Projekt 3. [MNUM]

Bartosz Rajkowski, grupa 2AR, wtorek 16:00
3.37

1. Zadanie 1.

1.1. Treść zadania:

Znaleźć wszystkie zera funkcji

$$f(x) = 7.2 - 2x - 5e^{-x}$$

w przedziale [-5,10], używając dla każdego zera programu z implementacją:

a) metody bisekcji,

b) metody siecznych,

c) metody Newtona.

1.2. Algorytmy:

a) metoda bisekcji

Metoda bisekcji polega na połowieniu aktualnego przedziału izolacji pierwiastka w kolejnych iteracjach.

Startujemy z początkowego przedziału [*a,b*] = [a_0,b_0] izolacji pierwiastka.

W *metodzie bisekcji*, w każdej n-tej iteracji:

1. Bieżący przedział zawierający zero funkcji [a_n,b_n] jest dzielony na dwie połowy, punktem środkowym c_n i obliczana jest wartość funkcji w nowym punkcie.
2. Obliczane są iloczyny $f(a_n) * f(c_n)\ i\ f(c) * f(b_n)$, nowy przedział zawierający pierwiastek wybierany jest jako ten z dwóch podprzedziałów, któremu odpowiada iloczyn ujemny.

Procedura jest powtarzana tak długo, aż np. $f(c_n) < \delta$, gdzie δ to założona dokładność. Dla precyzji można również sprawdzać także długość przedziału a, b. Dokładność rozwiązania zależy jedynie od ilości wykonanych iteracji, a nie zależy od dokładności obliczania wartości funkcji f(x) na krańcach kolejnych przedziałów izolacji pierwiastka. Metoda jest zbieżna liniowo (p=1) z ilorazem zbieżności k=0.5.

b) metoda siecznych

Metoda siecznych również polega na wyznaczeniu siecznej pomiędzy dwoma ostatnio wyznaczonymi punktami. Jeśli dwa ostatnie punkty oznaczymy przez $x_{n-1}\ i\ x_n$, to nowy punkt w metodzie siecznych jest zdefiniowany wzorem:

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

Metoda jest szybciej zbieżna niż metoda bisekcji (p≈1.618). Jednak jest ona zbieżna jedynie lokalnie, sąd w praktyce może być niezbieżna – jeśli początkowy przedział izolacji pierwiastka nie jest dostatecznie mały.

c) metoda Newtona

Metoda Newtona, zwana też metodą stycznych, zakłada aproksymację funkcji jej liniowym przybliżeniem wynikającym z uciętego rozwinięcia w szereg Taylora w aktualnym punkcie x_n (aktualnym przybliżeniem pierwiastka).

$$f(x) \approx f(x_n) + f'(x_n)(x - x_n)$$

Następny punkt x_{n+1}wynika z przyrównania do zera sformułowanej lokalnej liniowej aproksymacji funkcji $f(x)$, tzn. z równania

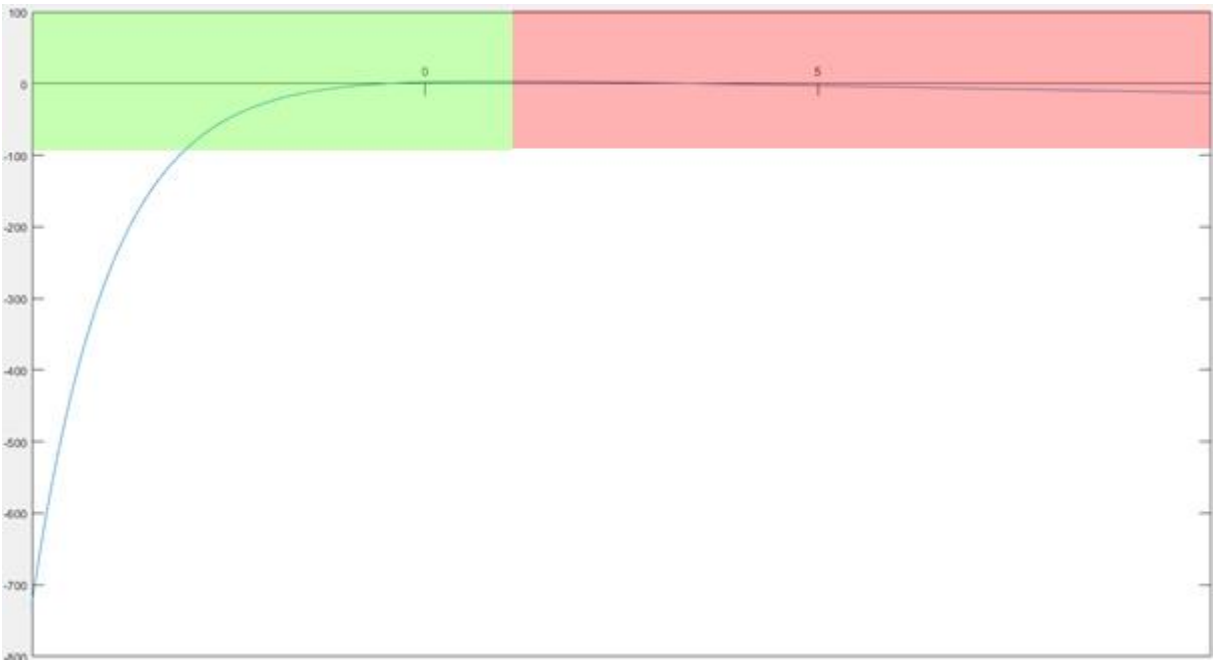
$$f(x_n) + f'^{(x_n)}(x - x_n) = 0$$

co prowadzi do zależności:

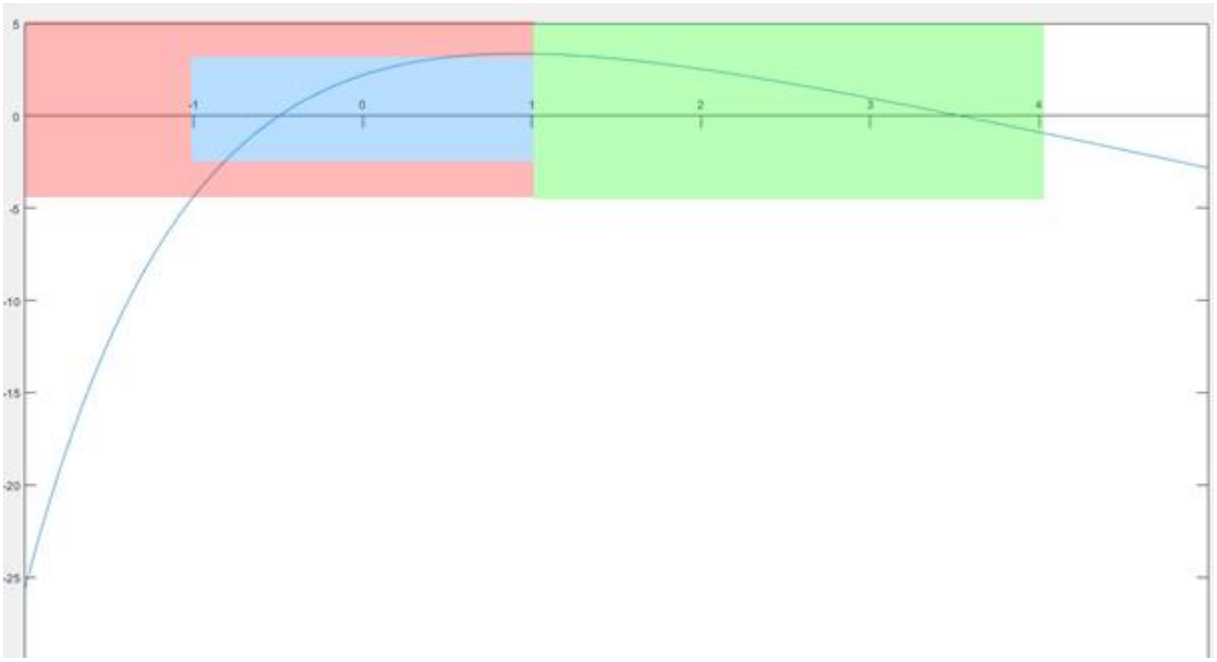
$$x_{n+1} = x_n - \frac{f(x_n)}{f'^{(x_n)}}$$

Metoda jest zbieżna lokalnie. Jeżeli zastosujemy ją w punkcie zbytnio oddalonym od rozwiązania, to może być ona rozbieżna. Natomiast metoda Newtona jest lokalnie bardzo szybka, jej zbieżność jest kwadratowa (p=2).

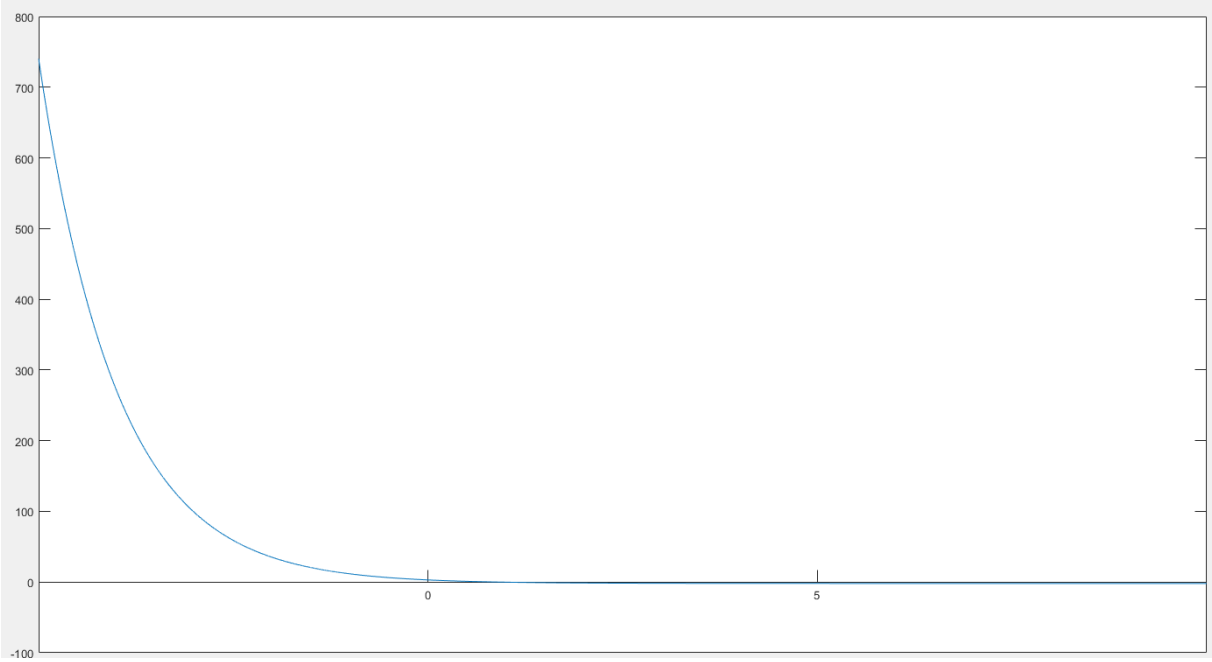
1.3. Wykres analizowanej funkcji oraz jej pochodnej:



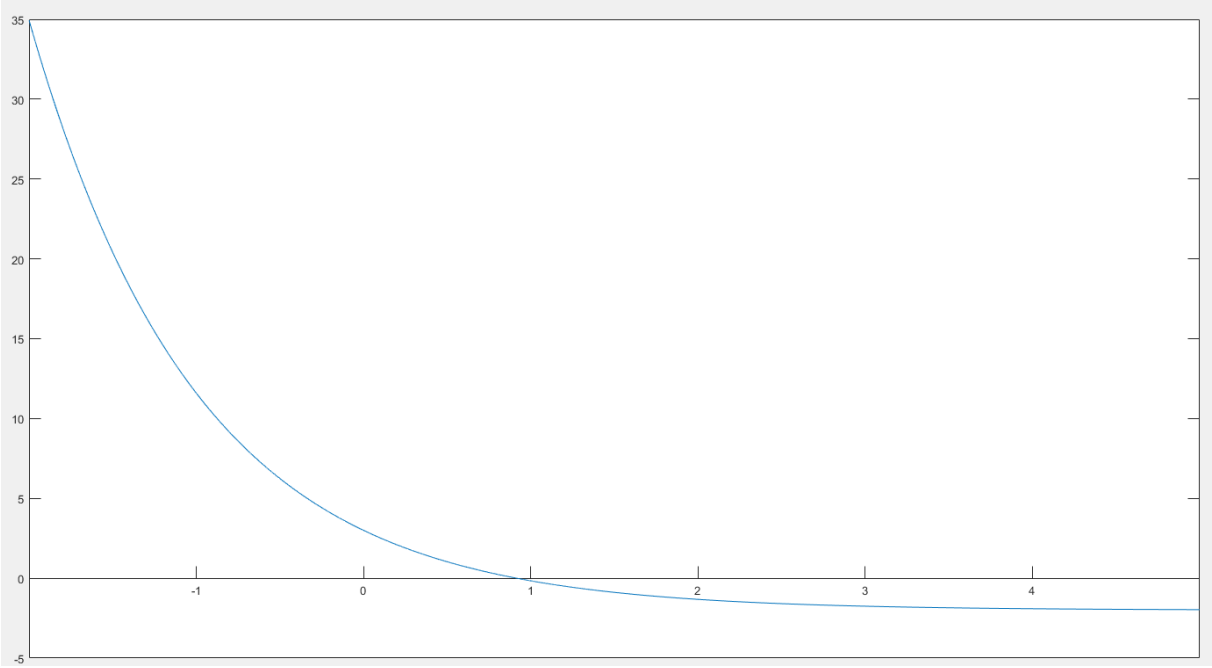
wykres funkcji w przedziale[-5;10] z zaznaczonymi przedziałami startowymi



wykres funkcji w przedziale [-2;5]] z zaznaczonymi przedziałami startowymi



wykres funkcji pochodnej w przedziale [-5;10]



wykres funkcji pochodnej w przedziale [-2;5]

1.4. Kod głównego programu:

```
function [Wynik, iteracje] = miejsce_zerowe(a,b,max_iter,epsilon,metoda)
%wyznaczanie miejsca zerowego, w przedziale a,b
%z metodą podaną w zmiennej metoda
%z podaną maksymalną ilością iteracji max_iter
%z zadaną dokładnością epsilon

if funkcja_zad1(a)*funkcja_zad1(b)>0
    error('zly przedzial - f(a)*f(b)>0');
end
iteracje =0; %licznik iteracji
Wynik=zeros(max_iter,2); %alokacja pamięci
for i=1:max_iter
    iteracje=iteracje+1;
    switch metoda %wybór metody
        case 'bisekcja'
            Wynik(iteracje,1)=(a+b)*0.5; %wyznaczenie punktu środkowego i
                                         %zapisanie w tablicy
            Wynik(iteracje,2)=funkcja_zad1(Wynik(iteracje,1)); %wyznaczenie
                                                    %wartości w punkcie
            if(funkcja_zad1(a)*Wynik(iteracje,2)<0) %decyzja o nowym
                                          %przedziale na podstawie iloczynu
               b=Wynik(iteracje,1);
            else
               a=Wynik(iteracje,1);
            end
            end

        case 'sieczne'
            %wyznaczanie nowego punktu
            c=b-((funkcja_zad1(b)*(b-a))/(funkcja_zad1(b)-
funkcja_zad1(a))); %wyznaczanie
            %zastąpnie poprzednich wartości kolejnymi
            a=b;
            b=c;
            %decyzja którą wartośc wybieramy jako wynik i zapisujemy ją w
            %tablicy
            if abs(funkcja_zad1(a))<abs(funkcja_zad1(b))
                 Wynik(iteracje,1)=a;
            else
                Wynik(iteracje,1)=b;
            end
            Wynik(iteracje,2)=funkcja_zad1(Wynik(iteracje,1));

            end
        case 'Newton'
            %wyznaczanie przecięcia stycznej w punkcie a
            %i zapisanie jej do tablicy
            Wynik(iteracje,1)=a-funkcja_zad1(a)/funkcja_poch_zad1(a);
            %wyliczenie wartości w tym punkcie
            Wynik(iteracje,2)=funkcja_zad1(Wynik(iteracje,1));
            %podstawienie nowego punktu
            a=Wynik(iteracje,1);
        otherwise
            error('Błędna metoda');
    end
    %sprawdzenie warunku końcowego
    if abs(Wynik(iteracje,2))<epsilon
        break;
    end
end
Wynik=Wynik(1:iteracje,:);%przycięcie tablicy wyników

end
```

1.5. Wynik działania:

Pierwiastki wyznaczone analitycznie: -0.492983 , 3.52648

Za punkt startowy w metodzie Newtona lewy brzeg przedziału wywołania.

Przedział : [-5;1], maksymalna ilość iteracji: 15, dokładność 10^{-12}

iteracja	bisekcja		metoda siecznych		metoda Newtona	
	X	Y	X	Y	X	Y
1	-2	-25,7453	1	3,360603	-4,02054	-263,415
2	-0,5	-0,04361	0,972311	3,364338	-3,0684	-94,2005
3	0,25	2,805996	2,720769	1,429341	-2,17582	-32,4954
4	-0,125	1,784258	3,440547	0,15867	-1,40299	-10,3307
5	-0,3125	0,99081	3,530427	-0,00732	-0,8396	-2,69797
6	-0,40625	0,506611	3,526465	2,57E-05	-0,55789	-0,41912
7	-0,45313	0,240146	3,526479	4,02E-09	-0,49566	-0,01657
8	-0,47656	0,100482	3,526479	-2,03E-15	-0,49299	-2,92E-05
9	-0,48828	0,028997	-	-	-0,49298	-9,12E-11
10	-0,49414	-0,00716	-	-	-0,49298	0
11	-0,49121	0,010952	-	-	-	-
12	-0,49268	0,001903	-	-	-	-
13	-0,49341	-0,00263	-	-	-	-
14	-0,49304	-0,00036	-	-	-	-
15	-0,49286	0,00077	-	-	-	-

Przedział : [-2;1], maksymalna ilość iteracji: 15, dokładność 10^{-12}

iteracja	bisekcja		metoda siecznych		metoda Newtona	
	X	Y	X	Y	X	Y
1	-0,5	-0,04361	0,653616	3,291961	-1,26327	-7,95828
2	0,25	2,805996	0,653616	3,291961	-0,75588	-1,93567
3	-0,125	1,784258	0,653615	3,29196	-0,53204	-0,24791
4	-0,3125	0,99081	0,653614	3,291959	-0,49397	-0,00609
5	-0,40625	0,506611	0,653614	3,291959	-0,49298	-3,96E-06
6	-0,45313	0,240146	0,624031	3,273036	-0,49298	-1,68E-12
7	-0,47656	0,100482	0,594777	3,252018	-0,49298	0
8	-0,48828	0,028997	0,594777	3,252018	-	-
9	-0,49414	-0,00716	0,53423	3,200938	-	-
10	-0,49121	0,010952	0,475419	3,14104	-	-
11	-0,49268	0,001903	0,475419	3,14104	-	-
12	-0,49341	-0,00263	0,310178	2,913062	-	-
13	-0,49304	-0,00036	0,164575	2,62958	-	-
14	-0,49286	0,00077	0,164575	2,62958	-	-
15	-0,49286	0,00077	-0,21213	1,442715	-	-

Przedział : [-1;1], maksymalna ilość iteracji: 15, dokładność 10^{-12}

iteracja	bisekcja		metoda siecznych		metoda Newtona	
	X	Y	X	Y	X	Y
1	0	2,2	0,132973	2,55661	-0,62115	-0,86303
2	-0,5	-0,04361	0,132973	2,55661	-0,50301	-0,06245
3	-0,25	1,279873	0,01364	2,240458	-0,49305	-0,00041
4	-0,375	0,675043	-0,08695	1,919698	-0,49298	-1,79E-08
5	-0,4375	3,31E-01	-0,68895	-1,38E+00	-0,49298	-1,78E-15
6	-0,46875	1,48E-01	-0,43716	3,33E-01	-	-
7	-0,48438	0,052949	-0,48608	0,042529	-	-
8	-0,49219	0,004921	-0,49324	-0,00161	-	-
9	-0,49609	-0,01928	-0,49298	7,38E-06	-	-
10	-0,49414	-0,00716	-0,49298	1,27E-09	-	-
11	-0,49316	-0,00112	-0,49298	-1,78E-15	-	-
12	-0,49268	0,001903	-	-	-	-
13	-0,49292	0,000393	-	-	-	-
14	-0,49304	-0,00036	-	-	-	-
15	-0,49298	1,52E-05	-	-	-	-

Przedział : [1;10], maksymalna ilość iteracji: 15, dokładność 10^{-12}

iteracja	bisekcja		metoda siecznych		metoda Newtona	
	X	Y	X	Y	X	Y
1	5,5	-3,82043	2,871527	1,173884	21,92493	-36,6499
2	3,25	0,506129	3,470349	0,103772	3,6	-0,13662
3	4,375	-1,61E+00	3,528418	-0,00359	3,526682	-0,00038
4	3,8125	-5,35E-01	3,526475	8,17E-06	3,526479	-3,03E-09
5	3,53125	-8,84E-03	3,526479	6,28E-10	3,526479	-3,89E-16
6	3,390625	0,250312	3,526479	-3,89E-16	-	-
7	3,460938	0,121123	-	-	-	-
8	3,496094	5,62E-02	-	-	-	-
9	3,513672	2,37E-02	-	-	-	-
10	3,522461	7,44E-03	-	-	-	-
11	3,526855	-0,0007	-	-	-	-
12	3,524658	0,003374	-	-	-	-
13	3,525757	0,001339	-	-	-	-
14	3,526306	3,21E-04	-	-	-	-
15	3,526581	-0,00019	-	-	-	-

Przedział : [1;5], maksymalna ilość iteracji: 15, dokładność 10^{-12}

iteracja	bisekcja		metoda siecznych		metoda Newtona	
	X	Y	X	Y	X	Y
1	3	0,951065	3,170129	0,649752	21,92493	-36,6499
2	4	-0,89158	3,511447	0,027838	3,6	-0,13662
3	3,5	0,049013	3,526725	-0,00045	3,526682	-0,00038
4	3,75	-4,18E-01	3,526479	2,73E-07	3,526479	-3,03E-09
5	3,625	-1,83E-01	3,526479	2,66E-12	3,526479	-3,89E-16
6	3,5625	-6,68E-02	3,526479	-3,89E-16	-	-
7	3,53125	-0,00884	-	-	-	-
8	3,515625	0,020104	-	-	-	-
9	3,523438	5,64E-03	-	-	-	-
10	3,527344	-1,60E-03	-	-	-	-
11	3,525391	2,02E-03	-	-	-	-
12	3,526367	0,000208	-	-	-	-
13	3,526855	-0,0007	-	-	-	-
14	3,526611	-0,00024	-	-	-	-
15	3,526489	-1,84E-05	-	-	-	-

1.6. Wnioski:

Otrzymane wyniki są zgodne z przewidywaniami teoretycznymi. Metoda Newtona daje wynik najszybciej ze wszystkich metod, jednak w dwóch przedziałach w jednej iteracji wyrzuciła nas poza zadany przedział. Działo się to w miejscu gdzie pochodna była bliska zeru, jednak później udało się wrócić w obszar poszukiwań. Metoda siecznych działa nie dużo gorzej, jednak ma tendencję do pomijania miejsca zerowego, gdy zadany przedział nie jest dostatecznie mały – zostaje wyliczony pierwiastek spoza zadanego przedziału lub wynik nie zostaje odnaleziony. Natomiast metoda bisekcji działa wolno, ale stabilnie. Pierwiastki zostają zawsze odnalezione – dokładność rozwiązania jest związana jedynie z ilością iteracji. Zmniejszanie początkowego obszaru poszukiwań poprawia dokładność rozwiązania, jednak taki sam efekt można uzyskać zwiększając ilość iteracji. Gdy funkcja nie zbiega monotonicznie do pierwiastka metody zawodzą.

2. Zadanie 2.

2.1. Treść zadania:

Używając metody Müllera MM2 znaleźć wszystkie pierwiastki rzeczywiste i zespolone wielomianu:

$$f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0, [a_4\ a_3\ a_2\ a_1\ a_0] = [-1, -2\ 5\ 2\ 1]$$

2.2. Algorytmy:

Metoda Müllera polega na aproksymacji wielomianu w otoczeniu rozwiązania funkcją kwadratową. Może być traktowana jako uogólnienie metod siecznych – zamiast interpolacji w dwóch punktach funkcją liniową wykonujemy interpolację w trzech punktach funkcją kwadratową. Metoda MM2 to realizacja oparta na wykorzystaniu informacji o wielomianie jedynie w jednym punkcie, tzn. wykorzystująca do wyznaczenia funkcji kwadratowej wartości wielomianu i jego pierwszej i drugiej pochodnej w aktualnym punkcie. Wersja MM2 jest efektywniejsza obliczeniowo z powodu, że obliczanie wartości wielomianu w k+1 punktach jest kosztowniejsze niż obliczenie wartości wielomianu i jego k kolejnych pochodnych w jednym punkcie.

Z definicji paraboli y(z) dla z=x-x_k w punkcie z=0 wynika:

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

co prowadzi do wzoru:

$$z_{+,-} = \frac{-2f(x_k)}{f\prime(x_k) \pm \sqrt{f'(x_k)^2 - 2f(x_k)f\prime\prime(x_k)}}$$

Do przybliżenia zera α bierzemy pierwiastek paraboli o mniejszym module:

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

Oprócz metody Müllera w zadaniu wykorzystałem deflację czynnikiem liniowym – po znalezieniu jednego z pierwiastków dzielę aktualny wielomian przez czynnik (x- α), gdzie α jest znalezionym pierwiastkiem. Uzyskany w ten sposób wielomian jest prostszy oraz nie wyznaczymy więcej tego samego pierwiastka. W zadaniu zastosowałem prosty schemat Hornera – jest on bardzo łatwy w realizacji. Daje on błędy numeryczne większe niż sklejany schemat Hornera, jednak w tym zadaniu analizujemy wielomian jedynie 4 stopnia.

2.3. Kod programu:

2.3.1. Obliczanie wartości wielomianu w punkcie x zapisanego w postaci macierzy:

```
function [ y ] = wielomian( x,W )
%obliczanie wartości wielomianu
n=length(W); %stopien
y=W(1); % pierwszy element
for i=2:n
    y=y*x+W(i); %domnażanie kolejnych potęg x i dodawanie współczynników
end
end
```

2.3.2. Obliczanie współczynników pochodnej wielomianu:

```
function [ W2 ] = pochodna(W)
%obliczanie pochodnej wielomianu
n=length(W)-1;%długość nowego wielomianu
if n==0%sprawdzanie czy pochodna nie jest zerowa
W2=0;
else
W2=zeros(1,n);
      for i=1:n
         W2(i)=W(i)*(n-i+1);%wyznaczanie kolejnych wsp.
      end
end
end
```

2.3.3. Schemat Hornera – prosty:

```
function [ W2 ] = schemat_hornera( W,a)
%dzielenie wielomianu przez (x-a)
n=length(W)-1;
W2(1)=W(1);%pierwszy wsp. bez zmian
for i=2:n
    W2(i)=W(i)+W2(i-1)*a;%kolejne wsp. wielomianu
end

end
```

2.3.4. Główny algorytm metody MM2:

```
function [ Wynik ] = mm2(f, x, iter )
df=pochodna(f);%pierwsza pochodna
ddf=pochodna(df);%druga pochodna
Wynik=zeros(iter,1);%alokacja pamięci
for i=1:iter
    %porównywanie modułów
    %wyznaczanie pierwiastkow zgodnie ze wzorem
    if abs(wielomian(x,df)+sqrt(wielomian(x,df)^2-
2*wielomian(x,f)*wielomian(x,ddf)))>abs(wielomian(x,df)-
sqrt(wielomian(x,df)^2-2*wielomian(x,f)*wielomian(x,ddf)))

        z=-2*wielomian(x,f)/(wielomian(x,df)+sqrt(wielomian(x,df)^2-
2*wielomian(x,f)*wielomian(x,ddf)));

    else

        z=-2*wielomian(x,f)/(wielomian(x,df)-sqrt(wielomian(x,df)^2-
2*wielomian(x,f)*wielomian(x,ddf)));

    end
    %zapis wyniku iteracji
    Wynik(i)=x+z;
    %wyznaczanie kolejnej wartosci
    x=x+z;
end
```

2.3.5. Program główny:

```
function [ Wynik ] = zad2()
W=[-1 -2 5 2 1]; %zadany wielomian
n=length(W)-1;%stopień wielomianu
Wynik=zeros(4,1);%alokacja pamięci
pocz=0;%punkt startowy
for i=1:n
   temp=mm2(W,pocz,20); %wyznaczenie pierwiastka
   Wynik(i)=temp(length(temp));%pierwiastek jest ostatnim elementem wetora
                               %wyniku
   pocz=Wynik(i);%nowy punkt startowy
   W=schemat_hornera(W,Wynik(i));%deflacja wielomianu
end
end
```

2.4. Wyniki:

Po wywołaniu głównego programu otrzymujemy wynik:

-0.1962 - 0.3658i
-0.1962 + 0.3658i
1.7357 - 0.0000i
-3.3433 + 0.0000i

Pierwiastki wyznaczone przy pomocy systemu Mathematica:
3.3433
1.7357
-0.19619 ± 0.36583 i

Jak widać pierwiastki zostały wyznaczone identycznie.

Natomiast analizując sam algorytm MM2 dla różnych punktów startowych otrzymujemy wyniki:

x / iteracje	-2	-1	0
1	-1.0000 + 0.0000i	0.0000 + 0.0000i	-0.2000 - 0.4000i
2	0.0000 + 0.0000i	-0.2000 - 0.4000i	-0.1962 - 0.3658i
3	-0.2000 - 0.4000i	-0.1962 - 0.3658i	-0.1962 - 0.3658i
4	-0.1962 - 0.3658i	-0.1962 - 0.3658i	-0.1962 - 0.3658i
5	-0.1962 - 0.3658i	-0.1962 - 0.3658i	-0.1962 - 0.3658i
6	-0.1962 - 0.3658i	-0.1962 - 0.3658i	-0.1962 - 0.3658i
7	-0.1962 - 0.3658i	-0.1962 - 0.3658i	-0.1962 - 0.3658i
8	-0.1962 - 0.3658i	-0.1962 - 0.3658i	-0.1962 - 0.3658i
9	-0.1962 - 0.3658i	-0.1962 - 0.3658i	-0.1962 - 0.3658i
10	-0.1962 - 0.3658i	-0.1962 - 0.3658i	-0.1962 - 0.3658i

x / iteracje	1	2	3	-3
1	0.2857 + 0.0000i	1.7253	2.0299 + 0.5455i	-3.3521
2	-0.4786 - 0.3237i	1.7357	1.8078 + 0.0360i	-3.3433
3	-0.2026 - 0.3643i	1.7357	1.7357 - 0.0002i	-3.3433
4	-0.1962 - 0.3658i	1.7357	1.7357 - 0.0000i	-3.3433
5	-0.1962 - 0.3658i	1.7357	1.7357 + 0.0000i	-3.3433
6	-0.1962 - 0.3658i	1.7357	1.7357 + 0.0000i	-3.3433
7	-0.1962 - 0.3658i	1.7357	1.7357 + 0.0000i	-3.3433
8	-0.1962 - 0.3658i	1.7357	1.7357 + 0.0000i	-3.3433
9	-0.1962 - 0.3658i	1.7357	1.7357 + 0.0000i	-3.3433
10	-0.1962 - 0.3658i	1.7357	1.7357 + 0.0000i	-3.3433

2.5. Wnioski:

Metoda MM2 zawsze znajduje lokalny pierwiastek funkcji. Gdy startujemy z dalszego punktu zajmuje to więcej iteracji. Sama metoda znajduje jeden pierwiastek zespolony. Problematyczny jest tutaj dobór kroku (gęstości próbkowania) – metoda znajduje jedynie najbliższy pierwiastek. Jeżeli pierwiastki będą występowały gęsto metoda staje się niewystarczająca, gdyż znajduje ona tylko najbliższy pierwiastek. Zastosowanie deflacji umożliwia znalezienie wszystkich pierwiastków. Wielokrotne operacje na wielomianach mogą powodować błędy numeryczne, jednak w tym zadaniu nie doprowadziły one to utraty jakości wyników.