1 Zadanie numeryczne N12

Szymon Pach

1.1 Opis metody

W celu znalezienia miejsc zerowych funkcji $f(x)$ narysowałem wykres, żeby określić miejsca występowania rozwiązań. Przy okazji policzyłem, w których podprzedziałach funkcja zmienia znak i zapisałem te wartości w wektorze. Następnie z podanego wektora wykluczyłem wartości bliskie $\frac{\pi}{2}$, $\frac{\pi}{2} + \pi$, oraz $\frac{\pi}{2} + 2\pi$, ponieważ w tych punktach funkcja nie jest określona (zatem nie jest też ciągła). Gdy już znalazłem wszystkie przedziały, w których funkcja zmienia znak, dla kolejnych przedziałów zastosowałem metodę regula falsi, oraz metodę Newtona.

1.2 Instrukcja uruchomienia

Program został napisany w języku Matlab. W środowisku Matlab należy przejść w zakładce *Current folder* do folderu, w którym znajduje się program o nazwie *main.m*, a następnie wpisać w oknie konsoli *main*, a następnie nacisnąć *enter*.

1.3 Rozwiązanie

Rozwiązanie uzyskane dla metody regula falsi.

x_falsi =

0.045118456615408

0.470287289362595

2.494908078998843

2.839803686857662

3.340294407746180

5.551357798939004

5.658552147032624

6.015226649339037

6.242482691397051

6.474525432599741

6.764768145291455

6.895997112677795

8.642726724456946

8.668883517863213

8.946258386665553

9.068451122035310

9.268047949901503

9.426805237339343

9.582687785664779

9.771357386250596

9.885377897179474

Rozwiązanie uzyskane dla metody Newtona.

x_newton =

0.045118456615405

0.470287289362743

2.494908078998834

2.839803686857664

3.340294407746187

5.551357798938982

5.658552147032640

6.015226649339036

6.242482691397058

6.474525432599786

6.764768145291437

6.895997112677820

8.642726724456896

8.668883517863295

8.946258386665553

9.068451122035310

9.268047949901453

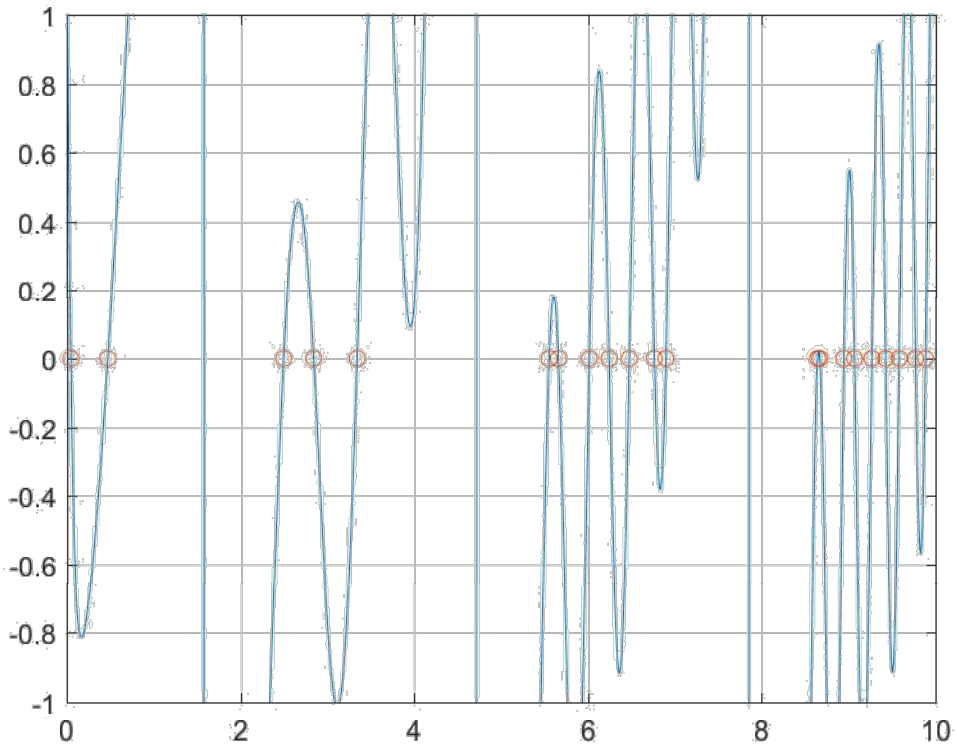
9.426805237339343

9.582687785664781

9.771357386250552

9.885377897179493

Wykres funkcji w przedziale z zaznaczonymi miejscami zerowymi.



1.4 Analiza błędów

Błędy uzyskane dla metody regula falsi.

errs_falsi =

1.0e-12 *

0.066918692809281
0.575317571360756
0.044519943287469
0.000499600361081
0.032446267894670
0.150102152929321
0.107136521876328
0.011435297153639
0.083363871361541
0.628747054420842
0.110023101740353
0.292765811593654

0.200284233642378

0.324740234702858

0.006106226635438

0.000444089209850

0.972166791513018

0.003789503433271

0.032807090377673

0.743405337289005

0.367039731941077

Liczba iteracji dla metody regula falsi.

iters_falsi =

10

4

6

4

5

6

6

6

4

5

7

7

11

8

8

6

5

3

6

5

7

Błędy uzyskane dla metody regula falsi.

errs_newton =

1.0e-13 *

0.004510281037540
0.034416913763380
0.004440892098501
0.099364960703952
0.258681964737661
0.006661338147751
0.034416913763380
0.031641356201817
0.018179902028237
0.031918911957973
0.781597009336110
0.017763568394003
0.032196467714130
0
0.061062266354384
0.004440892098501
0.090760732263107
0.037895034332713
0.111022302462516
0.147659662275146
0.095479180117763

Liczba iteracji dla metody regula falsi.

iters_newton =

4
2
3
3

2

3

4

3

3

3

3

4

4

6

4

4

3

3

3

3

4

Obliczenia wykonywałem w pakiecie 32-bitowym. Dla pewności znalazłem przybliżenia z dokładnością równą ok. 10^{-12}. Stosując metodę Newtona udało się znaleźć rozwiązania ok. 2 razy szybciej niż stosując metodę regula falsi.

1.5 Opis złożoności algorytmu

Metoda Newtona jest zbieżna kwadratowo, natomiast metoda regula falsi jest zbieżna liniowo, za to jest zbieżna globalnie w przeciwieństwie do metody Newtona.

1.6 Możliwa optymalizacja

Zastosowałem jedne z lepszych metod znajdowania miejsc zerowych dla podanych typów (metoda iteracyjna i metoda zawężająca przedział). Można by było ulepszyć metodę regula falsi do jej modyfikacji zwanej metodą Illinois. Szybszą zbieżność od metody Newtona można uzyskać np. stosując metodę Halley'a (złożoność kubiczna), jednak wadą tej metody jest to, że trzeba dokonać więcej obliczeń, oraz trzeba liczyć drugą pochodną funkcji.

1.7 Kod źródłowy

```
close all; clear; clc;
```

```
tol = 1e-12;
f = @(x) sin(x .^ 2 + log(x)) + tan(x);
df = @(x) cos(x .^ 2 + log(x)) .* (2 * x + 1 ./ x) + 1 ./ cos(x) .^ 2;

h = 0.01;
h2 = h / 2;
xx  = 0.01 : h : 10;
yy  = f(xx);
pp  = xx((yy(2:end) .* yy(1:(end-1))) < 0); n =
length(pp);

z = pi/2 : pi : 10;

for i = 1 : n
   if sum(abs(pp(i) - z) < h) > 0
      pp(i) = -1;
   end
end

pp  = pp(pp > 0); n
= length(pp);

x_falsi = zeros(n, 1);
errs_falsi = zeros(n, 1);
x_newton = zeros(n, 1);
errs_newton = zeros(n, 1);
iters_falsi = zeros(n, 1);
iters_newton = zeros(n, 1);

fig = figure;
plot(xx, yy);
grid on;
saveas(fig, 'fig12.png');
close(fig);
```

```
for i = 1 : n
    x = pp(i);
    a = x;
    b = x + h;
    fa = f(a);
    fb = f(b);

    % regula falsi for iter
    = 1 : 100
        c = (fa * b - fb * a) / (fa - fb); fc = f(c);



        if abs(fc) < tol
            break;
        end

        if fa * fc < 0
            b = c;
            fb = fc;
        else
            a = c;
            fa = fc;
        end
    end

    x_falsi(i) = c;
    errs_falsi(i) = abs(fc);
    iters_falsi(i) = iter;

    % metoda
    Newtona fx = f(x);
    for iter = 1 : 50
        x = x - fx / df(x); fx =
        f(x);
```

```
        if abs(fx) < tol
            break;
        end
    end

    x_newton(i) = x;
    errs_newton(i) = abs(fx);
    iters_newton(i) = iter;
end

display(x_falsi);
display(errs_falsi);
display(iters_falsi);

display(x_newton);
display(errs_newton);
display(iters_newton);
```