	Wydział WARiE	Imię i Nazwisko Jan Lubina oraz Konrad Makowski		Grupa A1	Kierunek AiR
Prowadzący dr inż. Janusz Pochmara		Ocena spraw.	Rok/Sem. 2020/21	Podgrupa L2	Data wyk. ćw. 26.05.2021r.
Uwagi:					

Metoda znajdowania miejsc zerowych wynaleziona przez Leonarda Bairstow'a

1. Wprowadzenie

Metoda Bairstow'a to algorytm znajdujący rzeczywiste i zespolone miejsca zerowe wielomianu. Jest wydajnym algorytmem polegajającym na upraszaniu danego wielomianiu poprzez wielokrotne dzielenie przez taki trójmian, który w przybliżeniu jest dzielnikiem wielomianu wejściowego. Współczynniki szukanego wielmianu wyznaczamy rekurencyjnie za pomocą wzorów wyznaczonych z „dzielenia pisemnego" wielomianu.

Reszta z dzielenia będzie funkcja liniową, której współczynniki zależą od wartośći $\boldsymbol{r}$ i $\boldsymbol{s}$. Naszym zadaniem jest wyznaczenie takich wartośći współczynników, żeby ta reszta była jak najbliższa 0.

Pierwsze wartości $\boldsymbol{r}$ i $\boldsymbol{s}$ wyznaczamy jako współczynniki znormalizowanego wielomianu stworzonego z 3 głównych współczynników wielomianu wejściowego.

$\boldsymbol{\Delta r}$ i $\boldsymbol{\Delta s}$ to wartości o które poprawiamy pierwotne wartośći $\boldsymbol{r}$ i $\boldsymbol{s}$ w kolejnych iteracjach. Wyznaczamy je rozwiązując konkretny układ równań, który zostanie opisany w kolejnym podpunkcie.

Powtarzamy całą operację, aż wielomian wejściowy zostanie zredukowany do wielomianu stopnia drugiego, bądź pierwszego. Wtedy ostatnie miejsca zerowe możemy uzyskać rozwiązując równanie kwadratowe, bądź liniowe.

1. Pełen opis metody [1]

Mając wielomian postaci:

$$f_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{1}$$

Metoda baristowa dzieli ten wielomian przez funkcje kwadratową.

$$x^2 - rx - s \tag{2}$$

Po podzieleniu uzyskamy wielomian $\boldsymbol{f_{n-2}(x)}$

$$f_{n-2}(x) = b_2 + b_3 x + b_4 x^2 + \cdots + b_{n-1} x^{n-3} + b_n x^{n-2} \tag{3}$$

A reszta z dzielenia ma postać funkcji liniowej $\boldsymbol{R(x)}$

$$R(x) = b_1(x - r) + b_0 \tag{4}$$

Współczynniki wielomianu $\boldsymbol{f_{n-2}(x)}$ i reszty $\boldsymbol{R(x)}$ uzyskujemy wykorzystując algorytm dzielenia (pisemnego) wielomianów. Wyznaczamy następujące równania rekurencyjne

$$b_n = a_n \tag{5.a}$$

$$b_{n-1} = a_{n-1} + r b_n \tag{5.b}$$

$$b_i = a_i + r b_{i+1} + s b_{i+2} \text{ for } i = n - 2\ do\ 0 \tag{5.c}$$

Jeżeli $\boldsymbol{x^2 - rx - s}$ jest czynnikiem $f_n(x)$ to reszta $\boldsymbol{R(x)}$ jest zerowa i miejsca zerowe $\boldsymbol{x^2 - rx - s}$ są miejscami zerowymi funkcji $\boldsymbol{f_n}\ (\boldsymbol{x})$. Metoda Baristowa sprowadza się do szukania wartości $\boldsymbol{r, s}$ dla których $\boldsymbol{R(x)}$ jest zerowe.

Skoro $\boldsymbol{b_0}$ i $\boldsymbol{b_1}$ są funkcjami zależnymi od r i s, możemy rozpisać je za pomocą wzoru Taylora

$$b_1(r + \Delta r, s + \Delta s) = b_1 + \frac{\partial b_1}{\partial r}\Delta r + \frac{\partial b_1}{\partial s}\Delta s + O(\Delta r^2, \Delta s^2) \tag{6.a}$$

$$b_0(r + \Delta r, s + \Delta s) = b_0 + \frac{\partial b_0}{\partial r}\Delta r + \frac{\partial b_0}{\partial s}\Delta s + O(\Delta r^2, \Delta s^2) \tag{6.b}$$

Dla $\Delta s, \Delta r \ll 1, O(\Delta r^2, \Delta s^2) \approx 0$. Wyrażenia drugiego i większego stponia możemy pominąc, więc $(\Delta r, \Delta s)$, czyli poprawę wartości (r, s) możemy uzyskać przyrównując równania (6.a) i (6.b) do 0

$$\frac{\partial b_1}{\partial r}\Delta r + \frac{\partial b_1}{\partial s}\Delta s = -b_1 \tag{7.a}$$

$$\frac{\partial b_0}{\partial r}\Delta r + \frac{\partial b_0}{\partial s}\Delta s = -b_0 \tag{7.b}$$

By rozwiązać układ powyższych równań potrzebujemy pochodne cząstkowe $\boldsymbol{b_0}, \boldsymbol{b_1}$ Bairstow wykazał, że te pochodne możemy uzyskać dzieląć ponownie $\boldsymbol{f_{n-2}(x)}$ przez wyznaczony wielomian, co sprowadza się do ponownego wykorzystania równań rekurencyjnych $(\mathbf{5}.\boldsymbol{a}) - (\mathbf{5}.\boldsymbol{c})$.

$$c_n = b_n \tag{8.a}$$

$$c_{n-1} = b_{n-1} + rc_n \tag{8.b}$$

$$c_i = b_i + rc_{i+1} + sc_{i+2} \text{ dla } i = 1,2, \dots, n-2 \tag{8.c}$$

Gdzie

$$\frac{\partial}{\partial r} = c_1, \frac{\partial b_0}{\partial s} = \frac{\partial b_1}{\partial r} = c_2, \frac{\partial b_1}{\partial s} = c_3 \tag{9}$$

Układ równań $(7.a) - (7.b)$ możemy zapisać jako

$$c_2\Delta r + c_3\Delta s = -b_1 \tag{10.a}$$

$$c_1\Delta r + c_2\Delta s = -b_0 \tag{10.b}$$

Rozwiązując ten układ równań uzyskamy wartości $(\Delta r, \Delta s)$, o które poprawimy pierwotne wartości $(r,\ s)$ do $(r + \Delta r, s + \Delta s)$

Na końcu możemy obliczyć błąd przybliżenia (r, s)

$$\left|\epsilon_{a,r}\right| = \left|\frac{\Delta r}{r}\right| \cdot 100;\ \left|\epsilon_{a,s}\right| = \left|\frac{\Delta s}{s}\right| \cdot 100 \tag{11}$$

Gdy jedna z wartośći błędu jest wieksza od wybranej tolerancji powtarzamy cały proces dla nowych wartości $(r + \Delta r, s + \Delta s)$. Gdy osiągniemy tolerancje miejsca zerowe możemy wyznaczyć z równania kwadratowego

$$x = \frac{r \pm \sqrt{r^2 + 4s}}{2} \tag{12}$$

Cały algorytm powtarzamy do uzyskania wielomianu drugiego bądź pierwszego stopnia. W takim przypadku ostatnie miejsca zerowe obliczamy jak równanie kwadratowe lub liniowe.

3. Rozwiązanie metody w środowisku SciLab

```
clear;mode(0);clc;
//Funkcja wyswietla miejsca zerowe
//W szcególności wartości zespolone
function printRoots(X)
    for n=1:1:max(size(X))
        if (isreal(X(n)))
            mprintf("x%u = %f\n",n,X(n));
        else
            mprintf("x%u = %f %+fi\n",n,real(X(n)),imag(X(n)));
            end
    end
endfunction

//Funkcja znajduje miejsca zerowe r. kwadratowego
//wywoływana jest pod koniec programu, gdy wielomian
//jest zredukowany do trójmianu
function X = solveQuadratic(P)
    p = coeff(P);
    a = p(3)
    b = p(2)
    c = p(1)
    X(1) = (-b+sqrt(b.^2 - 4*a*c))/(2*a)
    X(2) = (-b-sqrt(b.^2 - 4*a*c))/(2*a)
endfunction
//Funkcja znajduje miejsca zerowe r. liniowego
//wywoływana jest pod koniec programu, gdy wielomian
//jest zredukowany do dwumianu
function X = solveLinear(P)
    p = coeff(P);
    a = p(2);
    b = p(1);
```

```
    X(1) = (-b/a);
endfunction
//Główna funkcja znajduje miejsca zerowe wielomianu metodą Bairstowa
//Przyjmuje poly oraz tolerancje błędu w zakresie od 0 do 1
function [X, Time] = BairstowMethod(P,tolerance)
    tic();
    order = degree(P);
    X = [] //Output
    //Pierwsza pętla w funkcji, sprawdza czy wielomian
    //jest stopnia większego niż 2

    //Wartości początkowe r i s to współczynniki
    //znormalizowanego wielomianu

    //stworzonego z 3 głównych współczynników wielomianu wejściowego
    while order > 2
        n = order+1;
        a = coeff(P);
        //Initial Guess
        r = a(n-1)/a(n)
        s = a(n-2)/a(n)

        //W pętli wyznaczane są wartości:
        //b(n) - współczynniki wielomianu będącego wynikiem
        //dzielenia wejściowego przez x^2 - rx - s

        //c(n) - uzyskujemy dzieląc wielomian utworzony
        //z współczynników b(n) przez x^2 - rx - s

        //Pętla przerywa się, gdy zostanie osiągnieta tolerancja
        while 1==1
            n = order+1;
            //B
            b(n) = a(n);
            b(n-1) = a(n-1)+r*b(n)
            //C
            c(n) = b(n);
            c(n-1) = b(n-1)+r*c(n);

            for n = n-2:-1:1
                b(n) = a(n)+r*b(n+1)+s*b(n+2);
                c(n) = b(n) + r*c(n+1) + s*c(n+2);
            end

            //Szukanie Δr, Δs
```

```
            D = [c(3) c(4); c(2) c(3)]
            e = [b(2); b(1)];
            [delta] = linsolve(D,e);

            //Nowe wartości miejsc zerowych
            r = r + delta(1);
            s = s + delta(2);

            //Wyznaczanie błędu
            Err_r = abs(delta(1)/r)
            Err_s = abs(delta(2)/r);

            if(Err_s <= tolerance && Err_r <= tolerance)
                break; //Przerwij pętle po uzyskaniu tolerancji
            end
        end
        //Tworzymy wyznaczony trójmian
        Q = poly([-s -r 1], 'x', 'c');

        //Dodajemy wyznaczone miejsca zerowe do wektora rozwiązań
        X = [X; solveQuadratic(Q)];

        //Dzielimy wielomian wejściowy przez wyznaczony trójmian
        P = pdiv(P,Q);
        order = degree(P);
    end

    //Rozwiązanie wielomianu 2-go stopnia
    if(order == 2) then
        X = [X; solveQuadratic(P)];

    //Rozwiązanie wielomianu 1-go stopnia
    elseif(order == 1) then
        X = [X; solveLinear(P)];
    end
    //Wyznaczenie czasu obliczeń [ms]
    Time = 1000*toc();
endfunction
```

4. Instrukcja korzystania ze skryptu

- Należy uruchomić skrypt w środowisku SciLab
- Następnie, w celu wyznaczenia miejsc zerowych wielomianu należy skorzystać z funkcji Bairstow Method, gdzie pierwszy argument to wielomian stworzony za pomocą wbudowanej funkcji poly, a drugi argument to wartość tolerancji błędu w zakresie (0 – 1)
- Funkcja zwraca wektor zawierający miejsca zerowe funkcji oraz czas obliczeń

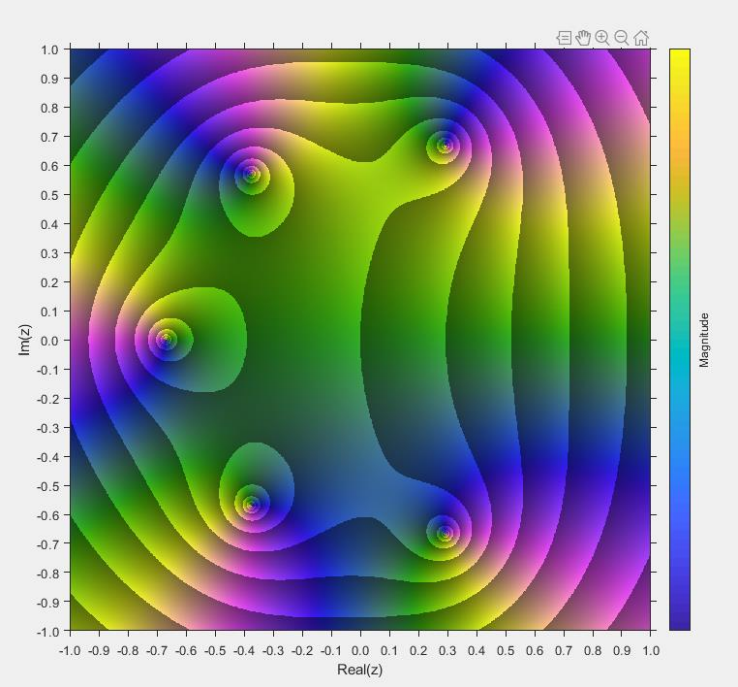
```
exec("BairstowMethod.sce");

//Definiowanie wielomianu
Input = poly([1 2 3 4 5 6],'x','c');

//Tolerancja błędu (0-1)
tolerance = 0.01;

//Wywołanie funkcji
 [X, t] = BairstowMethod(Input, tolerance);

 //Wyświetlenie informacji w konsoli
 mprintf ("Czas obliczeń: %fms \n",t);
 mprintf ("Tolerancja Błędu: %f \n",tolerance);
 mprintf ("Wielomian: ");
 disp(Input);
 mprintf("Miejsca zerowe: \n");
printRoots(X);
```

5. Przykładowe wywołanie skryptu

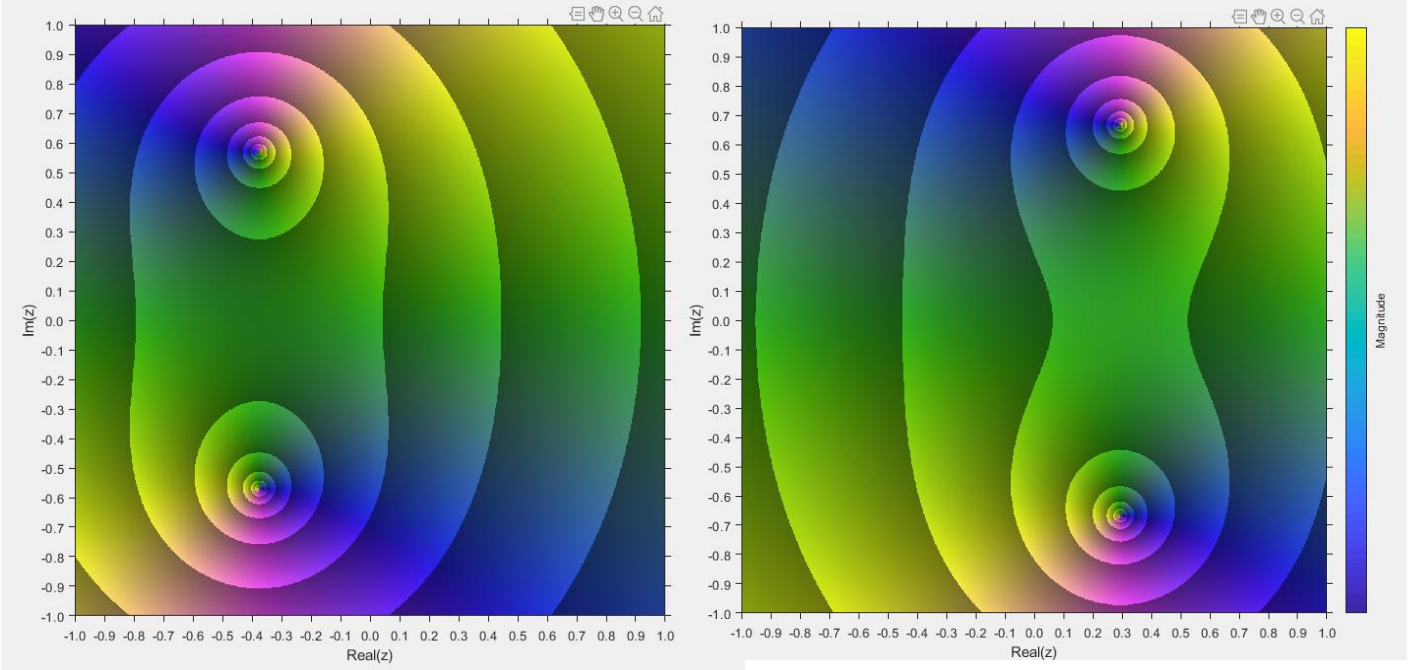
$$f(x) = 6x^5 + 5x^4 + 4x^3 + 3x^2 + 2x + 1$$

```
--> exec('C:\git\MNiS_BairstowMethod\Przykład.sce', -1)
Czas obliczeń: 6.856300ms
Tolerancja Błędu: 0.010000
Wielomian:
  1 +2x +3x² +4x³ +5x⁴ +6x⁵
Miejsca zerowe:
x1 = -0.375695 +0.570175i
x2 = -0.375695 -0.570175i
x3 = 0.294197 +0.668359i
x4 = 0.294197 -0.668359i
x5 = -0.670337
```

Wykres 1 Wykres wielomianu $f(x)$ w płaszczyźnie zespolonej z kolorowaną dziedziną [2]



Wykres 2 Wszystkie wyznaczone wielomiany $x^2 - rx - s$, przez które dzielono $f(x)$ [2]



6. Wnioski

Metoda Bairstow'a niesie ze sobą wiele korzyści. Najważniejszą z nich, w odróżnieniu od innych poznanych nam metod, jest znajdowanie wszystkich miejsc zerowych wielomianu, nawet zespolonych, używając do tego jedynie rzeczywistej arytmetyki. Nie potrzebujemy także początkowego przedziału, w którym zmienia się znak funkcji. Do tego trwa poniżej 10ms, co jest niesamowitym wynikiem. Inne metody dla porównania trwały od 0.5 do 2 ms, a mając na uwadze, że nasza metoda w podanym przykładzie znajduje aż 6 miejsc zerowych (10/6=1.67), to możemy stwierdzić, że jest bardzo zoptymalizowana.

Metoda Bairstow'a niesie ze sobą wiele korzyści. Przedewszystkim, w odróżnieniu od innych poznanych nam metod jest w stanie znaleźć wszystkie miejsca zerowe wielomianu, nawet te nierzeczywiste. Dla przykładowego wielomianu czas trwania programu wynosił ok. 7ms. Porównując ten czas do metody bisekcji lub metody Newtona, które kolejno trwały ok. 2ms i 0.5ms. Należy pamiętać, że te metody znajdują tylko pojedyńcze miejsca rzeczywiste. Dzieląc czas obliczeń przez ilość znalezionych miejsc zerowych otrzymamy następujące wartości:

Metoda Newtona	Metoda bisekcji	Metoda Bairstowa
$0.5ms$	$2.0ms$	$1.4ms$

Można zauważyć, że dla tego przykładu metoda Newtona jest o 0.6ms szybsza, ale wyznacza tylko jedno miejsce zerowe rzeczywiste.

7. Źródła

[1] - Bairstow Method (opis metody)
https://nptel.ac.in/content/storage2/courses/122104019/numerical-analysis/Rathish-kumar/ratish-1/f3node9.html

[2] – Wykresy stworzone z wykorzystaniem skryptu „colorcet.m" (matlab)
na licencji Creative Commons BY License https://colorcet.com

Dodatkowo:

Wikipedia - https://en.wikipedia.org/wiki/Bairstow%27s_method

Opis metody na przykładzie - https://www.youtube.com/watch?v=BjJJF1Wbuww