Metody numeryczne
Projekt nr 1

Malwina Wojewoda

9 grudnia 2021

1 Treść zadania

Wyznaczanie zer wielomianu (i wizualizacja szybkości zbieżności metody) w(z) = $\sum_{k=0}^{n} a_k z^k$ w dziedzinie zespolonej metodą Jarrata. Do obliczania wartości wielomianu i jego pochodnej zastosować algorytm Hornera.

2 Opis metody

2.1 Algorytm Hornera

W programie do obliczania wartości wielomianu i jego pochodnej stosuję algorytm Hornera.
Aby go wykorzystać zapisuję funkcję $w(z)$ jako $w(z) = a_m + z(a_{m-1} + z(... + z(a_2 + z(a_1 + za_0))...))$. Następnie, począwszy od wielomianu znajdującego się w najbardziej wewnętrznym nawiasie obliczam potrzebne wartości:
$w_0 = a_0$,
$w_1 = a_1 + zw_0$,
$w_2 = a_2 + zw_1$
i tak dalej aż do $w_m = w(z)$.
W ten sposób otrzymuję wartość wielomianu $w(z)$ w wybranym punkcie z.

Aby otrzymać wartość pochodnej wielomianu różniczkuję go w takiej postaci jak wyżej zapisałam i zachowując oznaczenia mam:
$w(z) = a_m + zw_{m-1}$, gdzie w_{m-1} jest funkcją zmiennej z.
Różniczkując powyższą funkcję otrzymuję:
$w'(z) = w_{m-1} + zw'_{m-1}$
Teraz różniczkuję w'_{m-1} i wychodzi:
$w'(z) = w_{m-1} + z(w_{m-2} + zw'_{m-2})$
i tak dalej aż do otrzymania wzoru pochodnej: $w'(z) = w_{m-1} + z(w_{m-2} + z(... + z(w_2 + z(w_1 + zw_0))...))$. W tym momencie postępuję analogicznie jak wyżej, gdzie obliczałam wartości wielomianu $w(z)$ w wybranym punkcie z.

2.2 Metoda Jarratta

Metoda Jarratta jest iteracyjnym algorytmem wyznaczania przybliżonych wartości pierwiastków funkcji jednej zmiennej. Aby móc ją zastosować do danej funkcji musi mieć ona ciągłą pochodną drugiego stopnia. W zadaniu zajmuję się wielomianami, więc założenie to jest spełnione. Wzór iteracyjny na (k+1)-wsze przybliżenie wygląda następująco:

$$x_{k-1} = x_k - \frac{f(x_k)}{f'(x_k - \frac{1}{2}\frac{f(x_k)}{f'(x_k)}}),$$

gdzie k = 0, 1, 2, ...

3 Opis programu obliczeniowego

Mój program obliczeniowy składa się z następujących funkcji:

3.1 metodaHornera

Funkcja ta przyjmuje jako argumenty:

- *wspolczynniki* - jest to wektor współczynników wielomianu, gdzie *wspolczynniki*(1)=a_n jest współczynnikiem przy n-tej, najwyższej potędze czyli są one podane w kolejności od najwyższej potęgi do najmniejszej (współczynniki(i) = a_{n+1-i}, gdzie a_{n+1-i} jest współczynnikiem przy x^{n+1-i})
- z - liczba, dla której wyliczamy wartość wielomianu

Funkcja ta zwraca wektor dwuelementowy, w którym pierwszym elementem jest wartość wielomianu w danym punkcie z, a drugim oraz wartość pochodnej wielomianu w tym punkcie

```
function [wartosc, wartoscPochodnej] = metodaHornera(wspolczynniki, z)
    n = length(wspolczynniki);
    if n == 0       % pusta lista
        error("Lista wspolczynnik w nie moze byc pusta")
    end
    if n == 1       % wielomiany stopnia 0
        wartosc = wspolczynniki(1);
        wartoscPochodnej = 0;
        return
    end
    % wielomiany stopnia >=1:
    wartoscTmp = wspolczynniki(1); % zaczyna od a_{0}, czyli od najbardziej
       wewnetrznego nawiasu
    pochodnaTmp = wartoscTmp; % analogicznie zaczyna w najbardziej wewnetrznym
       nawiasie
    for iterator = 2:1:n-1  %oblicza wartosc iteracyjnie, zgodnie z algorytmem
       Hornera
        wartoscTmp = wartoscTmp*z + wspolczynniki(iterator);
        pochodnaTmp = pochodnaTmp*z + wartoscTmp;
    end
    wartosc = wspolczynniki(n) + wartoscTmp*z; %dodajemy poza petla, bo przy
       obliczaniu wartosci wielomianu jest o jeden nawias wiecej niz w
       przypadku obliczania wartosci pochodnej wielomianu
    wartoscPochodnej = pochodnaTmp;
```

3.2 metodaJarrattaWynik

Funkcja ta przyjmuje jako argumenty:

- *wspolczynniki* - jest to wektor współczynników wielomianu, gdzie *wspolczynniki*(1)=a_n jest współczynnikiem przy n-tej, najwyższej potędze czyli są one podane w kolejności od najwyższej potęgi do najmniejszej (współczynniki(i) = a_{n+1-i}, gdzie a_{n+1-i} jest współczynnikiem przy x^{n+1-i})
- *ile* - liczba iteracji, które mają się wykonać
- x_0 - przybliżenie początkowe

Funkcja ta zwraca przybliżenie pierwiastka wielomianu z wykorzystaniem metody Jarratta, po wykonaniu *ile* iteracji.

```
function wynik = metodaJarrattaWynik(wspolczynniki, ile, x_0)
    n = length(wspolczynniki);
    if n == 0       % pusta lista
        error('Lista wsp  czynnik w nie mo e by  pusta');
    end
    if n == 1 % wielomiany stopnia 0
```

```
        if wspolczynniki == 0 % funkcja stala rowna 0
            warning('Funkcja stala r wna 0');
            wynik = Inf;
            return
        else
            warning('Funkcja stala - brak miejsc zerowych');
        return
        end
    end
    x = complex(x_0); % zamienia podana liczbe na zespolona
    for i = 1:ile
        [f_x, fPrime_x] = metodaHornera(wspolczynniki, x); % wartosc wielomianu
            i jego pochodnej w punkcie x
        nawias = x - f_x/(2*fPrime_x); % wartosc nawiasu w mianowniku
        [~, fPrime_nawias] = metodaHornera(wspolczynniki, nawias); %wartosc
           mianownika
        x = x - f_x/fPrime_nawias; % kolejne przyblizenie, jesli petla bedzie
           sie dalej wykonywac staje sie to nowym punktem startowym
    end
    wynik = x;
```

3.3 metodaJarrattaIleIteracji

Funkcja ta przyjmuje jako argumenty:

- *wspolczynniki* - jest to wektor współczynników wielomianu, gdzie *wspolczynniki*(1)=a_n jest współczynnikiem przy n-tej, najwyższej potędze czyli są one podane w kolejności od najwyższej potęgi do najmniejszej (współczynniki(i) = a_{n+1-i}, gdzie a_{n+1-i} jest współczynnikiem przy x^{n+1-i})
- x_0 - przybliżenie początkowe
- *przyblizenie* - jeśli różnica między kolejnymi wyznaczonymi punktami jest mniejsza od tej wartości to następuje przerwanie pętli (warunek stopu)

Funkcja ta zwraca liczbę iteracji, dla których różnica między kolejnymi wyznaczonymi punktami jest mniejsza niż zadana wartość *przyblizenie*.

```
function licznik = metodaJarrattaIleIteracji(wspolczynniki, x_0, przyblizenie)
      n = length(wspolczynniki);
    if n == 0       % pusta lista
        error("Lista wspolczynnik w nie mo e  by   pusta");
    end
    if n == 1       % wielomiany stopnia 0
        if wspolczynniki == 0 % funkcja stala r wna 0
            warning('Funkcja stala r wna 0');
            licznik = 0;
            return
        else
            warning("Funkcja stala - brak miejsc zerowych");
        return
        end
    end
    x = complex(x_0);
    poprzednie = Inf; % poczatkowo wartosc poprzedniego przyblizenia ustawiona
       jest na Inf, aby przy pierwszym przejsciu petli z niej nie wyjsc
    licznik = 0; %licznik iteracji
    roznica = Inf; % poczatkowo wartosc roznicy ustawiona jest na Inf, aby przy
        pierwszym przejsciu petli z niej nie wyjsc
    while (roznica > przyblizenie && licznik < 100)
        [f_x, fPrime_x] = metodaHornera(wspolczynniki, x); % wartosc wielomianu
            i jego pochodnej w punkcie x
```

```
22          nawias = x - f_x/(2*fPrime_x); % wartosc nawiasu w mianowniku
23          [~, fPrime_nawias] = metodaHornera(wspolczynniki, nawias); %wartosc
                mianownika
24          x = x - f_x/fPrime_nawias; % kolejne przyblizenie
25           roznica = abs(x - poprzednie); % roznica miedzy i-tym a (i-1)-wszym
                przyblizeniem miejsca zerowego
26           poprzednie = x; %przy przejsciu do nowej petli zapisana zostaje
                wartosc poprzedniego przyblizenia
27           licznik = licznik +1;
28      end
29      if licznik == 100 %jesli liczba iteracji przekroczy 100 to uznaje, ze z
            tego punktu nie ma zbieznosci
30          licznik = Inf;
31      end
```

3.4 wizualizacja

Funkcja ta przyjmuje jako argumenty:

- *wspolczynniki* - jest to wektor współczynników wielomianu, gdzie *wspolczynniki*(1)=a_n jest współczynnikiem przy n-tej, najwyższej potędze czyli są one podane w kolejności od najwyższej potęgi do najmniejszej (współczynniki(i) = a_{n+1-i}, gdzie a_{n+1-i} jest współczynnikiem przy x^{n+1-i})
- a, b - krańcowe wartości na osi liczb rzeczywistych dla których będę badać zbieżność
- n - wyznaczy n punktów rozłożonych równomiernie na przedziale [a, b]
- c, d - krańcowe wartości na osi liczb zespolonych dla których będę badać zbieżność
- m - wyznaczy m punktów rozłożonych równomiernie na przedziale [c, d]
- *dokladnosc* - określa jak małą różnicę między i-tym a (i+1)-wszym przybliżeniem miejsca zerowego chcemy otrzymać

Funkcja ta zwraca wizualizację zbieżności metody dla wybranych punktów. Na osi poziomej oznaczone są części rzeczywiste punktu startowego, a na osi pionowej wartość części urojonej tego punktu. Im jaśniejszy kolor, tym szybciej funkcja zbiega do miejsca zerowego. Jeśli liczba iteracji przekroczyła 30, kolorowane jest na czarno. Na pionowym pasku przedstawiona została legenda, która mówi ile iteracji musiało zostać wykonanych aby otrzymać zadaną dokładność.

```
1  function obrazek = wizualizacja(wspolczynniki, a, b, n, c, d, m, dokladnosc)
2  %siatka punktow poczatkowych, dla ktorych bede badac zbieznosc
3  x = linspace(a, b, n); % n wartosci na osi liczb rzeczywistych, roz ozonych
       rowno na przedziale [a, b]
4  y = linspace(c, d, m); % m wartosci na osi liczb zespolonych, roz o onych
       rowno na przedziale [c, d]
5  [X,Y] = meshgrid(x, y); % X to macierz, w kt rej kazdy wiersz jest kopia x,
6  % a Y, w ktorej kazda kolumna jest kopia y
7  points = [X(:), Y(:)]; % tworzy macierz, w ktorej w pierwszej kolumnie sa
       wartosci na osi rzeczywistej, a w drugiej na zespolonej, ka dy wiersz
       reprezentuje jakis punkt poczatkowy
8  poczatkowePunkty = complex(points(:, 1), points(:, 2)); % wektor tych punktow
       jako liczb zepolonych
9  A = zeros(m, n); % macierz zer o m wierszach i n kolumnach
10 for iterator = 1:length(poczatkowePunkty) % iteruje po kolejnych punktach
11     if mod(iterator, m) == 0 %jesli element znajduje sie w pierwszym wierszu
12         indeksWiersza = 1; % obliczam na jakim miejscu w macierzy b dzie
               trzeba wstawic wartosc
13         indeksKolumny = fix(iterator/m);
14     else
15         indeksWiersza = m-mod(iterator, m)+1;
```

```
16          indeksKolumny = fix(iterator/m)+1;
17      end
18      if metodaJarrattaIleIteracji(wspolczynniki, poczatkowePunkty(iterator),
            dokladnosc) > 30 %jesli po 30 iteracjach nie b dzie spelniony warunek
            stopu to pole b dzie pokolorowane na maksymalna wartosc
19          A(indeksWiersza, indeksKolumny) = Inf;
20      else
21          % jesli warunek stopu bedzie szybciej niz w 30 iteracji to te liczbe
                iteracji wpisuje w odpowiednia komorke macierzy
22          A(indeksWiersza, indeksKolumny) = metodaJarrattaIleIteracji(
                wspolczynniki, poczatkowePunkty(iterator), 0.001);
23      end
24  end
25  colormap(flipud(hot)); %zmiana kolorowania, flipud odwraca os kolorowania
26  obrazek = imagesc(x, y, A); % wyswietla szybkosc zbieznosci dla wybranych
        punktow z osiami podpisanynmi wartosciami na osi rzeczywistej (poziomej) i
        zespolonej (pionowej)
27  colorbar % wyswietla pasek legendy
```

4 Przykłady

4.1 Przykład 1: $w(z) = 0$

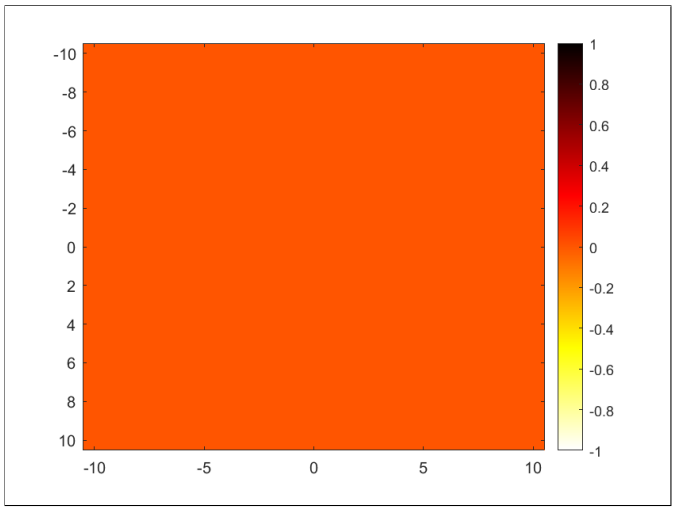
Weźmy wielomian stopnia 0 $w(z) = 0$ dla 10 iterowań, z punktem początkowym $z_0 = 8$:
Zwrócony wynik to:

```
>> metodaJarrattaWynik(0, 10, 8)
Warning: Funkcja stała równa 0
> In metodaJarrattaWynik (line 17)

ans =
   Inf
```

Sprawdźmy teraz jak ile iteracji musimy wykonać dla tego punktu, aby otrzymać wynik z dokładnością do 0.01:

```
>> metodaJarrattaIleIteracji(0, 8, 0.01)
Warning: Funkcja stała równa 0
> In metodaJarrattaIleIteracji (line 19)

ans =
     0
```

Wizualizacja zbieżności dla punktów $x_k + iy_j \in \mathbb{C}$, takich że $x_k \in [-10, 10], y_j \in [-10, 10]$,
$x_k = a + kh_1, k = 0, ..., n$,
$y_j = -10 + jh_2, j = 0, ..., m$,
$h_1 = \frac{10-(-10)}{20}, h2 = \frac{10-(-10)}{20}$
została przedstawiona na Rysunku 1.



Rysunek 1: Wizualizacja zbieżności $w(z) = 0$

Widać, że w tym przypadku funkcja działa poprawnie, ponieważ każdy punkt początkowy tej funkcji jest miejscem zerowym. Jest ich nieskończenie wiele.

4.2 Przykład 2: $w(z) = 8$

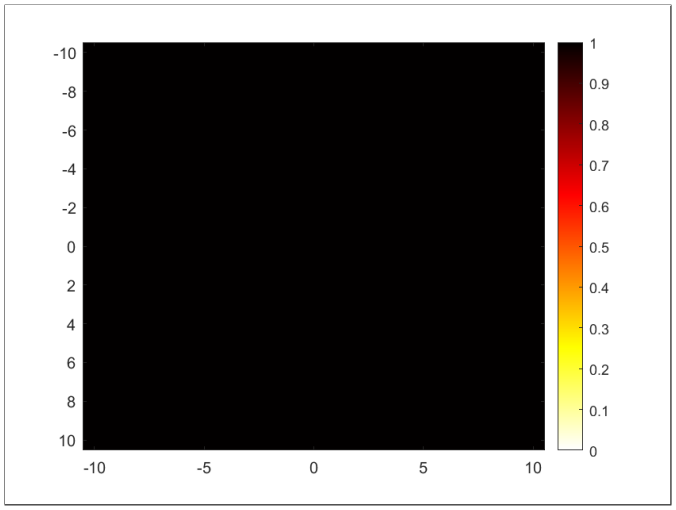
Weźmy wielomian stopnia 0 $w(z) = 8$ dla 5 iterowań, z punktem początkowym $z_0 = -5$:
Zwrócony wynik to:

```
>> metodaJarrattaWynik(8, 5, -5)
Warning: Funkcja stała - brak miejsc zerowych
> In metodaJarrattaWynik (line 21)
```

Sprawdźmy teraz jak ile iteracji musimy wykonać dla tego punktu, aby otrzymać wynik z dokładnością do 0.01:

```
>> metodaJarrattaIleIteracji(8, -5, 0.01)
Warning: Funkcja stała - brak miejsc zerowych
> In metodaJarrattaIleIteracji (line 23)

ans =
   Inf
```

Wizualizacja zbieżności dla punktów $x_k + iy_j \in \mathbb{C}$ została przedstawiona na Rysunku 2.



Rysunek 2: Wizualizacja zbieżności $w(z) = 8$

Widać, że w tym przypadku funkcja działa poprawnie, ponieważ ta funkcja nie ma miejsc zerowych. Przyjęłam konwencję, że wówczas wartość licznika iteracji jest równa nieskończoność, co spowodowało pokolorowanie całej macierzy na czarno.

4.3 Przykład 3: $w(z) = 2x - 15$

Mamy wielomian stopnia 1 $w(z) = 2x - 15$ dla 3 iterowań, z punktem początkowym $z_0 = 10$:
Wiadomo, że wielomian stopnia 1 ma 1 miejsce zerowe.

Zwrócony wynik to:

```
>> p3J = metodaJarrattaWynik([2, -15], 3, 10)

p3J =
   7.500000000000000
```

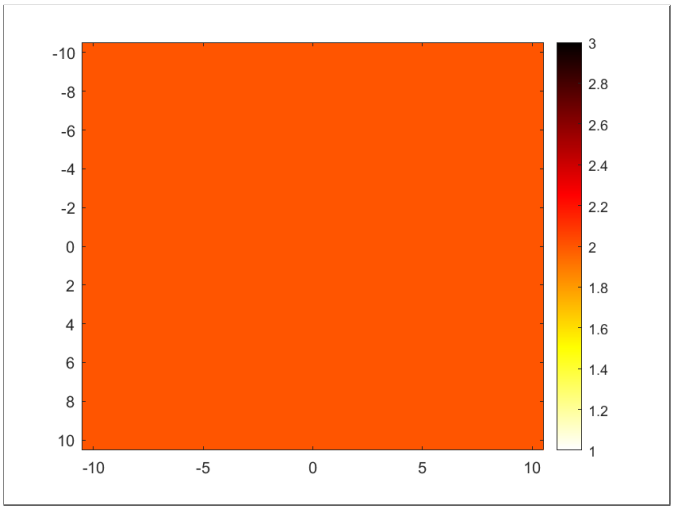
Sprawdzam, czy jest zgodny z rzeczywistym miejscem zerowym, korzystając z wbudowanej funkcji roots():

```
>> p3R = roots([2, -15])
p3diff = abs(p3R - p3J)

p3R =
   7.500000000000000

p3diff =
     0
```

Widać, że otrzymana wartość jest prawidłowa. Sprawdźmy teraz jak ile iteracji musimy wykonać dla tego punktu, aby otrzymać wynik z dokładnością do 0.0001:

```
>> p3Ile = metodaJarrattaIleIteracji([2, -15], 10, 0.0001)

p3Ile =
     2
```

Wizualizacja zbieżności dla punktów $x_k + iy_j \in \mathbb{C}$ została przedstawiona na Rysunku 3.



Rysunek 3: Wizualizacja zbieżności $w(z) = 2x - 15$

Widać, że w tym przypadku funkcja działa poprawnie, ponieważ wyznaczone miejsce zerowe jest takie samo jak obliczone przy pomocy funkcji roots(). Aby otrzymać ten wynik muszą zostać wykonane 2 iteracje, dla dowolnego punktu z płaszczyzny zespolonej.

4.4 Przykład 4: $w(z) = x^2 + x - 6$

Ten wielomian 2 miejsca zerowe rzeczywiste.
Obliczam przybliżenie miejsca zerowego wielomianu stopnia 2 $w(z) = x^2 + x - 6$ dla 5 iterowań, z punktem początkowym $z_0 = 20$:

Zwrócony wynik to:

```
>> p4J = metodaJarrattaWynik([1, 1, -6], 5, 20)

p4J =
     2
```

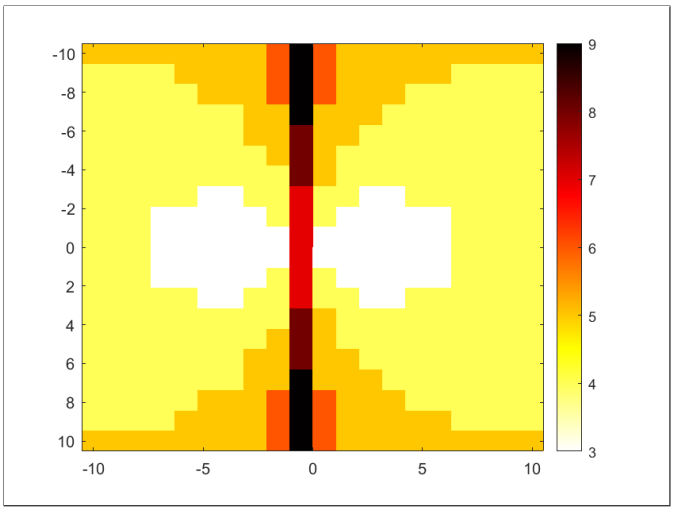
Sprawdzam, czy jest zgodny z rzeczywistym miejscem zerowym, korzystając z wbudowanej funkcji roots():

```
>> p4R = roots([1, 1, -6])

p4R =
    -3
     2
```

Widać, że wartość otrzymana przy użyciu metody Jarratta jest prawidłowa. Sprawdźmy teraz jak ile iteracji musimy wykonać dla tego punktu, aby otrzymać wynik z dokładnością do 0.005:

```
>> p4Ile = metodaJarrattaIleIteracji([1, 1, -6], 20, 0.005)

p4Ile =
     5
```

Wizualizacja zbieżności dla punktów $x_k + iy_j \in \mathbb{C}$ została przedstawiona na Rysunku 4.



Rysunek 4: Wizualizacja zbieżności $w(z) = x^2 + x - 6$

4.5 Przykład 5: $w(z) = 9x^2 - 36x + 366$

Ten wielomian ma 1 miejsce zerowe rzeczywiste.
Obliczam przybliżenie miejsca zerowego wielomianu stopnia 2 $w(z) = 9x^2 - 36x + 366$ dla 8 iterowań, z punktem początkowym $z_0 = -16$:

Zwrócony wynik to:

```
>> p5J = metodaJarrattaWynik([9, -36, 36], 8, -16)

p5J =
   1.997256515774989
```

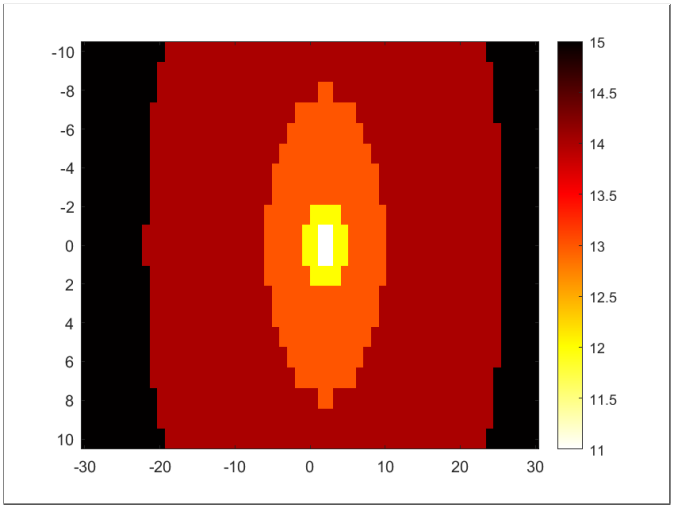
Sprawdzam, czy jest zgodny z rzeczywistym miejscem zerowym, korzystając z wbudowanej funkcji roots():

```
>> p5R = roots([9, -36, 36])

p5R =
     2
     2
>> p5diff = abs(p5R(1:1) - p5J)

p5diff =
   0.002743484225011
```

Widać, że wartość otrzymana przy użyciu metody Jarratta jest dość dokładna. Sprawdźmy teraz jak ile iteracji musimy wykonać dla tego punktu, aby otrzymać wynik z dokładnością do 0.00001:

```
>> p5Ile = metodaJarrattaIleIteracji([9, -36, 36], -16, 0.00001)

p5Ile =
    14
```

Wizualizacja zbieżności dla punktów $x_k + iy_j \in \mathbb{C}$ została przedstawiona na Rysunku 5.



Rysunek 5: Wizualizacja zbieżności $w(z) = 9x^2 - 36x + 366$

4.6 Przykład 6: $w(z) = 64x^2 - 28x + 652$

Ten wielomian nie ma pierwiastków rzeczywistych.
Obliczam przybliżenie miejsca zerowego wielomianu stopnia 2 $w(z) = 64x^2 - 28x + 652$ dla 20 iterowań, z punktem początkowym $z_0 = 1$:

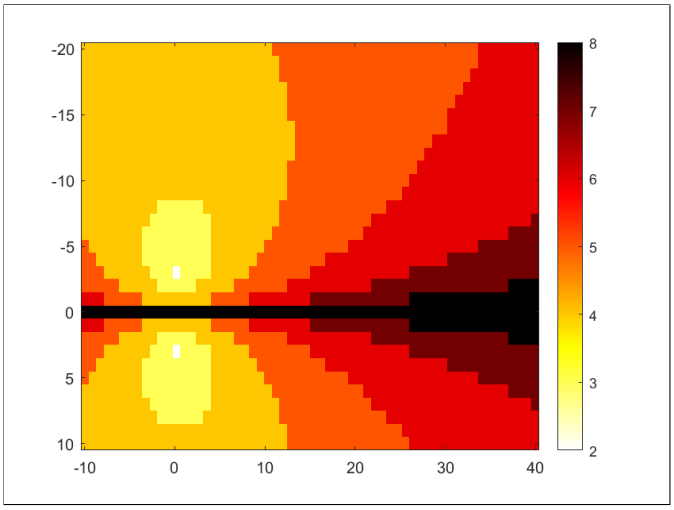
Zwrócony wynik to:

```
>> p6J = metodaJarrattaWynik([64, -28, 652], 20, 1)

p6J =
  -1.029628784192954
```

Sprawdzam, czy jest zgodny z rzeczywistym miejscem zerowym, korzystając z wbudowanej funkcji roots():

```
p6R = roots([64, -28, 652])

p6R =
  0.218750000000000 + 3.184281463297489i
  0.218750000000000 - 3.184281463297489i
```

Widać, że wartość otrzymana przy użyciu metody Jarratta jest błędna, ponieważ jest to wartość rzeczywista, nie liczba zespolona. Zobaczmy więc wizualizację zbieżności:
Wizualizacja zbieżności dla punktów $x_k + iy_j \in \mathbb{C}$ została przedstawiona na Rysunku 6.



Rysunek 6: Wizualizacja zbieżności $w(z) = 64x^2 - 28x + 652$

Widać, że jeśli weźmiemy inny punkt możemy otrzymać prawidłowy wynik. Spróbuję zatem dla punktu początkowego $z_0 = 2 + 3i$:

```
>> p6J2 = metodaJarrattaWynik([64, -28, 652], 20, 2+3i)

p6J2 =
  0.218750000000000 + 3.184281463297490i
```

Oraz obliczę błąd tego przybliżenia dla 20 iteracji:

```
>> p6diff = abs(p6R(1,1) - p6J2)

p6diff =
     8.886119947416683e-16
```

Błąd przybliżenia jest niewielki.

4.7 Przykład 7: $w(z) = 87x^4 + 2x^3 - 10x^2 - x + 14$

Obliczam przybliżenie miejsca zerowego wielomianu stopnia 4 $w(z) = 87x^4 + 2x^3 - 10x^2 - x + 14$ dla 12 iterowań, z punktem początkowym $z_0 = -40$:

Zwrócony wynik to:

```
>> p7J = metodaJarrattaWynik([87, 2, -10, -1, 14], 12, -40)

p7J =
   4.788018492767351
```

Sprawdzam, czy jest zgodny z rzeczywistym miejscem zerowym, korzystając z wbudowanej funkcji roots():

```
>> p7R = roots([87, 2, -10, -1, 14])

p7R =
 -0.484720428578795 + 0.420896795556327i
 -0.484720428578795 - 0.420896795556327i
  0.473226175705232 + 0.408088262892661i
  0.473226175705232 - 0.408088262892661i
```

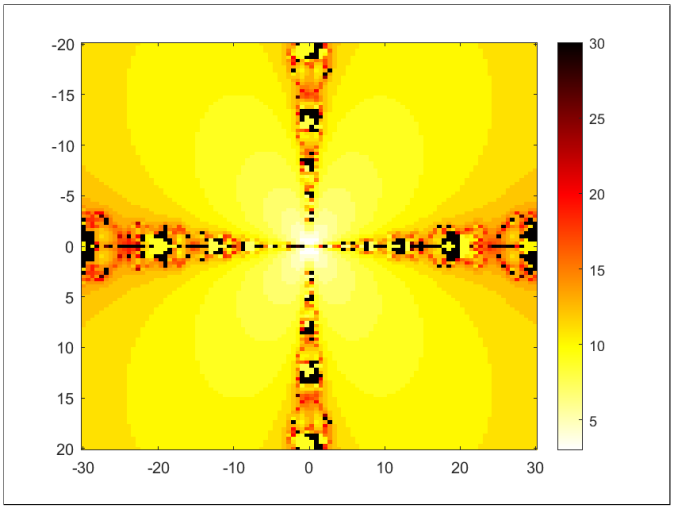
Widać, że wartość otrzymana przy użyciu metody Jarratta jest błędna. W takim razie sprawdzę jak będzie wyglądać, jeśli punktem startowym będzie liczba zespolona $13 - 4i$:

```
>> p7J2 = metodaJarrattaWynik([87, 2, -10, -1, 14], 12, 13-4i)

p7J2 =
  0.473226175705232 - 0.408088262892661i
```

Sprawdźmy teraz jak ile iteracji musimy wykonać dla tego punktu, aby otrzymać wynik z dokładnością do 0.00001:

```
>> p5Ile = metodaJarrattaIleIteracji([9, -36, 36], -16, 0.00001)

p5Ile =
    14
```

Wizualizacja zbieżności dla punktów $x_k + iy_j \in \mathbb{C}$ została przedstawiona na Rysunku 7.



Rysunek 7: Wizualizacja zbieżności $w(z) = 87x^4 + 2x^3 - 10x^2 - x + 14$

4.8 Przykład 8: $w(z) = 12x^6 + 11x^5 - 358x^4 - 840x^3 - 48x^2 + 96x + 1152$

Obliczam przybliżenie miejsca zerowego wielomianu stopnia 6 $w(z) = 12x^6 + 11x^5 - 358x^4 - 840x^3 - 48x^2 + 96x + 1152$ dla 7 iterowań, z punktem początkowym $z_0 = 34$:

Zwrócony wynik to:

```
>> p8J = metodaJarrattaWynik([12, 11, -358, -840, -48, 96, 1152], 7, 34)

p8J =
   6.095937958103034
```

Sprawdzam, czy jest zgodny z rzeczywistym miejscem zerowym, korzystając z wbudowanej funkcji roots():

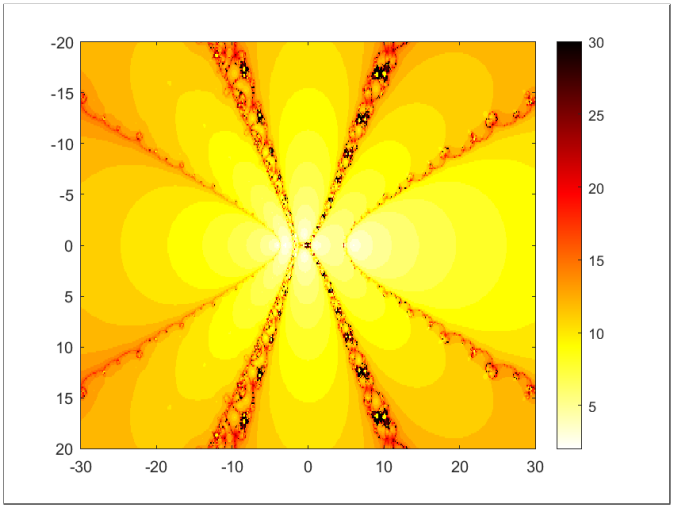
```
>> p8R = roots([12, 11, -358, -840, -48, 96, 1152])

p8R =
  5.999999999999996 + 0.000000000000000i
 -3.999999999999990 + 0.000000000000000i
 -3.056719876958696 + 0.000000000000000i
 -0.433216952657122 + 1.054742179817032i
 -0.433216952657122 - 1.054742179817032i
  1.006487115606267 + 0.000000000000000i
```

```
>> p7J2 = metodaJarrattaWynik([87, 2, -10, -1, 14], 12, 13-4i)

p7J2 =
  0.473226175705232 - 0.408088262892661i
```

Sprawdźmy teraz jak ile iteracji musimy wykonać dla tego punktu, aby otrzymać wynik z dokładnością do 0.01:

```
>> p8Ile = metodaJarrattaIleIteracji([12, 11, -358, -840, -48, 96, 1152], 34, 0.01)

p8Ile =
     9
```

Wizualizacja zbieżności dla punktów $x_k + iy_j \in \mathbb{C}$ została przedstawiona na Rysunku 8.



Rysunek 8: Wizualizacja zbieżności $w(z) = 12x^6 + 11x^5 - 358x^4 - 840x^3 - 48x^2 + 96x + 1152$

4.9 Przykład 9: $w(z) = 381x^9 - 8x^8 + 128x^7 + 5x^6 + 4x^5 + 8x^4 + 9x^3 + 1x^2 + 13x + 25$

Obliczam przybliżenie miejsca zerowego wielomianu stopnia 9 $w(z) = 381x^9 - 8x^8 + 128x^7 + 5x^6 + 4x^5 + 8x^4 + 9x^3 + 1x^2 + 13x + 25$ dla 8 iterowań, z punktem początkowym $z_0 = 16 + 2i$:

Zwrócony wynik to:

```
p9J = metodaJarrattaWynik([381, -8, 128, 5, 4, 8, 9, 1, 13, 25], 8, 16+2i)

p9J =
  3.408098004794578 + 0.428246802277727i
```

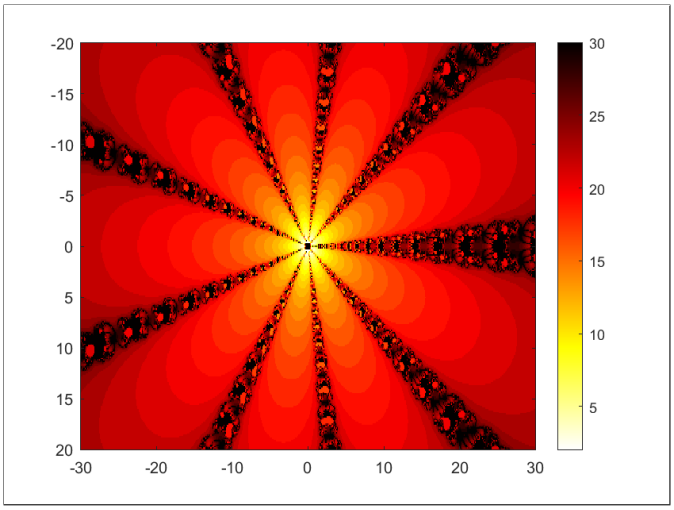
Sprawdzam, czy jest zgodny z rzeczywistym miejscem zerowym, korzystając z wbudowanej funkcji roots():

```
p9R = roots([381, -8, 128, 5, 4, 8, 9, 1, 13, 25])

p9R =
  0.673251081415587 + 0.300343998630966i
  0.673251081415587 - 0.300343998630966i
  0.324542245799711 + 0.697392310458979i
  0.324542245799711 - 0.697392310458979i
 -0.131483243118991 + 0.786470399959953i
 -0.131483243118991 - 0.786470399959953i
 -0.527286696522960 + 0.458704770563869i
 -0.527286696522960 - 0.458704770563869i
 -0.657049399818611 + 0.000000000000000i
```

Sprawdźmy teraz jak ile iteracji musimy wykonać dla tego punktu, aby otrzymać wynik z dokładnością do 0.01:

```
p9Ile = metodaJarrattaIleIteracji([381, -8, 128, 5, 4, 8, 9, 1, 13, 25], 16+2i, 0.001)

p9Ile =
    20
```

Wizualizacja zbieżności dla punktów $x_k + iy_j \in \mathbb{C}$ została przedstawiona na Rysunku 9.



Rysunek 9: Wizualizacja zbieżności $w(z) = 381x^9 - 8x^8 + 128x^7 + 5x^6 + 4x^5 + 8x^4 + 9x^3 + 1x^2 + 13x + 25$

4.10 Przykład 10: $w(z) = -56x^{13} + 333x^{12} + 147x^{11} + 999x^{10} - 12x^9 + 32x^7 + 97x^6 + 5x^5 + 6x^4 + 1x^2 + 3x + 51$

Obliczam przybliżenie miejsca zerowego wielomianu stopnia 9 $w(z) = -56x^{13} + 333x^{12} + 147x^{11} + 999x^{10} - 12x^9 + 32x^7 + 97x^6 + 5x^5 + 6x^4 + 1x^2 + 3x + 51$ dla 13 iterowań, z punktem początkowym $z_0 = 12 + 6i$:

Zwrócony wynik to:

```
p10J = metodaJarrattaWynik([-56, 333, 147, 999, -12, 0, 32, 97, 5, 6, 0, 1, 3,  51], 13, 12+6i)

p10J =
  2.013649376224300 + 0.603662690174133i
```

Sprawdzam, czy jest zgodny z rzeczywistym miejscem zerowym, korzystając z wbudowanej funkcji roots():

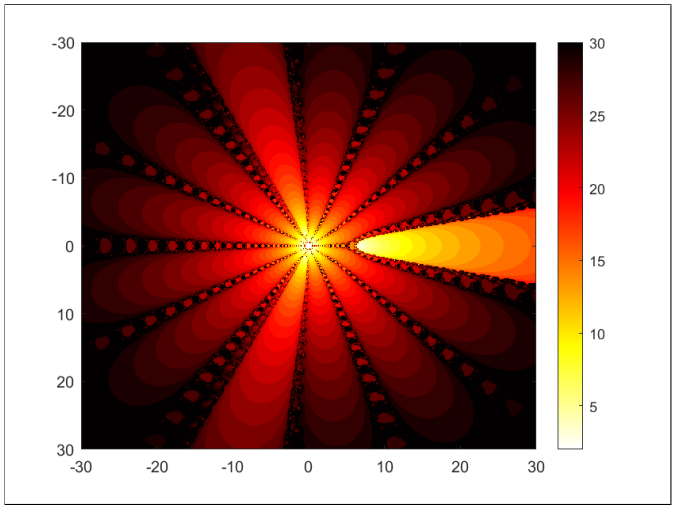
```
>> p10R = roots([-56, 333, 147, 999, -12, 0, 32, 97, 5, 6, 0, 1, 3,  51])

p10R =
  6.729740631503693 + 0.000000000000000i
 -0.395611234707528 + 1.590414229795951i
 -0.395611234707528 - 1.590414229795951i
 -0.693192519077972 + 0.237375631020338i
 -0.693192519077972 - 0.237375631020338i
 -0.470965550707565 + 0.598618765451894i
 -0.470965550707565 - 0.598618765451894i
  0.011425194239622 + 0.732070962330975i
  0.011425194239622 - 0.732070962330975i
  0.474330884460236 + 0.591757240495396i
  0.474330884460236 - 0.591757240495396i
  0.682357195755650 + 0.242999742280586i
  0.682357195755650 - 0.242999742280586i
```

Sprawdźmy teraz jak ile iteracji musimy wykonać dla tego punktu, aby otrzymać wynik z dokładnością do 0.001:

```
p10Ile = metodaJarrattaIleIteracji([-56, 333, 147, 999, -12, 0, 32, 97, 5, 6, 0, 1, 3,  51], 12+6i, 0.001)
```

```
p10Ile =
    22
```

Wizualizacja zbieżności dla punktów $x_k + iy_j \in \mathbb{C}$ została przedstawiona na Rysunku 10.



Rysunek 10: Wizualizacja zbieżności $w(z) = -56x^{13} + 333x^{12} + 147x^{11} + 999x^{10} - 12x^9 + 32x^7 + 97x^6 + 5x^5 + 6x^4 + 1x^2 + 3x + 51$

5 Analiza wyników

Porównam teraz zbieżność metody Jarratta, dla wybranych przykładów, sprawdzając jakie wartości są otrzymywane dla kolejnych iteracji. W pierwszej kolumnie zostały podane numery iteracji.

	przykład 3	**przykład 4**	**przyklad 8**	**przykład 10**
1	7.500000000000000	6.602999210734017	25.295130392823790	10.610374324582388 + 5.2249815162754 98i
2	7.500000000000000	2.618792459294002	18.867590589657567	9.395940472354161 + 4.534188702206354i
3	7.500000000000000	2.006687388295180	14.148328030957984	8.334137398050942 + 3.912784102090538i
4	7.500000000000000	2.000000011914840	10.727115498181211	7.402961345278147 + 3.346016355829732i
5	7.500000000000000	2	8.325178941884589	6.578205628160775 + 2.819552395951016i
6	7.500000000000000	2	6.792181874078571	5.829089427214266 + 2.323235378450696i
7	7.500000000000000	2	6.095937958103034	5.117859789435719 + 1.864656247315842i
8	7.500000000000000	2	6.000360301134374	4.427819671102359 + 1.482558888662360i
9	7.500000000000000	2	6.000000000021446	3.792567666048138 + 1.200166595200505i
10	7.500000000000000	2	6	3.239057075597964 + 0.993299551931028i

Sprawdzę jeszcze ile iteracji trzeba wykonać dla każdego z przykładów, aby otrzymać dokładność 0.001

	punkt startowy	liczba potrzebnych iteracji
przykład 1	8	0
przykład 2	-5	Inf
przykład 3	10	2
przykład 4	20	5
przykład 5	-16	10
przykład 6	2+3i	3
przykład 7	13-4i	10
przykład 8	34	9
przykład 9	16+2i	20
przykład 10	12+6i	22

Poniższa tabela przedstawia jaka liczba iteracji musi zostać wykonana, aby osiągnąć dokładność 0.01, dla kilku przykładów:

punkt startowy	przykład 5	przykład 8	przykład 10
60	9	11	21
40	9	10	18
34	8	9	17
27	8	8	15
24	8	8	14
20	8	7	13
15	8	6	10
12	7	5	8
8	7	4	5
7	7	3	3
6	7	2	17
5	6	6	34
4	6	4	23
1	5	2	4
0	6	2	2
-2	7	2	8
-5	7	4	15
-7	7	5	17
-10	8	7	Inf

Z testów można było zaobserowować, że dla punktu startowego, który jest liczbą rzeczywistą nie otrzymamy miejsca zerowego, które jest liczbą zespoloną, nierzeczywistą. Zbieżność zależy od przypadku, jak widać z wizualizacji, zwykle jest to w jakiś sposób symetryczne względem jakiejś osi przechodzącej przez punkt $0 + 0i$. Ponadto dla wielomianów wyższych stopni zbieżność wydaje się być wolniejsza niż dla wielomianów niższych stopni. Zbieżność metody zależy również w dużym stopniu od wybrania punktu startowego. Raczej należy wybierać takie, które znajdują się w pobliżu punktu $0 + 0i$.