Warszawa, 2-6.05.2020

PROJEKT WNUM

ZADANIE 2 #25

Politechnika Warszawska

Wydział Elektroniki i Technik Informacyjnych

Przedmiot: WNUM

Prowadzący projekt: dr inż. Andrzej Miękina

Wykonawca: Maciej Kaczkowski

1. CEL DOŚWIADCZEŃ, WSTĘP TEORETYCZNY

Celem doświadczeń było wyznaczenie zer wielomianu

$w(x) = x^6 + 9x^5 - 59x^4 - 5511x^3 + 1316x^2 + 44308x - 162720$

za pomocą kombinacji metod: siecznych, stycznych (Newtona), deflacji liniowej i kwadratowej oraz metody Mullera w wersji I i II. Następnie zbadano wpływ wielkości Δx, używanej jako kryterium zatrzymania metod iteracyjnych, na dokładność wyznaczania pierwiastków. Jako składowe wektora odniesienia przyjęto wartości wyznaczone za pomocą procedury *roots*. W przypadku każdej z metod korzystano ze wzorów podanych na wykładzie.

Metoda siecznych

Metoda wyznacza następny punkt za pomocą siecznej przechodzącej przez poprzedni i aktualny punkt. Miejsce jej przecięcia z osią OX wskazuje następny punkt.

$\rho \cong 1{,}618$

$C \cong \left(\frac{f'(x)}{2 \cdot f''(x)}\right)^{1{,}118}$

Metoda stycznych (Newtona)

Metoda wykorzystująca funkcję pochodną, aby wyznaczyć styczną do funkcji w danym punkcie. Następnie miejsce przecięcia stycznej z osią OX wskazuje kolejny punkt w którym zaczyna od nowa.

$\rho = 2$

$C = \frac{f'(x)}{2 \cdot f''(x)}$

Metoda Mullera

Opiera się na aproksymacji funkcji w otoczeniu punktu za pomocą funkcji kwadratowej. Występuje w dwóch wersjach: I opartej na pochodnych i szeregu Taylora (uogólniona metoda siecznych), II opartej na interpolacji dwóch punktów (uogólniona metoda stycznych).

$\rho_I = 3$

$\rho_{II} = 1{,}84$

Deflacja

Jest to procedura obniżania stopnia wielomianu, w sytuacji kiedy znamy jeden z jego pierwiastków (lub parę pierwiastków zespolonych). W doświadczeniu użyto deflacji za pomocą algorytmu Hornera, liniowej (jeden pierwiastek rzeczywisty) lub kwadratowej (para pierwiastków zespolonych).

Jako kryterium zatrzymania metod iteracyjnych przyjęto wartość bezwzględną różnicy pomiędzy pierwiastkiem odniesienia, uzyskanym za pomocą polecenie *roots*, a pierwiastkiem uzyskanym w danej iteracji metody.

Do oszacowania błędów użyto zagregowanych błędów względnych wektora estymat pierwiastków wielomianu, normy 2 i ∞.

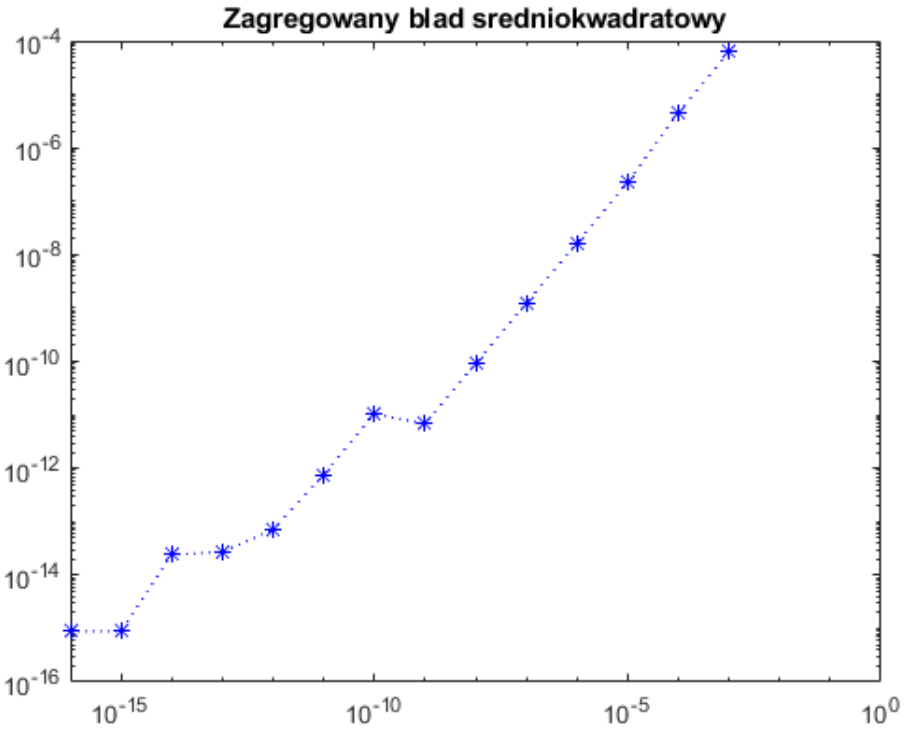
2. PUNKT 1

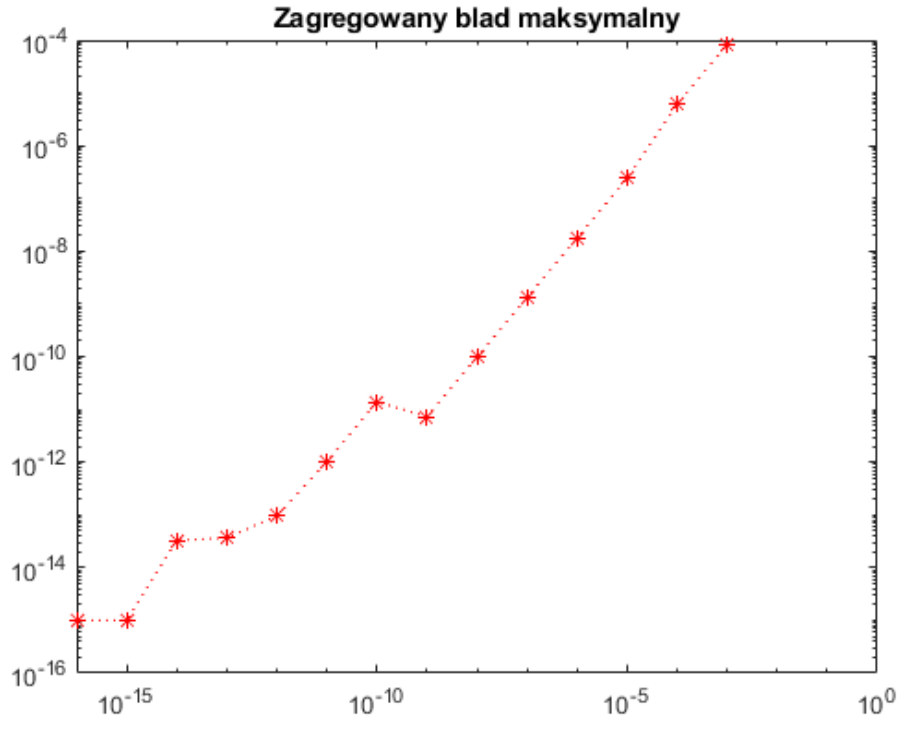
Dla podanej dokładności $\Delta x = 10^{-3}$ obliczono kolejne pierwiastki wielomianu, według następującego algorytmu:

1. Metoda siecznych (większy pierwiastek rzeczywisty wynoszący 8)
2. Deflacja liniowa
3. Metoda stycznych (mniejszy pierwiastek rzeczywisty wynoszący -9)
4. Deflacja liniowa
5. Metoda Mullera w wersji I (para pierwiastków zespolonych o mniejszej części rzeczywistej wynoszące -8+7j oraz -8-7j)
6. Deflacja kwadratowa
7. Metoda Mullera w wersji II (para pierwiastków zespolonych o większej części rzeczywistej wynoszące 4+2j oraz 4-2j)

3. PUNKT 2

Zbadano wpływ zmian parametru Δx na dokładność wyznaczenia pierwiastków wielomianu w algorytmie opisanym w punkcie 1. Zakres badania wynosił $[10^{-16}; 10^{-3}]$ w krokiem co dekadę. Uzyskano następujące zależności:





4. OBSERWACJE, WNIOSKI OGÓLNE

Zgodnie z przewidywaniami im mniejsza wartość parametru Δx (a co za tym idzie lepsza dokładność wyznaczenia pierwiastków) tym mniejszy zagregowany błąd. Jedyne wyjątek od tej reguły występuje pomiędzy $\Delta x = 10^{-10}$ a $\Delta x = 10^{-9}$.

Dla najmniejszych wartość parametru Δx wartość zagregowanego błędu przestaje się zmieniać. Prawdopodobnie jest to związane ze zbliżeniem się do wartości epsilona maszynowego, co powoduje, że osiągnięcie lepszej dokładności nie jest możliwe.

Zagregowany błąd maksymalny jest w niewielkim stopniu większy niż zagregowany błąd średniokwadratowy.

Badane metody iteracyjne, w normalnej pracy, wykonują kilka lub kilkanaście iteracji. Osiągnięcie maksymalnej ilości iteracji (1000) świadczy o tym, że w implementacji występuje błąd lub żądana dokładność jest niemożliwa do osiągnięcia ze względu na epsilon maszynowy.

W metodzie Mullera w wersji II dodano zabezpieczenie na wypadek gdyby powstała w kolejnej iteracji macierz była osobliwa, a co za tym idzie nieodwracalna. Było to spowodowane sporadycznym występowaniem błędu związanego z tym, że operacji odwracania nie dało się wykonać, zatem wynik metody osiągał wartość NaN.

Metoda Mullera tak naprawdę wyznacza jeden pierwiastek zespolony, a nie parę. Drugi, sprzężony pierwiastek wyznaczano za pomocą funkcji conj().

5. LISTING KODU

GŁÓWNY M-PLIK

```
clc
clear all

coefficients0 = [1, 9, -59, -1155, 1316,
44308, -162720];
p_roots = transpose(roots(coefficients0));
%pierwiastki uzyskane z roots, w kolejności sa
to -8+7j, -8-7j, -9, 8, 4+2j, 4-2j
c_roots = zeros(1,6); %miejsce na pierwiastki
obliczone metodami iteracyjnymi

%przyblizenie calkowite wartosci p_roots(przy
użyciu format long widać, że
%procedura roots nie daje dokładnie wartości
całkowitych pierwiastków
for n = 1:6

    if isreal(p_roots(n))
        p_roots(n) = round(p_roots(n));
    else
        a = round(real(p_roots(n)));
        b = round(imag(p_roots(n)));
        p_roots(n) = a + b*1i;
    end
end

Deltax = 10^-3;

% krok 1 - metoda siecznych
c_roots(4) = sieczne(10, Deltax,
coefficients0);

%krok 2 - deflacja liniowa za pomoca algorymtu
Hornera
coefficients1 = deflacja_lin(c_roots(4),
coefficients0);

%krok 3 - metoda stycznych (Newtona)
c_roots(3) = styczne(-10, Deltax,
coefficients1);

%krok 4 - deflacja liniowa za pomoca algorymtu
Hornera
coefficients2 = deflacja_lin(c_roots(3),
coefficients1);

%krok 5 - metoda Mullera w wersji I
c_roots(2) = muller_I(-10, Deltax,
coefficients2);
c_roots(1) = conj(c_roots(2));

%kork 6 - deflacja kwadratowa za pomoca
algorytmu Hornera
coefficients3 = deflacja_kw(c_roots(2),
coefficients2);

%krok 7 - metoda Mullera w wersji II
c_roots(5) = muller_II(10, Deltax,
coefficients3);
c_roots(6) = conj(c_roots(5));


Deltax = zeros(1, 13);
Agreg_sqr = zeros(1, 13);
Agreg_inf = zeros(1,13);

for n = 1:14

    Deltax(n) = 10^-(17-n);

    c_roots(4) = sieczne(10, Deltax(n),
coefficients0);
    coefficients1 = deflacja_lin(c_roots(4),
coefficients0);
    c_roots(3) = sieczne(-10, Deltax(n),
coefficients1);

    coefficients2 = deflacja_lin(c_roots(3),
coefficients1);
    c_roots(2) = muller_I(-10, Deltax(n),
coefficients2);
    c_roots(1) = conj(c_roots(2));

    coefficients3 = deflacja_kw(c_roots(2),
coefficients2);
    c_roots(5) = muller_II(10,
Deltax(n),coefficients3);
    c_roots(6) = conj(c_roots(5));

    Agreg_sqr(n) = norm(c_roots-p_roots,
2)/norm(p_roots,2);
    Agreg_inf(n) = norm(c_roots-p_roots,
Inf)/norm(p_roots,Inf);

end

figure(1);
loglog(Deltax, Agreg_sqr, 'b:*');
title('Zagregowany blad sredniokwadratowy');

figure(2);
loglog(Deltax, Agreg_inf, 'r:*');
title('Zagregowany blad maksymalny');
```

FUNKCJE POMOCNICZE

```
function [root] = sieczne(start, Deltax,
coeff)

x_next = start + 1;
x_i = start;
x_prev = start + 1;
delta = abs(x_next - x_i);
count = 0;

while ((delta >= Deltax) && (count<1000))

    x_next = x_i - ((x_i -
x_prev)/(polyval(coeff,x_i)-
polyval(coeff,x_prev)))*polyval(coeff,x_i);
    delta = abs(x_next - x_i);
    x_prev = x_i;
    x_i = x_next;
    count = count+1;

end

root = x_i;

end



function [root] = styczne(start, Deltax,
coeff)

x_next = start + 1;
x_i = start;
delta = abs(x_next - x_i);
count = 0;

while ((delta >= Deltax) && (count<1000))

    x_next = x_i -
(polyval(coeff,x_i))/(polyval(polyder(coeff),x
_i));
    delta = abs(x_next - x_i);
    x_i = x_next;
    count = count+1;
```

```
end

root = x_i;

end
```

```
function [root] = muller_I(spoint, Deltax,
coeff)

x_next = 0;
x_i = spoint;
delta = abs(x_next - x_i);
count = 0;

while ((delta >= Deltax) && (count<1000))

    a =
(1/2)*polyval(polyder(polyder(coeff)),x_i);
    b = polyval(polyder(coeff),x_i);
    c = polyval (coeff,x_i);
    x_next = x_i - (2*c)/(b + sign(b)*sqrt(b^2
- 4*a*c));
    delta = abs(x_next - x_i);
    x_i = x_next;
    count = count+1;

end

root = x_i;

end
```

```
function [root] = muller_II(start, Deltax,
coeff)

x_i = start;
x_prev_prev = start + 1;
x_prev = start + 0.5;
x_next = start + 1;

delta = abs(x_next - x_i);
count = 0;

while ((delta > Deltax) && (count<1000))

    M = [-(x_i-x_prev)^2, x_i-x_prev; -(x_i-
x_prev_prev)^2, x_i-x_prev_prev];
    v = [polyval (coeff,x_i) -
polyval(coeff,x_prev); polyval(coeff,x_i) -
polyval (coeff,x_prev_prev)];

    if det(M)==0
        break;
    end

%zamiast funkcji Inv() użyto dzielenia
macierzy za pomocą operatora '\'
    wektor = M\v;


    a = wektor(1);
    b = wektor(2);
    c = polyval(coeff,x_i);
    x_next = x_i - (2*c)/(b + sign(b)*sqrt(b^2
- 4*a*c));
    delta = abs(x_next - x_i);
    x_prev_prev = x_prev;
    x_prev = x_i;
    x_i = x_next;
    count = count+1;

end

root = x_i;

end
```

```
function [deflated] = deflacja_lin(root,
coeff)

deflated = zeros(1,length(coeff)-1);
deflated(1) = coeff(1);

for n = 2 : length(coeff)-1
    deflated(n) = coeff(n) + (deflated(n-
1)*root);
end

end
```

```
function [deflated] = deflacja_kw(root, coeff)

deflated = zeros(1,length(coeff)-2);
p = 2*real(root);
r = -(abs(root))^2;

deflated(1) = coeff(1);
deflated(2) = coeff(2) + deflated(1)*p;

for n = 3 : length(coeff)-2
    deflated(n) = coeff(n) + deflated(n-1)*p +
deflated(n-2)*r;
end

end
```