Metody Numeryczne

Projekt 3 – zad 3.26

Aneta Skoniecka, 253141

Spis treści

1. Zadanie 1

Należy wyznaczyć wszystkie zera funkcji:

$$f(x) = 1.2 * \sin(x) - 2 * \ln(x + 2)$$

W przedziale [2, 12], używając dla każdego zera programu z implementacją medoty bisekcji i siecznych.

1.1 Zero funkcji

Zerem (lub używaną zamiennie nazwą: pierwiastkiem) funkcji f(x) jest takie x, dla którego jest spełnione:

$$f(x) = 0$$

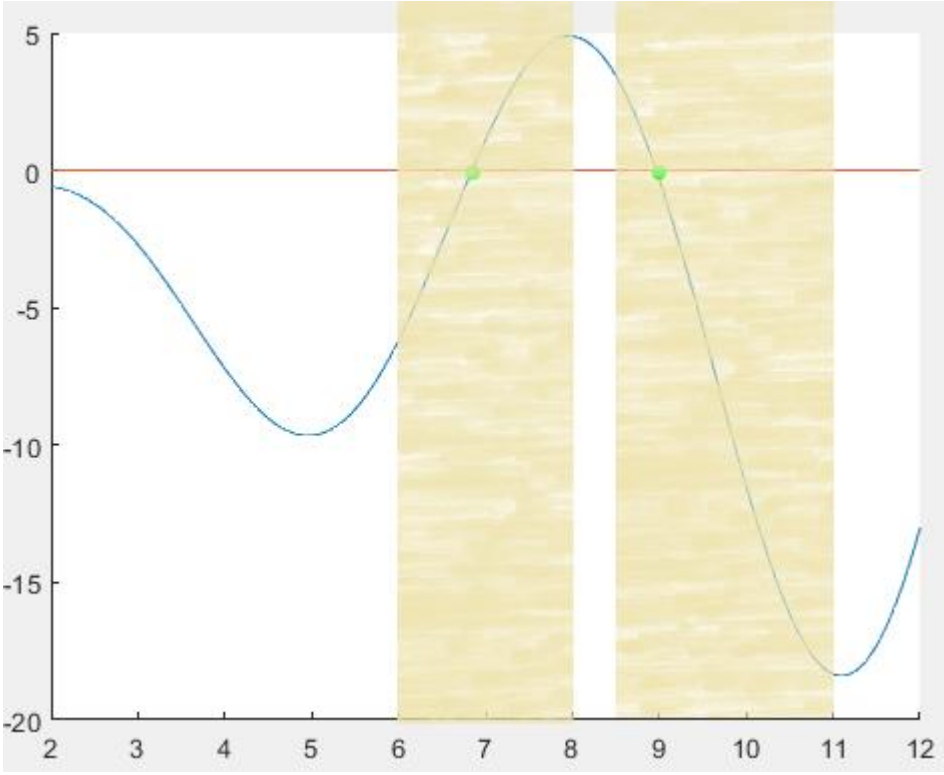
O ile znane są metody wyznaczania zer dla równań liniowych i kwadratowych, to dla funkcji nieliniowych sprawa nie jest taka oczywista.

1.2 Wyznaczanie zer funkcji nieliniowej

Aby wyznaczyć zera funkcji należy dla każdego zera wyznaczyć przedział <a,b>, w którym znajduje się tylko dane jedno zero. Mając okreslony taki przedział można wykorzystując metody iteracyjne wyznaczyć przybliżenie danego zera.

1.3 Wyznaczanie przedziału zawierającego pierwiastek

a. Można ręcznie oszacować przedział zawierający pierwiastek za pomocą wykresu, rysującego funkcję. Na potrzeby tego sprawozdania przygotowano funkcję matlabową generującą taki wykres, na którym można znaleźć zera funkcji.



Rysunek 1 - wykres funkcji z zadania 1

b. Drugą metodą jest użycie algorytmu, który wyznaczy przedział zawierający jeden pierwiastek. Zadając mu jakiś mały przedział początkowy algorytm bada iloczyn wartości

funkcji na krańcach przedziału i sukcesywnie go zmienia do momentu znalezienia zera funkcji. Warunkiem istnienia conajmniej jednego zera w przedziale <a,b> jest prawdziwość: $f(a) * f(b) < 0$.
Proponowane rozwiązanie nie jest idealne, jednak pozwala na szybkie oszacowanie przedziału.

```
function [ a, b ] = przedzial( x10, x20, n )
    beta = 1.2;
    x1 = x10;
    x2 = x20;
    for j=1:n
        if funI(x1) * funI(x2) < 0
            a = x1;
            b = x2;
        elseif abs(funI(x1)) < abs(funI(x2))
            x1 = x1 * beta * (x2 - x1);
        else
            x2 = x2 + beta * (x2 - x1);
        end
    end
end
```

1.4 Wybrane przedziały

a. Dla pierwszego zera: <6 ; 8>
b. Dla drugiego zera: <8.5 ; 11>

1.5 Metoda bisekcji

1.5.1 Opis algorytmu

Jest to metoda iteracyjna pozwalająca wyznaczyć zero funkcji.
Mając dany przedział <a,b> ,gdzie a i b spełniają równanie

$$f(a) * f(b) < 0 \text{ , } war.\ 1$$

dzielimy go w połowie by otrzymać przedziały <a,c> i <c,b>. Sprawdzamy w, którym przedziale znajduje się zero za pomocą *war. 1* i ten przedział staje się nowym przedziałem <a,b>, wchodzącym w kolejną iterację algorytmu.

1.5.2 Dokładność metody

Dokładność metody zależy od ilości iteracji n.

1.5.3 Zbieżność metody

Niech ε_i oznacza długość przedziału w i-tym kroku (iteracji). Mamy:

$$\varepsilon_{i+1} = \frac{1}{2} * \varepsilon_i \text{ ,}$$
$$\varepsilon_n = 2^{-n} * \varepsilon_0$$

Stąd startując z przedziału o szerokości ε_0 i oznaczając przez n ilość iteracji potrzebną do uzyskania dokładności ε_n dostajemy:

$$n = log_n(\frac{\varepsilon_0}{\varepsilon_n})$$

Metoda bisekcji jest więc zbieżna liniowo (p=1), ze współczynnikiem $k = \frac{1}{2}$.

1.5.4 Kod programu

```
function [ X, Y ] = bisekcja( a,b,n )
    if funI(a) * funI(b) > 0
```

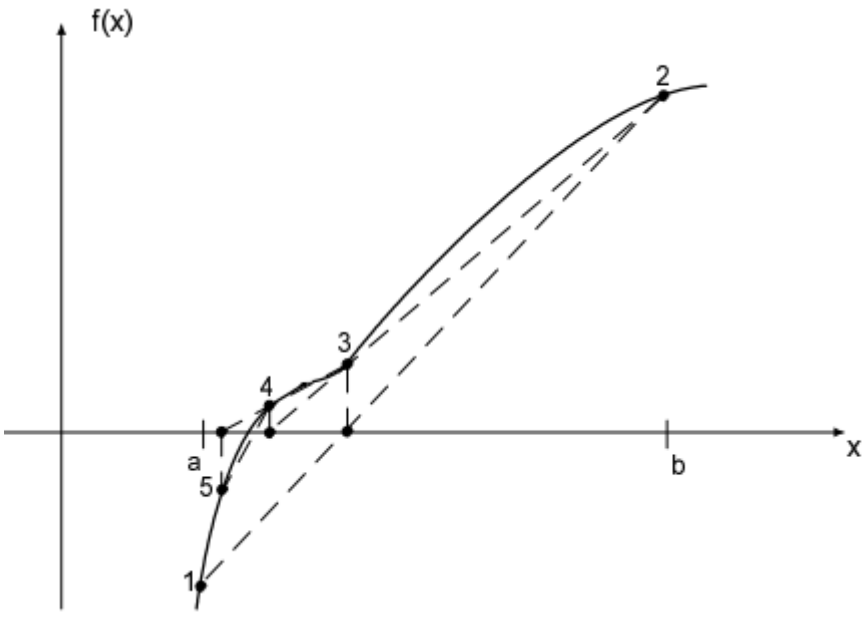
```
            error('niepoprawny przedzial \n');
        end
        i = 0;
        Y=zeros(n,1);
        X=zeros(n,1);
        for i=1:n
            c = (a+b) / 2;
            if funI(c) * funI(a) < 0
                b = c;
            elseif funI(c) * funI(b) < 0
                a = c;
            else
                error('niepoprawny przedzial \n');
            end
            if abs(funI(a)) < abs(funI(b))
                X(i) = a;
                Y(i) = funI(a);
            else
                X(i) = b;
                Y(i) = funI(b);
            end
        end
end

```

1.6 Metoda siecznych

1.6.1 Opis metody

Metoda ta polega na przeprowadzeniu siecznych pomiędzy końcami przedziału ostatnio wyznaczonego. Jak na rysunku:



Jeżeli te dwa punkty oznaczymy przez x_{n-1} i x_n, to nowy punkt w metodzie tej jest zdefiniowany wzorem:

$$x_{n-1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} = \frac{x_{n-1}f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}$$

1.6.2 Zbieżność metody

Rząd zbieżności: $p = (1 + \sqrt{5})/2 \approx 1.618$. Oznacza to, że metoda ta jest szybsza od metody bisekcji. Jest jednak ona zbieżna tylko lokalnie. Może być zawodna jeśli przedział izolacji zera funkcji jest zbyt mały.

1.6.3 Kod algorytmu

```
function [ X, Y ] = sieczne( a,b,n )
    if funI(a)*funI(b) > 0
        error('niepoprawny przedzial \n');
    end
    Y=zeros(n,1);
    X=zeros(n,1);
    for i=1:n
        c = b - ( ( funI(b)*(b-a) ) / ( funI(b)-funI(a) ) );
        a=b;
        b=c;
        if abs( funI(a) ) < abs( funI(b) )
            X(i)=a;
        else
            X(i)=b;
        end
        Y(i) = funI(X(i));
    end
end
```

1.7 Wyniki

1.7.1 Pierwsze zero

Bisekcja		Sieczne	
X(i)	Y(i)	X(i)	Y(i)
7.0000	1.1242	7.1155	1.8944
6.5000	-2.6022	7.1155	1.8944
6.7500	-0.6928	6.8549	0.0891
6.8750	0.2359	6.8432	0.0025
6.8125	-0.2244	6.8428	-0.0000
6.8438	0.0069	6.8428	0.0000
6.8281	-0.1085	6.8428	-0.0000
6.8359	-0.0508	6.8428	-0.0000
6.8398	-0.0219		
6.8418	-0.0075		
6.8428	-0.0003		
6.8433	0.0033		
6.8430	0.0015		
6.8429	0.0006		
6.8428	0.0001		

1.7.2 Drugie zero

Bisekcja		Sieczne	
X(i)	Y(i)	X(i)	Y(i)
9.7500	-8.6661	8.8952	0.6155
9.1250	-1.5848	8.9636	-0.0025

8.8125	1.3164	8.9633	0.0001
8.9688	-0.0505	8.9633	0.0000
8.8906	0.6558	8.9633	-0.0000
8.9297	0.3081	8.9633	-0.0000
8.9492	0.1302	8.9633	-0.0000
8.9590	0.0402		
8.9639	-0.0050		
8.9614	0.0176		
8.9626	0.0063		
8.9633	0.0006		
8.9636	-0.0022		
8.9634	-0.0008		
8.9633	-0.0001		

1.7.3 Legenda oznaczeń w wynikach

Gdzie:

i – numer iteracji algorytmu

X(i) – przybliżone zero

Y(i) – wartość funkcji w przybliżonym zerze

1.7.4 Wnioski z wyników

Widać, wyraźnie że to co zapisane zostało w pkt 1.7.2 – metoda siecznych jest dużo szybciej zbieżna niż metoda bisekcji, dzięki czemu szybciej wyznaczone zostało zero funkcji. Jednak aby metoda siecznych była skuteczna należy pamiętać o umiejętnie dobranym przedziale izolacji pierwiastka, tzn. By był odpowiednio wąski.

Metody iteracyjne mogą zawieźć gdy funkcja nie zbiega monotonicznie do pierwiastka, co można było zauważyć przy źle dobranych przedziałach.

2. Zadanie 2a

Używając metody Newtona, proszę znaleźć wszystkie pierwiastki rzeczywiste wielomianu czwartego stopnia, ze współczynnikami [a4, a3, a2, a1] = [2, 0.5, -5, 2 ,-3].

2.1 Metoda Newtona

Korzystając z rozwinięcia w szereg Taylora i wykorzystując przybliżenie liniowe jesteśmy w stanie dojść do iteracyjnej metody wyznaczania pierwiastków wielomianu. Metoda Newtona jest metodą przybliżającą jeden z pierwiastków wielomianu. Aby wyznaczyć wszystkie pierwiastki trzeba użyć metody Newtona dla wszystkich punktów, które podejrzewamy o sąsiedztwo z pierwiastkami.

Z przekształceń:

$$f(x_k + \Delta x_k) \approx f(x_k) + f'(x_k)\Delta x_k$$

$$f'(x_k) = [\frac{\partial f_i}{\partial x_j}(x_k)]_{nxn}$$

$$f(x_k) + f'(x_k)(x_{k+1} - x_k) = 0$$

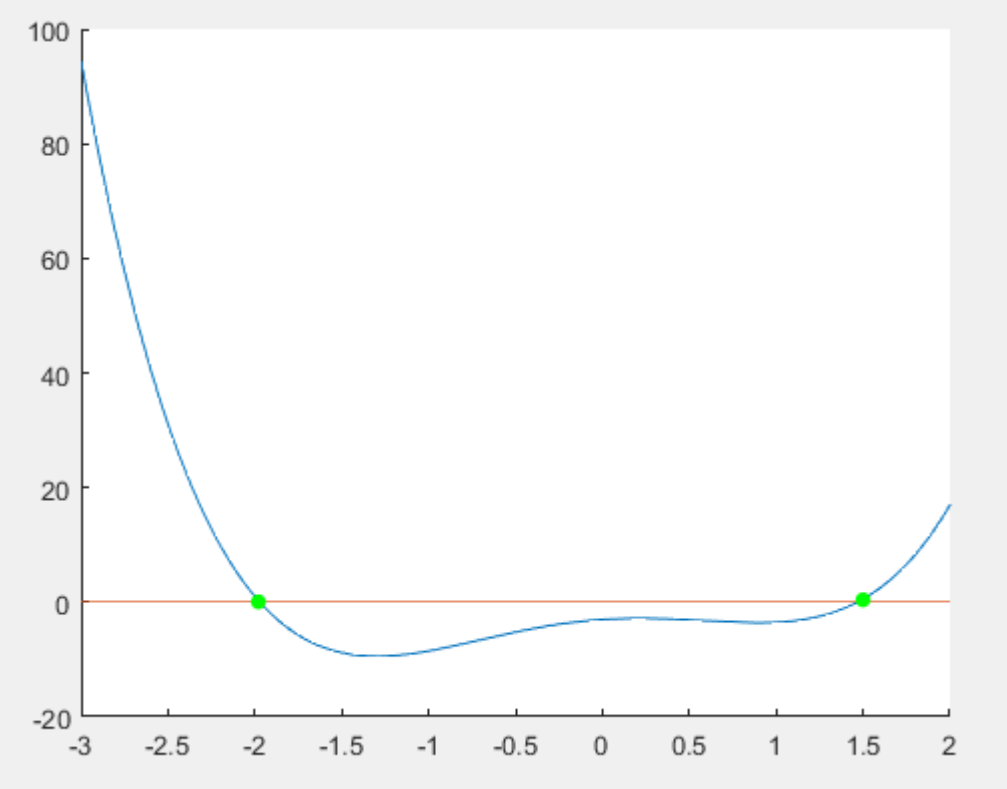
$$x_{k+1} = x_k - [f'(x_k)]^{-1} f(x_k)$$

Otrzymujemy:

$$\boldsymbol{f'(x_k)\Delta x_k = -f(x_k)}$$

$$\boldsymbol{x_{k+1} = x_k + \Delta x_k}$$

2.2 Wykres funkcji



2.3 Zbieżność metody Newtona

Metoda ta jest zbieżna lokalnie i jest rzędu drugiego. Jest metodą niezapamiętującą wcześniejszych stanów.

Prawdziwa jest dla niej zależność:

$$||x_{k+1} - \hat{x} \ \ || \approx ||x_k - \hat{x}||^2$$

2.4 Kod algorytmu

```
function [ X ] = newton( a,n )
    X=zeros(n,1);
    for i=1:n
        c=a-wielomian(a)/wielomianPochodna(a);
        a=c;
        X(i) = a;
    end
end
```

2.5 Wyniki

Punkt zero w pobliżu x=-2	Punkt zero w pobliżu x=1.5
-1.9722	1.4676
-1.9713	1.4660
-1.9713	1.4660

Korzystając z wykresu wielomianu widać było że mamy dwa pierwiastki (badany był obszar w przedziale max <-20,20>). W pobliżu tych dwóch punktów zastosowano metodę newtona. Jak widać już po drugiej iteracji algorytm zaczął zwracać te same wyniki. Wyznaczone punkty nie są idealnymi zerami funkcji, ale bardzo bliskie zeru.

2.6 Komentarz do wyników

W tym przypadku metoda Newtona bardzo sprawnie rozwiązała nasz problem (szybko zbieżna), jednak należy przyznać że przy szerszcyh przedziałach – punktach bardziej oddalonych od zer będzie mniej. Należy też zauważyć, że metody newtona może nam przemknąć od jednego zera do drugiego jeśli źle dobierzemy punkt początkowy.

Przedstawione metody iteracyjne zawodzą również, gdy funkcja nie zbiega monotonicznie do pierwiastka.

3. Zadanie 2b

Poszukiwanie pierwiastków rzeczywistych i zespolonych wielomianu metodami Mullera MM1 i MM2. Porównanie ich efektywności z algorytmem Newtona.

3.1 Wyliczenie pierwiastków inną metodą

Aby mieć porównanie wyników, za pomocą programu http://www.wolframalpha.com/ wyliczono pierwiastki rzeczywiste i zespolone:

[-1.97132 ; 1.46604 ; 0.127637 – 0.709036i ; 0.127637 – 0.709036i]

3.2 Metoda Mullera MM1

3.2.1 MM1 algorytm

Metoda ta aproksymuje wielomian w otoczeniu poszukiwanego pierwiastka rozwiązania. Jest ono podobne do metody siecznych. Do metody potrzebne są nam trzy punkty x_0, x_1, x_2 wraz z wartościami funkcji $f(x_0), f(x_1), f(x_2)$, które będziemy interpolować do funkcji kwadratowej. Mając funkcję kwadratową można wyznaczyć jej pierwiastki i jeden z nich będzie naszym k-tym przybliżeniem rozwiązania funkcji wyjściowej.

Niech x_2 będzie aktualnym przybliżeniem pierwiastka wielomianu. Zdefiniujmy sobie zmienną przyrostową i dokonajmy przekształceń:

$$z = x - x_2,$$

$$z_0 = x_0 - x_2,$$

$$z_1 = x_1 - x_2$$

$y(z) = az^2 + bz + c,$

Mając 3 punkty i wzór poszukiwanej paraboli to aby wyliczyć a,b można zastosować wzory:

$$az_0^2 + bz_0 = f(x_0) - f(x_2)$$

$az_1^2 + bz_2 = f(x_1) - f(x_2),$

Obliczamy pierwiastki równania z dobrze znanych wzorów na pierwiastki kwadratowe:

$$z_+ = \frac{-2c}{b+\sqrt{b^2-4ac}}, z_- = \frac{-2c}{b-\sqrt{b^2-4ac}}$$

Przybliżenie:

$$x_3 = x_2 + z_{min}, \text{gdzie}$$

$$z_{min} = z_+, \text{gdy } |b + \sqrt{b^2 - 4ac}| \geq |b - \sqrt{b^2 - 4ac}|$$

$$z_{min} = z_+, w\ p.p.$$

Z trzech punktów x_0, x_1, x_2 odrzucamy ten najbardziej oddalony od ostatnio wyznaczonego przybliżenia.

W sposób ręczny zostały wyznaczone wzory na współczynniki a i b:

$$a = \frac{f(x_1) - f(x_2)}{z_1^2 - z_0 z_1} - \frac{f(x_0) - f(x_1)}{z_1 - z_0^2}, b = \frac{f(x_0) - f(x_2) - az_0^2}{z_0}$$

3.2.2 MM1 kod

```
function [ X, Y ] = MM1( x0,x1,x2,n )
    Y=zeros(n,1);
    X=zeros(n,1);
    for i=1:n
        fx0 = wielomian(x0);
        fx1 = wielomian(x1);
        fx2 = wielomian(x2);
        z0 = x0 - x2;
        z1 = x1 - x2;
        a = (fx1-fx2)/(z1^2-z0*z1) - (fx0-fx1)/(z1-z0^2);
        b = (fx0-fx2-a*z0^2) / z0;
        c = fx2;
        zp = -2*c / ( b + sqrt(b^2 - 4*a*c) );
        zm = -2*c / ( b - sqrt(b^2 - 4*a*c) );
        if( abs(b + sqrt(b^2 - 4*a*c)) >= abs(b - sqrt(b^2 -
4*a*c)) )
            zmin = zp;
        else
            zmin = zm;
        end
        x2 = x2 + zmin;
        X(i) = x2;
        Y(i) = wielomian(X(i));
    end
end
```

3.3 Metoda Mullera MM2

3.3.1 Algorytm MM2

Metoda ta wykorzystuje wartości wielomianu i jego pochodnych (pierwszego i drugiego rzędu) w aktualnym punkcie.

Wykorzystujemy definicję paraboli y(z) w punkcie z=0:

$$y(0) = c = f(x_k)$$

$$y'(0) = b = f'(x_k)$$

$$y''(0) = 2a = f''(x_k)$$

Więc jeśli $z = x - x_k$ to możemy wyprowadzić wzór na pierwiastki wielomianu:

$$z_{\pm} = \frac{-2f(x_k)}{f'(x_k) \pm \sqrt{\left(f'(x_k)\right)^2 - 2f(x_k)f''(x_k)}}$$

A w następnej iteracji do przybliżenia pierwiastka używamy pierwiastka paraboli o mniejszym module:

$$x_{k+1} = x_k + z_{min}$$

Metoda MM2 tak samo jak MM1 może być wykorzystywana do poszukiwań pierwiastków zespolonych. Jest ona zbieżna lokalnie. Powinna być efektywniejsza od metody siecznych, troche wolniejsza od metody Newtona.

3.3.2 Kod algorytmu

```
function [ X, Y ] = MM2( x0,n )
    Y=zeros(n,1);
    X=zeros(n,1);
    xk = x0;
    for i=1:n
        zp = -
2*wielomian(xk)/(wielomianPochodna(xk)+sqrt(wielomianPochodna(xk)^2-
2*wielomian(xk)*wielomianPochodnaDruga(xk)));
        zm = -2*wielomian(xk)/(wielomianPochodna(xk)-
sqrt(wielomianPochodna(xk)^2-
2*wielomian(xk)*wielomianPochodnaDruga(xk)));
        if abs(wielomianPochodna(xk)+sqrt(wielomianPochodna(xk)^2-
2*wielomian(xk)*wielomianPochodnaDruga(xk))) > abs(wielomianPochodna(xk)-
sqrt(wielomianPochodna(xk)^2 -
2*wielomian(xk)*wielomianPochodnaDruga(xk)))
            zmin = zp;
        else
            zmin = zm;
        end
        xk = xk+zmin;
        X(i) = xk;
        Y(i) = wielomian(X(i));
    end
end
```

3.4 Wyniki

MM1		MM1		MM1	
x0 = -2.5, x1 = -2, x2 = -1.5		x0 = 1, x1 = 1.5, x2 = 2		x0=-0.5, x1=0, x2=0.5	
X(i)	Y(i)	X(i)	Y(i)	X(i)	Y(I)
-1.6572	-7.2373	1.7489	6.5914	0.6220 - 1.3333i	0.6220 - 1.3333i
-1.7788	-5.1692	1.5781	2.0727	0.2430 - 1.3123i	0.2430 - 1.3123i
-1.8651	-3.1664	1.4910	0.4076	-0.0761 - 1.2216i	-0.0761 - 1.2216i
-1.9203	-1.6223	1.4651	-0.0143	...	...
-1.9510	-0.6687	1.4663	0.0038	0.1276 - 0.7091i	0.0002 - 0.0001i
-1.9649	-0.2162	1.4660	-0.0009	0.1276 - 0.7090i	0.0001 - 0.0002i
-1.9696	-0.0576	1.4661	0.0002	0.1276 - 0.7090i	-0.0001 - 0.0002i
-1.9709	-0.0140	1.4660	-0.0001	0.1276 - 0.7090i	-0.0002 - 0.0001i
-1.9712	-0.0033	1.4660	0.0000	0.1276 - 0.7090i	-0.0002 + 0.0001i
-1.9713	-0.0008	1.4660	-0.0000	0.1277 - 0.7090i	-0.0000 + 0.0002i

MM2		MM2		MM2	
x0=-4		x0 =4		x0=-1	
X(i)	Y(i)	X(i)	Y(i)	X(i)	Y(i)
-2.7680 - 0.7945i	8.4909 +91.2259i	2.7098 - 0.8750i	0.1793 - 1.1211i	-0.1601 + 0.0000i	-3.4489 + 0.0000i
-2.2927 - 0.0313i	15.3143 + 1.9904i	2.1273 - 0.0129i	0.2439 - 0.0083i	0.2055 + 0.7521i	0.2145 - 0.8533i
-1.9516 + 0.0074i	-0.6525 - 0.2372i	1.5190 + 0.2872i	-0.0114 + 0.0496i	0.1277 + 0.7095i	0.0040 - 0.0017i
-1.9713 - 0.0000i	0.0001 + 0.0001i	1.4724 + 0.0154i	0.0010 + 0.0025i	0.1276 + 0.7090i	-0.0000 + 0.0000i
-1.9713 - 0.0000i	-0.0000 + 0.0000i	1.4660 + 0.0000i	0.0000 + 0.0000i	0.1276 + 0.7090i	-0.0000 + 0.0000i

-1.9713 + 0.0000i	0.0000 + 0.0000i	1.4660 - 0.0000i	-0.0000 - 0.0000i	0.1276 + 0.7090i	-0.0000 + 0.0000i
-1.9713 + 0.0000i	0.0000 + 0.0000i	1.4660 + 0.0000i	0.0000 + 0.0000i	0.1276 + 0.7090i	-0.0000 + 0.0000i
-1.9713 + 0.0000i	0.0000 + 0.0000i	1.4660 + 0.0000i	0.0000 + 0.0000i	0.1276 + 0.7090i	-0.0000 + 0.0000i
-1.9713 + 0.0000i	0.0000 + 0.0000i	1.4660 + 0.0000i	0.0000 + 0.0000i	0.1276 + 0.7090i	-0.0000 + 0.0000i
-1.9713 + 0.0000i	0.0000 + 0.0000i	1.4660 + 0.0000i	0.0000 + 0.0000i	0.1276 + 0.7090i	-0.0000 + 0.0000i

3.5 Komentarz do wyników

Metoda MM1 jest bardzo wolno zbieżna – należało wykonać 150 iteracji aby uzyskać pierwiastek zespolony. Dla pierwiastków rzeczywistych prędkość była normalna: ~10 iteracji wystarczy.

Metoda MM2 jest bardzo szybko zbieżna. Przy punkcie początkowym bliskim pierwiastkowi metoda potrafiła podać poprawny wynik już po dwóch iteracjach. Dlatego w ramach projektu używano oddalonych punktów by lepiej przetestować metodę. Wciąż jest szybsza od metody Newtona.