Wizualizacja szybkości zbieżności metody Czebyszewa (w dziedzinie zespolonej) zastosowanej do znalezienia zera wielomianu: $w_n(x) = a_0 + a_1x + ... + a_{n-1}x^{n-1} + T_n(x))$

Elżbieta Jowik, Filip Chrzuszcz

13 stycznia 2020

Spis treści

1 Wstęp

1.1 Ogólny schemat metod iteracyjnego znajdowania zer funkcji

Krok 1.
Równanie $f(x) = 0$ przekształcamy do postaci:

$$x = \phi(x),$$

gdzie: $\phi(x) = x - g(x)f(x)$, g jest funkcją ciągłą, $g \neq 0$
Punkt x^* taki, że równanie jest spełnione nazywa się *punktem stałym.*
Krok 2.
Tworzymy ciąg kolejnych przybliżeń $x^{(0)}$, $x^{(1)}$, ..., $x^{(p)}$, ... (w założeniu zbieżny do x^*) taki, że:

$$x^{(p+1)} = \phi(x^{(p)})$$

gdzie $x^{(0)}$ jest *przybliżeniem początkowym.*
Taka procedura nazywana jest *procedurą iteracyjną* a funkcja ϕ *funkcją iteracyjną.*
Krok 3.
Procedurę iteracyjną kończymy, wówczas gdy kolejne przybliżenia x^* różnią się odpowiednio mało (mówimy wtedy o zbieżności) lub wykonana już została maksymalna zadana liczba kroków (wówczas mówimy o brak zbieżności).

1.2 Metoda Czebyszewa

Metoda Czebyszewa jest przykładem procedury, w której dla danego równania:

$$f(x) = 0$$

ciąg punktów jest obliczany wg wzoru:

$$x^{n+1} = \phi(x^n) \ (n \geq 0),$$

gdzie ϕ wyraża się przez f.
Dlatego też metoda Czebyszewa określana jest mianem metody iteracyjnej.
Załóżmy, że $x^{(0)} \in \mathbb{C}$ jest danym przybliżeniem początkowym.

Kolejne przybliżenia $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, ... w metodzie Czebyszewa wyznacza się za pomocą następujących wzorów:
dla k = 0, 1, ...

$$y^{(k)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$
$$x^{(k+1)} = y^{(k)} - \frac{f''(x^{(k)})(y^{(k)}-x^{(k)})^2}{2f'(x^{(k)})}$$

Na potrzeby projektu, zrezygnowaliśmy ze sprowadzania zadanego wielomianu do postaci naturalnej. Do obliczenia wartości wielomianu $w_n(x)$ oraz pochodnych wykorzystaliśmy związek rekurencyjny spełniany przez *wielomiany Czebyszewa*.

1.3 Wielomiany Czebyszewa

Wielomiany Czebyszewa, ze względu na mnogość własności są ważnym narzędziem matematyki stosowanej.
Wielomiany Czebyszewa I rodzaju można określić wzorem rekurencyjnym:

$$T_0(x) = 1,\ T_1(x) = x$$
$$T_n(x) = 2 \cdot T_{n-1}(x) - T_{n-2}(x),\ (n \geq 2).$$

2 Funkcje opracowane na potrzeby realizacji założeń projektu

2.1 Funkcja służąca do obliczania wartości wielomianu T_n oraz jego pochodnych w punkcie x

```
function [t,dt, ddt] = Values(n,x)
    % Funkcja wykorzystuje związek rekurencyjny, spełn. przez w. Czebyszewa.

    % Funkcja zwraca:
    % t = Tn(x), dt =  Tn'(x), ddt = Tn''(x)

    % Wiadomo, że:
    % T0(x) = 1,  T1(x) = x,  Tk(x) = 2x * T_k-1(x) + T_k-2(x)

    % Wyznaczenie wektorów t, dt i ddt t.że:
    % t(i) = Ti(x), dt(i) = Ti'(x), ddt(i) = Ti''(x)

    t = zeros(n);
    dt = zeros(n);
    ddt = zeros(n);

    t(1) = x;
    t(2) = 2 * x^2 - 1;
    dt(1) = 1;
    dt(2) = 4 * x;
    ddt(1) = 1;
    ddt(2) = 4;

    for i = 3:1:n
        t(i) = 2*x * t(i-1) - t(i-2);
        dt(i) = 2 * dt(i-1) + 2 * x * dt(i-1) - t(i-2);
        ddt(i) = 4 * dt(i-1) + 2 * x * ddt(i-1) - ddt(i-2);
    end

    t = t(n);
    dt = dt(n);
    ddt = ddt(n);

end
```

2.2 Funkcja obliczająca wartość wielomianu w punkcie z wykorzystaniem schematu Hornera

```
function y = Horner(a,x)
    n = max(size(a));
    m = max(size(x));
    x=x(:); a=a(:);
    y = zeros(m,1);
    for k = 1 : n
            y = a(k) + x .* y;
    end

end
```

2.3 Funkcja odpowiadająca za wyznaczenie zer wielomianu: $w_n(x) = a_0 + a_1 * x + a_2 * x^2 + ... + a_{n-1} * x^{n-1} + T_n$ metodą Czebyszewa.

```
function[count] = IterativeMethod(a, x0, acc, iter_max)

    % Konwencja nazewnicza stworzona na potrzeby programu:
    % w_x = w(x)
    % a - współczynniki ai wielomianu wn
    % iter_max - maksymalna zadana liczba iteracji
    % x0 - dane przybliżenie początkowe
    % acc - dokładność z jaką funkcja wyznacza pierwiastki wielomianu w

    n = max(size(a));
    dx = acc + 1;
    count = 0; % liczba wykonanych iteracji

    a = fliplr(a);
    da = polyder(a); % 1. pochodna wielomianu o współczynnikach z a
    dda = polyder(da);  % 2. pochodna wielomianu o współczynnikach z a

    while abs(dx) > acc && count <= iter_max

        % do pętli wchodzimy tak długo, jak długo dokładność wyniku jest
        % niesatysfakcjonująca i nie wykonaliśmy maksymalnej liczby iteracji
        % (zarówno pożądana dokładność jak i maksymalna l. iteracji zadane są
        % na wejściu)

        % Obliczenie wartości wielomianu, jego 1. i 2. pochodnej w punkcie x0
        w1 = Horner(a, x0);
        dw1 = Horner(da, x0);
        ddw1 = Horner(dda, x0);

        [w2, dw2, ddw2] = Values(n, x0);
        w = w1 + w2;

        % Wykorzystujemy fakt, że pochodna sumy jest równa sumie pochodnych
        dw = dw1 + dw2;
        ddw = ddw1 + ddw2;
```

```
        if (abs(w) <= acc)
            x = x0;
            return
        end

        if (dw == 0)
            disp('Dzielenie przez 0!');
            return;
        end

        y = x0 - w/dw;
        dx = w/dw + (ddw * (y - x0)^2)/(2*dw);
        x = x0 - dx;
        x0 = x;
        count = count + 1;
    end
end
```

2.4 Funkcja wizualizująca szybkość zbieżności metody Czebyszewa zastosowanej do wyznaczania zera wielomianu w dziedzinie zespolonej

```
function = Visualization(a, b, c, d, n, m, poly, acc, iter_max)

    % Opis działania funkcji:
    % W danym obszarze prostokątnym [a, b] x [c, d] tworzymy siatkę punktów
    % (xk, yk),  gdzie xk należy do przedziału [a, b] a yj do [c, d],
    % xk = a + kh1, k = 0, 1, ..., n
    % yj = c + jh2, j = 0, 1, ..., m
    % h1 = (b-a)/n
    % h2 = (d-c)/m

    h1 = (b-a)/n;
    h2 = (d-c)/m;
    X = zeros(1, n+1); % Lista współrzędnych x punktów z siatki
    Y = zeros(1, n+1); % Lista współrzędnych y punktów z siatki

    for k = 0:1:n
        x = a + k*h1;
        X(k+1) = x;
    end

    for j = 0:1:m
        y = c + j  * h2;
        Y(j+1) = y;
    end


    % Tworzymy macierz A o wymiarach (n+1) x (m+1) wypełnioną zerami
    % lub jedynkami

    A = ones(n+1, m+1);

    % Dla każdego z punktów siatki (xk, yj), wykonujemy obliczenia metodą
    % Czebyszewa przyjmując punkt xk + i * yj jako przybliżenie początkowe
    % Wykonaną liczbę iteracji zapamiętujemy w odpowiedniej komórce macierzy A
```

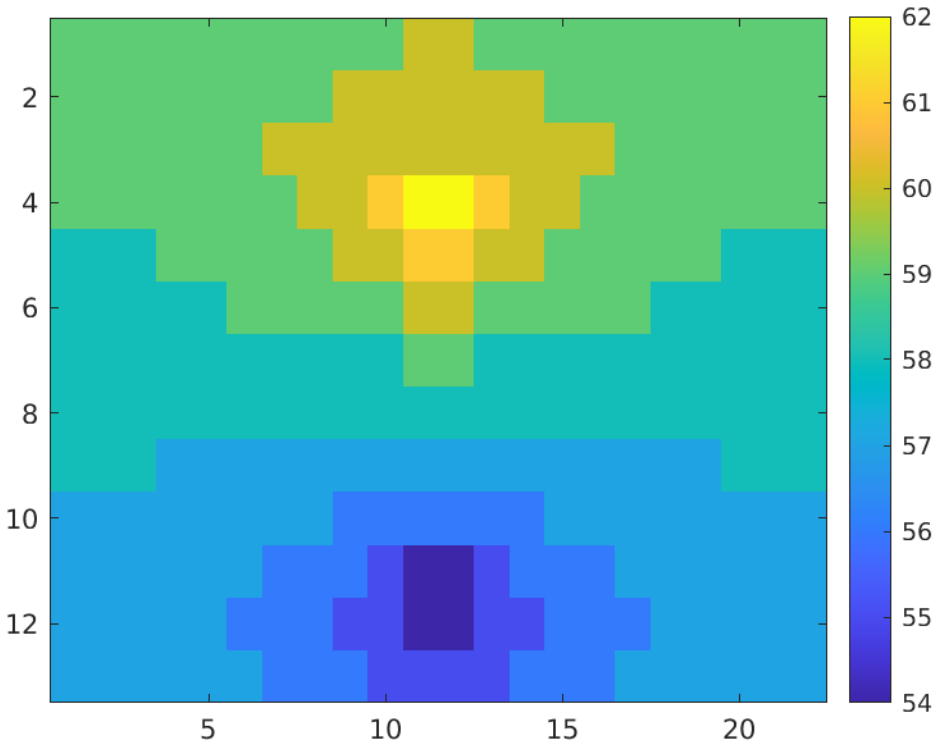
```
    for k = 0:1:n
        for j = 0:1:m
           x0 = X(k+1) +  Y(j+1) * i;
           count = IterativeMethod(poly, x0, acc, iter_max);
           A(k+1, j+1) = count;
        end
    end

    % Wyświetlamy macierz A
    imagesc(A);
    colormap("default");
    colorbar;
end
```

3 Testy

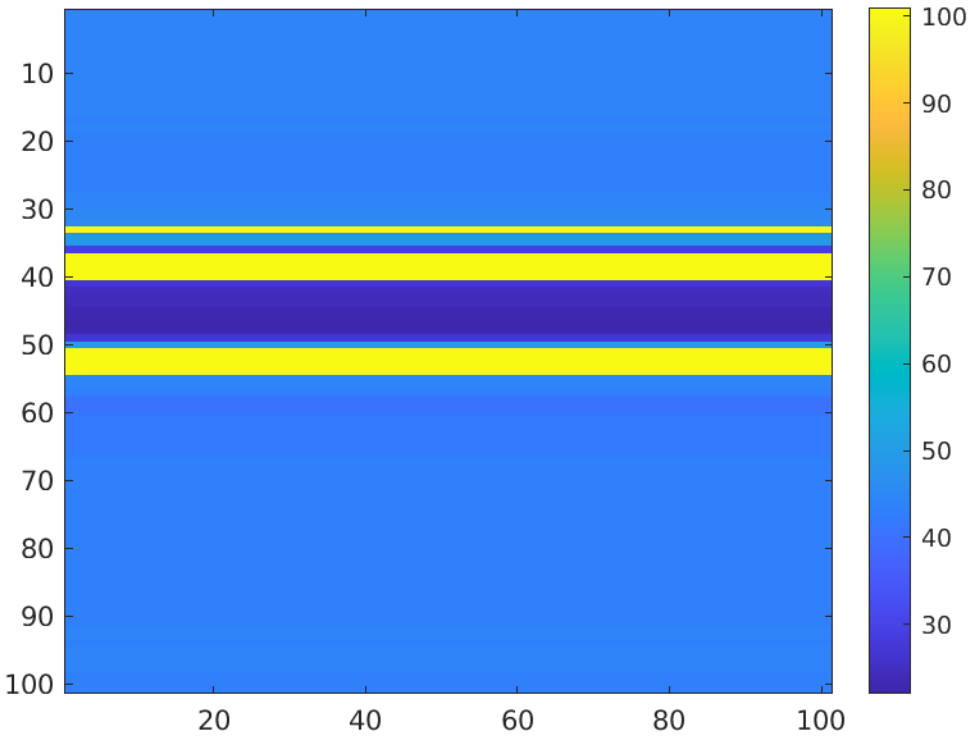
3.1 Test 1.:

```
a=-1;b=1;c=-1;d=1;m=21;n=12;acc=eps;iter_max=100;poly=[0,0,1];
Visualization(a, b, c, d, n, m, poly, acc, iter_max);
```



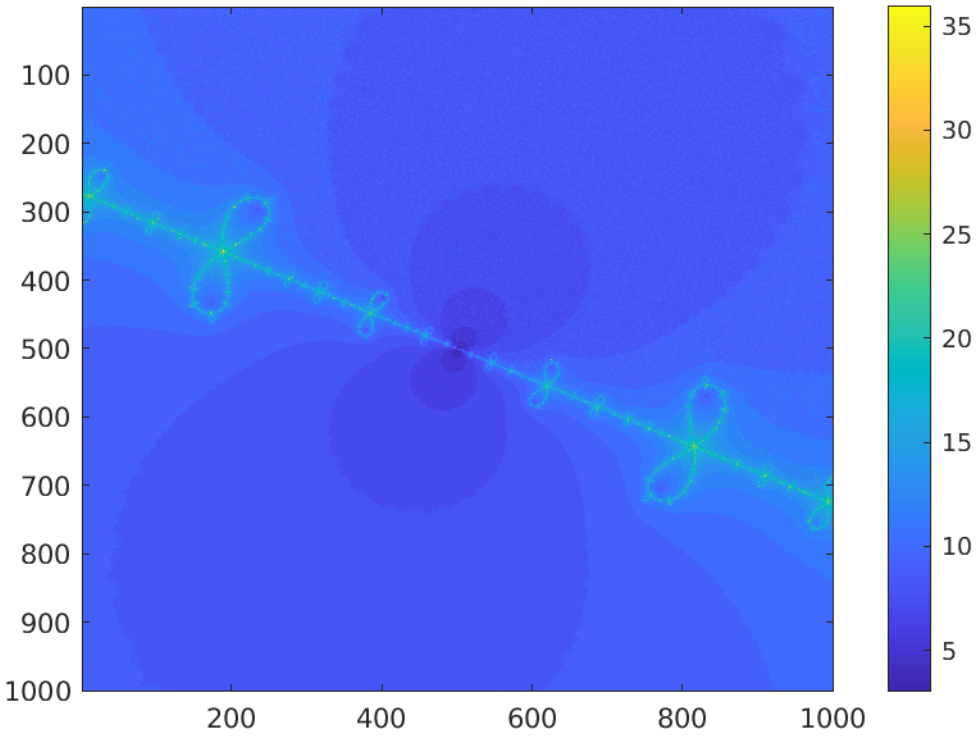
3.2 Test 2.:

```
a=-10;b=10;c=-eps;d=eps;m=100;n=100;acc=eps;iter_max=100;
poly=[10,21i,-21+10i,-15i];
Visualization(a, b, c, d, n, m, poly, acc, iter_max);
```



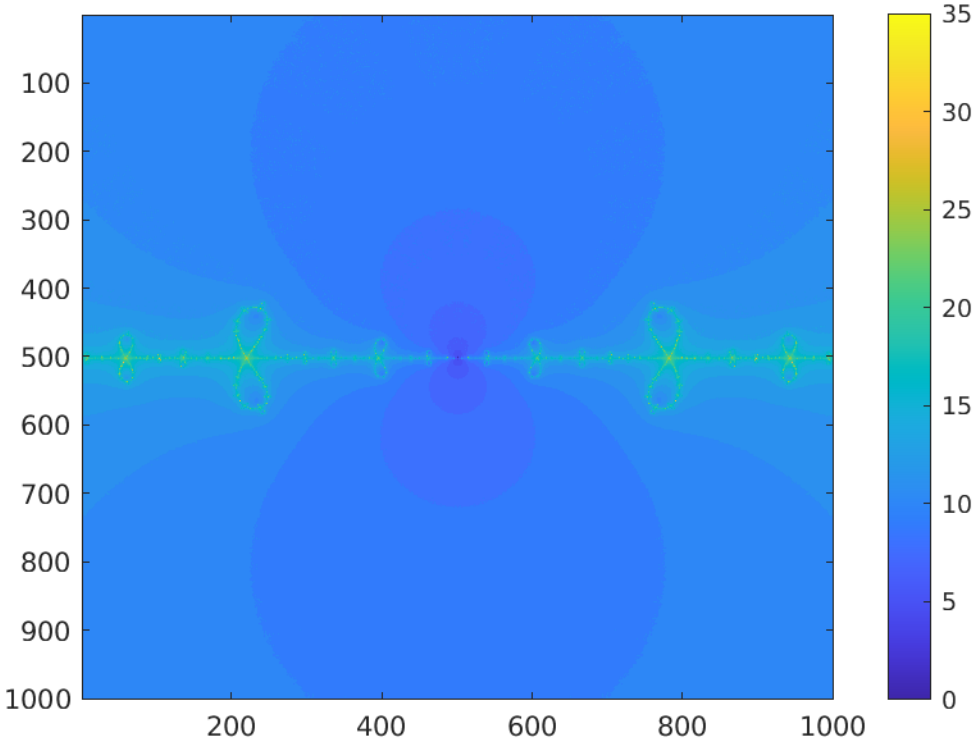
3.3 Test 3.:

```
a=-100;b=100;c=-100;d=100;m=1000;n=1000;acc=eps;iter_max=1000;poly=[i,-i];
Visualization(a, b, c, d, n, m, poly, acc, iter_max);
```



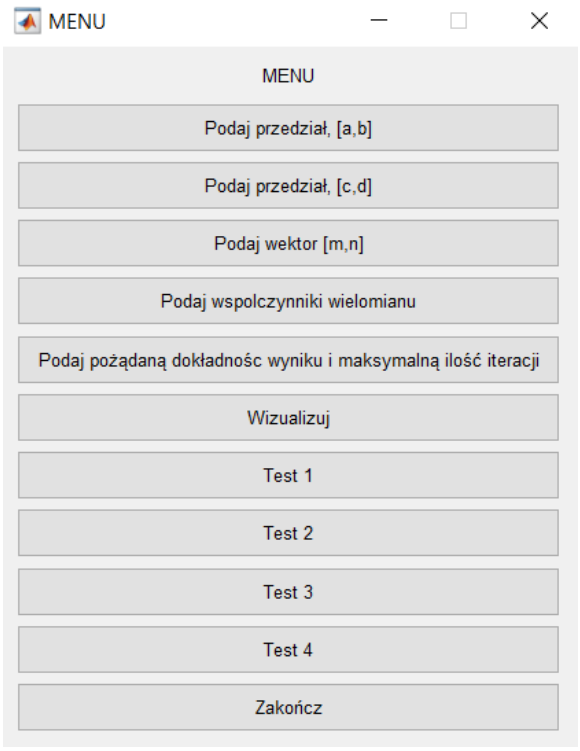
3.4 Test 4.:

```
a=-100;b=100;c=-100;d=100;m=1000;n=1000;acc=eps;iter_max=1000;poly=[1,-1];
Visualization(a, b, c, d, n, m, poly, acc, iter_max);
```



4 Program obliczeniowy

4.1 Interfejs programu obliczeniowego



4.2 Skrypt programu obliczeniowego

```
clear
clc

finish=12;
kontrol=1;
while kontrol~=finish

    kontrol=menu('MENU', 'Podaj przedział [a,b]', 'Podaj przedział [c,d]',
                 'Podaj wektor [m,n]','Podaj wspolczynniki wielomianu','
                 Podaj dokładność i ilość iteracji','Wizualizuj',
                 'Test 1','Test 2','Test 3','Test 4','Zamknij');
```

```
    switch kontrol
        case 1
            a =input('Podaj a ');
            b = input('Podaj b ');
        case 2
            c =input('Podaj c ');
            d= input('Podaj d ');
        case 3
            m =input('Podaj m ');
            n= input('Podaj n ');
        case 4
             poly = input('Podaj wielomian ');
        case 5
            acc = input('Tolerancja ');
            iter_max = input('Ilosc iteracji ');
        case 6
           Visualization(a, b, c, d, n, m, poly, acc, iter_max);
        case 7
            a=-1;b=1;c=-1;d=1;m=21;n=12;acc=eps;iter_max=100;poly=[0,0,1];
            Visualization(a, b, c, d, n, m, poly, acc, iter_max);
        case 8
            a=-10;b=10;c=-eps;d=eps;m=100;n=100;
            acc=eps;iter_max=100;poly=[10,21i,-21+10i,-15i];
            Visualization(a, b, c, d, n, m, poly, acc, iter_max);
        case 9
             a=-100;b=100;c=-100;d=100;m=1000;
             n=1000;acc=eps;iter_max=1000;poly=[i,-i];
            Visualization(a, b, c, d, n, m, poly, acc, iter_max);
        case 10
             a=-100;b=100;c=-100;d=100;m=1000;
             n=1000;acc=eps;iter_max=1000;poly=[1,-1];
            Visualization(a, b, c, d, n, m, poly, acc, iter_max);
        case 11
            break

    end
end
```