 3

```
--- Program rozwiazujacy rownanie ln(x) + x - 5 = 0. ---

Podaj liczbe a i b, takie ze 0 < a < b
Podaj a: 0
Podaj a: -1
Podaj a: 1
Podaj b: 0
Podaj b: 4
Podaj epsilon, takie ze 0 < epsilon < 1
Podaj epsilon: 0.000001

Metoda polowien:
3.6934406757
Ilosc krokow: 22.

Metoda Newtona:
3.6934413590
Ilosc krokow: 5.

Podaj epsilon, takie ze 0 < epsilon < 1
Podaj epsilon: 0.5

Metoda polowien:
3.6250000000
Ilosc krokow: 3.

Metoda Newtona:
3.6934325794
Ilosc krokow: 3.

Podaj epsilon, takie ze 0 < epsilon < 1
Podaj epsilon: 20
Podaj epsilon: 0.9

Metoda polowien:
3.2500000000
Ilosc krokow: 2.

Metoda Newtona:
3.6760407835
Ilosc krokow: 2.
```