Metody numeryczne w technice

Laboratorium 1 – Rozwiązywanie równań nieliniowych

Funkcje wykorzystywane w ćwiczeniu:

`x = fzero(fcn, x0)` – wyznaczanie pierwiastka równania nieliniowego metodą Brendta.

`x = fsolve(fcn, x0)` – rozwiązywanie układu równań nieliniowych metodą Newtona.

`r = roots(p)` – wyznaczanie zer wielomianu.

`y = polyval(p, x)` – obliczenie wielomianu `p` dla wektora punktów `x`.

Metoda Banacha iteracji prostej

Krok 1. Dane jest równanie $f(x)=0$.

Krok 2. Przekształcamy równanie $f(x)=0$ na równanie postaci $x=g(x)$.

Krok 3. Definiujemy punkt startowy x_0.

Krok 4. Obliczamy kolejne przybliżenia $x_{i+1}=g(x_i)$ dla $i=1, 2, 3,...$

Krok 5. Przerywamy obliczenia jeżeli jest spełniony jeden z warunków:

A. Liczba iteracji osiągnie zadaną przez nas wartość.

B. Błąd przybliżenia jest mniejszy lub równy od zadanej tolerancji.

W tej metodzie miejscem zerowym jest punkt przecięcia funkcji $g(x)$ oraz funkcji $f(x)=x$.

Przebieg ćwiczenia

1. Wyznaczyć zera funkcji $f(x)=0.25x-\sin(x)$ z wykorzystaniem funkcji *fzero* i *fsolve.*
2. Wyznaczyć zera wielomianu $p(x)=x^4-3x^3+10x^2+6x-20$. Wykreślić jego wykres.
3. Napisać funkcję w Matlabie implementującą metodę Banacha iteracji prostej.
4. Z wykorzystaniem funkcji z punktu 3. zbadać zbieżność metody Banacha iteracji prostej dla funkcji $x^4-x-10=0$. Przekształć to równanie na postać $x=g(x)$ na kilka sposobów:

   a) $x=\dfrac{10}{x^3-1}$

   b) $x=(x+10)^{\frac{1}{4}}$

   c) $x=\dfrac{(x+10)^{\frac{1}{2}}}{x}$

5. Z wykorzystaniem funkcji z punktu 3. Zbadać zbieżność metody Banacha iteracji prostej dla funkcji $f(x)=0.25x-\sin(x)$.
6. Przetestuj działanie funkcji wbudowanych w Matlabie:
   - `eval`
   - `sign`
   - `errror`
   - `disp`

Kod do metody Banacha

1. funkcja

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Napisać funkcję implementującą metodę Banacha rozwiązującą iteracyjnie
% równanie postaci x = g(x) (założenie: jako parametr funkcja ma
% przyjmować uchwyt do funkcji g. Ponadto funkcja powinna sprawdzać
% zbieżność metody dla zadanej funkcji g oraz zadanego początkowego
% przybliżenia.) i porównać jej działanie z wbudowaną
% funkcją Octave - fzero.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% g - podawany na wejście uchwyt do funkcji
% NOfIter - liczba iteracji
% x0 - punkt startowy

function[wynik] = banach(g, NOfIter, x0)

wynik = zeros(NOfIter,1);
wynik(1) = x0;

for ii=2:NOfIter

    wynik(ii) = g(wynik(ii-1));

end
```

2. plik główny

```
clear all
close all
format longe

% METODA ITERACJI PROSTEJ BANACHA
%
% KROK 1. Dane jest równanie f(x) = 0.
% KROK 2. Przekształcamy równanie f(x) = 0 na równanie postaci x = g(x)
% KROK 3. Definiujemy punkt startowy x0.
% KROK 4. Obliczamy kolejne przybliżenia x(i+1) = g(x(i)) dla i = 1,2,3,...
% KROK 5. Przerywamy obliczenia jeżeli jest spełniony jeden z warunków:
%           A. Liczba iteracji osiągnie zadaną przez nas wartość
%           B. Błąd przybliżenia jest mniejszy lub równy od zadanej
%              tolerancji

%%
% Niech dane będzie równanie x^4 - x - 10 = 0
% Możemy je przekształcić na równanie postaci x=g(x) na kilka sposobów:
% x = 10 / (x^3 - 1)
% x = (x + 10)^(1/4)
% x = ((x + 10)^(1/2))/x
% w tej metodzie miejscem zerowym jest punkt przecięcia funkcji g(x) oraz
% funkcji y = x

% g = @(x) 10./(x.^3 - 1);                    % wariant 1
% g = @(x) (x + 10).^(1/4);                     % wariant 2
g = @(x) ((x + 10).^(1/2))./x;              % wariant 3

x = -10:0.001:10;

figure
plot(x,g(x),'LineWidth',2)
hold on
plot(x,x,'LineWidth',2)
grid on
ylim([0 10])
xlim([0 5])

NOfIter = 100;                  % liczba iteracji
x0 = 4;                         % punkt startowy
```

```
xb = banach(g, NOfIter, x0);

% poniższy rysunek jest tylko i wyłącznie dla wizualizacji działania
% metody
figure
plot(x,g(x),'LineWidth',2)
hold on
plot(x,x,'LineWidth',2)
hold on
for ii=1:NOfIter-1
plot(xb(ii),g(xb(ii)),'r.','MarkerSize', 15, 'LineWidth',2)
hold on
plot([xb(ii) xb(ii+1)],[g(xb(ii)) g(xb(ii))],'r-','LineWidth',1)
hold on
plot([xb(ii+1) xb(ii+1)],[g(xb(ii)) g(xb(ii+1))],'r-','LineWidth',1)
hold on
end
plot(xb(NOfIter),g(xb(NOfIter)),'b.','MarkerSize', 20, 'LineWidth',5)
grid on
ylim([0 5])
xlim([0 5])


%% funkcja fzero
% tutaj na wejściu jest oryginalna funkcja f(x) = x^4 - x - 10

gg = @(x) x.^4 - x - 10;

xf = fzero(gg, x0);


% proszę sobie popróbować te 3 warianty przekształcenia oryginalnej
% funkcji.
%
% W 1-szym wariancie widać, że metoda jest rozbieżna.
% W 2-gim wariancie metoda jest zbieżna i szybko prowadzi do rozwiązania,
% już kilka iteracji daje nam ten sam wynik co funkcją 'fzero'
% W 3-cim wariancie metoda jest zbieżna ale potrzebuje większej liczby
% iteracji, np. 100.
```