MNUM-PROJEKT, zadanie 3.43 Łukasz Świtaj, 283777

Spis treści

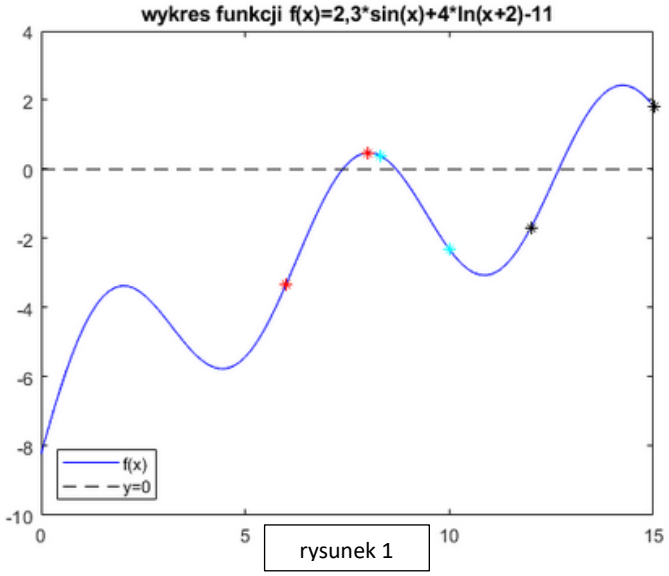
Zadanie 1. **Metody rozwiązywania równań nieliniowych z jedną niewiadomą**

Celem zadania jest napisanie programu obliczającego wszystkie zera funkcji
$f(x) = 2.3*sin(x)+4*ln(x+2)–11$ w przedziale [2, 12].

1. Wybór przedziałów startowych oraz ograniczeń.
2. Implementacja metod (oraz stworzenie warunków, w których minimum jedna z nich zawodzi):
   a. bisekcji
   b. siecznych
   c. stycznych (Newtona)

Koncepcja rozwiązania

1. Początkowo napisana została metoda wspomagająca rysowanie wykresu funkcji (rysunek 1).
   Przedziały zaś zostały dobrane wg. dwóch kryteriów:
   1) maksymalna wielkość
   2) możliwość poprawnego wykonania każdej metody dla zadanej funkcji w zadanym przedziale.



rysunek 1

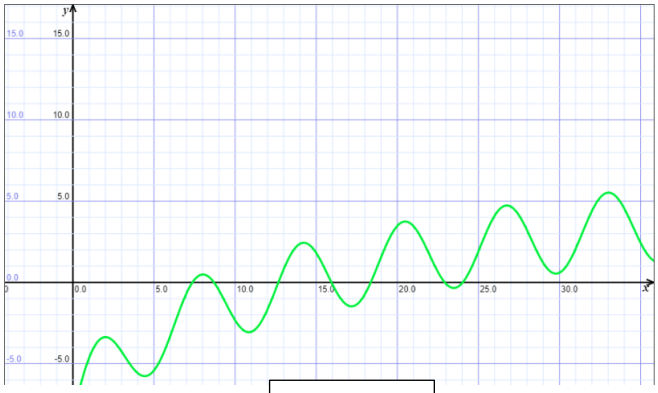
Ostateczne przedziały startowe zostały wybrane *metodą inżynierskiej intuicji* oraz *prób i błędów* jak przedstawiono na rysunku 1 - [6;8], [8.3;10], [12; 15].

Ograniczenia:

- **Każda z metod** została ograniczona parametrem globalnym **`dokladnosc_zer`** określającym maksymalny moduł liczby mogącej być uznawanej w przybliżeniu za równą 0.
- **Metoda bisekcji oraz siecznych** zostały również ograniczone parametrem globalnym **`wielkosc_przedzialu`**, aby zapobiegać przypadkom, w którym dla funkcji o małym nachyleniu pierwiastki zostaną wyznaczone niedokładnie.
- **Metoda stycznych (Newtona)** została ograniczona ze względu na maksymalną liczbę iteracji - **`ilosc_iteracji`**.
- **Metoda stycznych (Newtona)** zostaje przerwana gdy któryś z kolejno wyznaczonych punktów wychodzi poza początkowy przedział – funkcja **`nowy_przedzial_sieczny`** chroni przed pochodną o zbyt małym nachyleniu zgłaszając błąd w niedozwolonym przypadku.

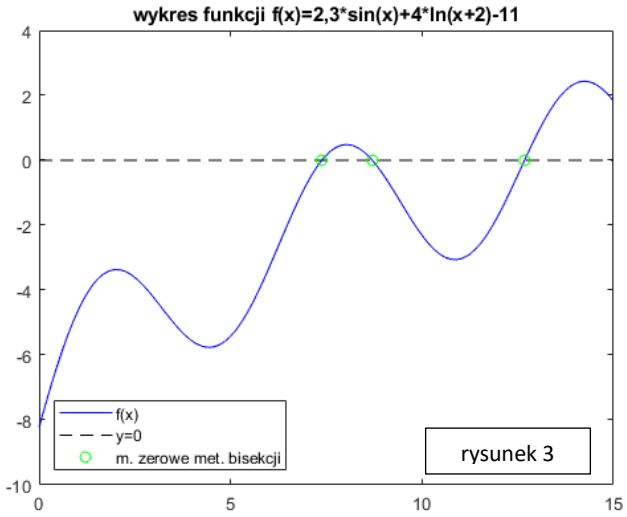
Sprawdzenie

1. Do sprawdzenia poprawności wyznaczonego wykresu użyty został generator wykresów ze strony: matemaks.pl/program-do-rysowania-wykresow-funkcji.html
   Wygenerowany został rysunek 2 pokrywający się z rysunkiem 1.



rysunek 2

2. Z kolei sprawdzenie poprawności wyznaczonych miejsc zerowych następowało na dwóch drogach – analityczne wyliczenie za pomocą funkcji `wartość_funkcji(x)` oraz graficznie- wszystkie 3 metody wygenerowały wykres z rysunku 3 co potwierdza poprawność ich działania.



rysunek 3

Komentarz

a. W celu zautomatyzowania wyboru przedziałów startowych dla metody bisekcji oraz siecznych można napisać funkcję **`wybierz_przedziały`** działającą wg. listy kroków:
   1) Podziel badany przedział na 100 części – małych przedziałów. Przedziały te powinny nachodzić na siebie z dokładnością do epsilona, tak aby nie było przypadku, w którym jeden kończy się na miejscu zerowym, a drugi na nim zaczyna (przypadek nieobsłużony przez metody).
   2) Dla każdego małego przedziału przeprowadź test **`sprawdzenie_przedziału`** sprawdzający warunek f(x1)*f(x2)<0.
   3) W przypadku spełnienia warunku dopisz go do wektora przedziałów, w których znajduje się pierwiastek funkcji.
   4) Na podstawie utworzonego wektora oblicz miejsca zerowe.

W projekcie metoda ta nie została zaimplementowana ze względu na wymóg doboru szerokich przedziałów startowych.

Zadanie 1a. **Metoda bisekcji**

Koncepcja rozwiązania

Zaimplementowana została klasyczna metoda bisekcji:

1. Z przedziału [x1,x2] wyznacz punkt c = (x1+x2)/2.
2. Jeśli c jest miejscem zerowym oraz przedział jest odpowiednio mały zwróć c.
3. Jeśli nie za pomocą metody `sprawdź_przedział` wybierz [x1,c] lub [c,x2], w którym znajduje się miejsce zerowe.
4. Powróć do kroku i.

Sprawdzenie

Dla parametrów `dokladnosc_zer=0.001` oraz `wielkosc_przedzialu=0.1`.

```
metoda bisekcji przedzial nr 1 ([6;8])
x=7.000000 y=-0.700033
x=7.500000 y=0.162567
x=7.250000 y=-0.208420
x=7.375000 y=-0.006645
x=7.437500 y=0.082156
x=7.406250 y=0.038790
x=7.390625 y=0.016329
x=7.382813 y=0.004906
x=7.378906 y=-0.000853
metoda bisekcji przedzial nr 2 ([8.3;10])
x=9.150000 y=-0.730176
x=8.725000 y=-0.028380
x=8.512500 y=0.229330
x=8.618750 y=0.110034
x=8.671875 y=0.043095
x=8.698438 y=0.007909
x=8.711719 y=-0.010100
x=8.705078 y=-0.001061
x=8.701758 y=0.003432
x=8.703418 y=0.001187
x=8.704248 y=0.000064
metoda bisekcji przedzial nr 3 ([12;15])
x=13.500000 y=1.812064
x=12.750000 y=0.184950
x=12.375000 y=-0.775508
x=12.562500 y=-0.295103
x=12.656250 y=-0.054088
x=12.703125 y=0.065796
x=12.679688 y=0.005930
x=12.667969 y=-0.024062
x=12.673828 y=-0.009061
x=12.676758 y=-0.001564
x=12.678223 y=0.002183
x=12.677490 y=0.000310
```

Komentarz

a. Metoda bisekcji działa zgodnie z oczekiwaniami – w każdym kroku widać, że o połowę zmniejsza zadany w danej iteracji przedział. Co ciekawe jest to metoda, która w każdym kolejnym kroku nie zawsze zmniejsza wartość bezwzględną y na co przykładem jest przedział nr 1.
   W zadanych warunkach **średnia ilość iteracji** wynosiła **10,66**.

Warunki, w których metoda nie działa

a. Metoda bisekcji jest najprostszą metodą, która może najłatwiej zawieźć przez błąd programisty – zły dobór przedziału startowego (niespełniający warunku f(x1)*f(x2)<0. Gdy uruchomiłem program właśnie dla takiego przedziału zapętlił on po ok. 20 iteracjach jeden wynik.
   Przypuszczałem, że metoda bisekcji może zawieźć gdy przedziałem startowym będzie `[x1 pierwiastek_funkcji]` zaś `dokładność_zer` będzie mała. Uruchomiłem więc następującą funkcję:

```
x_bisekcja = bisekcja([0 7.3794839], 0.000001, wielkosc_przedzialu)
```

   output

```
ilosc iteracji 24 x_zero 7.379483e+00f, wartosc -3.645547e-07f
```

   Jak widać metoda bisekcji nawet w takim przypadku szybko zbiega do poprawnego wyniku.

Zadanie 1b. **Metoda siecznych**

Koncepcja rozwiązania

Zaimplementowana została metoda siecznych w postaci:

1. Na podstawie przedziału [x1,x2] wyznacz funkcję liniową zawierającą te punkty.
2. Wyznacz miejsce zerowe tej funkcji korzystając z wzoru:

```
c = (x2*y1-x1*y2)/(y1-y2);
```

3. Jeśli c jest miejscem zerowym oraz przedział jest odpowiednio mały zwróć c.
4. Jeśli nie za pomocą metody `sprawdź_przedział` wybierz [x1,c] lub [c,x2], w którym znajduje się miejsce zerowe.
5. Powróć do kroku i.

Sprawdzenie

Dla parametrów `dokladnosc_zer=0.001` oraz `wielkosc_przedzialu=0.1`.

```
metoda siecznych przedzial nr 1 ([6;8])
x=7.745004 y=0.393375
x=7.560390 y=0.232099
x=7.294705 y=-0.132650
x=7.391328 y=0.017351
x=7.380152 y=0.000986
metoda siecznych przedzial nr 2 ([8.3;10])
x=8.552678 y=0.186585
x=8.769956 y=-0.092217
x=8.698089 y=0.008377
x=8.704074 y=0.000299
metoda siecznych przedzial nr 3 ([12;15])
x=13.435563 y=1.703456
x=12.712354 y=0.089322
x=12.672334 y=-0.012885
x=12.677379 y=0.000026
```

Komentarz

b. Metoda siecznych w każdym kolejnym kroku zmniejsza wartość bezwzględną y na co przykładem jest przedział nr 1. Przy zadanych warunkach zmiany te są bardzo widoczne – od 2 do 10 razy.
   W zadanych warunkach **średnia ilość iteracji** wynosiła **4,33**. Czyli 2,46 raza szybciej niż metoda bisekcji co jest wartością większą niż ta teoretyczna (rząd zbieżności 1,618 raza większy od metody bisekcji).

Warunki, w których metoda nie działa

b. Nie udało mi się wpaść na pomysł jak w podanej implementacji metoda siecznych może zawieźć.

Zadanie 1c. **Metoda stycznych (Newtona)**

Koncepcja rozwiązania

Zaimplementowana została klasyczna metoda bisekcji:

1. Zapamiętaj przedział [x1 x2] jako przedział początkowy.
2. Poprowadź prostą *n* styczną do punktu x1. (m = f'(x1))
3. Wyznacz miejsce zerowe (x0) prostej *n* z wzoru:
   ```
   x0 = (m*x1-y1)/m;
   ```
4. Sprawdź czy x0 należy do przedziału początkowego.
5. Jeśli nie – przerwij wykonywanie algorytmu i poinformuj o źle dobranym przedziale.
6. Wybierz nowy przedział [x0 x2].
7. Sprawdź czy osiągnięto maksymalną ilość iteracji – jeśli tak to przerwij.
8. Powróć do kroku 2.

Sprawdzenie

Dla parametrów `dokladnosc_zer=0.001` oraz `wielkosc_przedzialu=0.1`.

```
metoda stycznych przedzial nr 1 ([6;8])
x=7.227625 y=-0.247805
x=7.366706 y=-0.019047
x=7.379371 y=-0.000167
metoda stycznych przedzial nr 2 ([8.3;10])
x=8.968366 y=-0.406259
x=8.729374 y=-0.034459
x=8.704639 y=-0.000466
metoda stycznych przedzial nr 3 ([12;15])
x=12.753573 y=0.193995
x=12.676923 y=-0.001141
x=12.677369 y=-0.000000
```

Komentarz

c. Metoda stycznych w zadanych warunkach działa zdecydowanie najlepiej zmniejszając w każdym kroku wartość bezwzględną y ok. dziesięciokrotnie.
   W zadanych warunkach **średnia ilość iteracji** wynosiła **3**. Jednakże zagadnieniem było dobranie przedziału startowego tak aby krok 4 algorytmu został spełniony.

Warunki, w których metoda nie działa

c. Metoda stycznych jest najłatwiejsza do „zepsucia" – trzeba dobrać punkt startowy tak aby pochodna w kolejnych punktach była na tyle stroma aby nie wyszła poza przedział startowy.
   Przykładowo dla wywołania:

```
x_styczne = met_stycznych([0 8], dokladnosc_zer, ilosc_iteracji)
```

```
metoda stycznych przedzial nr 1
x=1.913351 y=-3.376055
    1.9134    8.0000

   15.4401
```

Error using **mnum_0301>nowy_przedzial_styczny** (line 122)
Error. Punkt poza przedzialem. Nalezy wybrac inny przedzial poczatkowy

Error in **mnum_0301>met_stycznych** (line 109)
[c przedzialy(i,:)] = nowy_przedzial_styczny(przedzialy(i,:), pierwotny_przedzial);

Wyznaczony punkt leży poza przedziałem startowym co jest sygnalizowane przez komunikat o błędzie.

Wniosek

Najszybszą metodą okazała się metoda stycznych – rekomendowałbym ją jako funkcję pierwszego wyboru dla funkcji, których wykres wydaje się mieć nachylenia dalekie od płaskich. Jednakże przy tej metodzie trzeba uważnie dobierać przedział startowy.

Najlepszą metodą w mojej opinii pod względem stosunku niezawodności do szybkości jest metoda siecznych. Tę rekomendowałbym jako uniwersalną.

Z kolei najprostszą okazała się metoda bisekcji, która nie wymagała podczas implementacji żadnych obliczeń analitycznych.

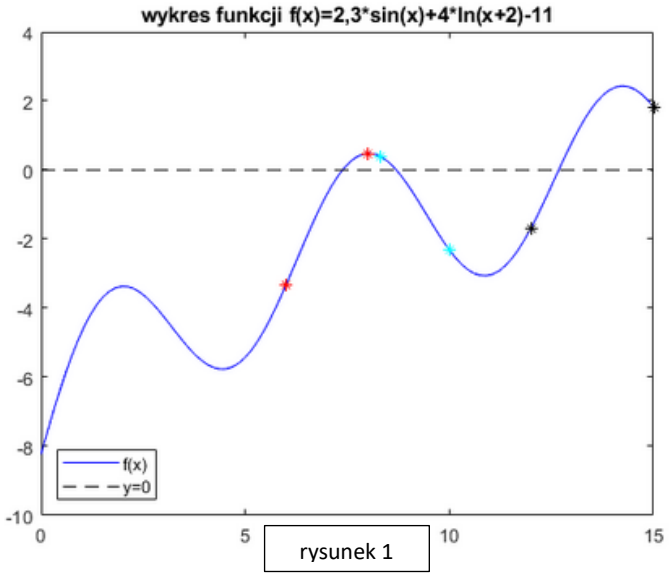
Zadanie 2. **Metoda Mullera MM1**

Celem zadania jest napisanie programu obliczającego za pomocą metody Mullera MM1 wszystkie zera funkcji:

$$f(x) = -x^4+2.5x^3+2.5x^2+x+0.5$$

Koncepcja rozwiązania

1. Początkowo napisana została metoda wspomagająca rysowanie wykresu funkcji (rysunek 1).
   Przedziały zaś zostały dobrane wg. dwóch kryteriów:
   3) maksymalna wielkość
   4) możliwość poprawnego wykonania każdej metody dla zadanej funkcji w zadanym przedziale.



rysunek 1

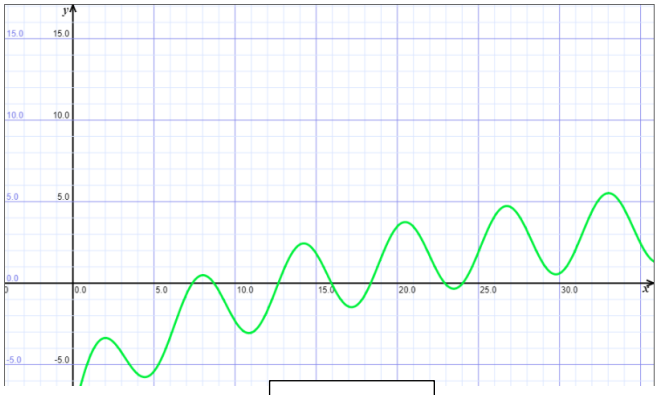
   Ostateczne przedziały startowe zostały wybrane *metodą inżynierskiej intuicji* oraz *prób i błędów* jak przedstawiono na rysunku 1 - [6;8], [8.3;10], [12; 15].

   Ograniczenia:

   - **Każda z metod** została ograniczona parametrem globalnym **`dokladnosc_zer`** określającym maksymalny moduł liczby mogącej być uznawanej w przybliżeniu za równą 0.
   - **Metoda bisekcji oraz siecznych** zostały również ograniczone parametrem globalnym **`wielkosc_przedzialu`**, aby zapobiegać przypadkom, w którym dla funkcji o małym nachyleniu pierwiastki zostaną wyznaczone niedokładnie.
   - **Metoda stycznych (Newtona)** została ograniczona ze względu na maksymalną liczbę iteracji - **`ilosc_iteracji`**.
   - **Metoda stycznych (Newtona)** zostaje przerwana gdy któryś z kolejno wyznaczonych punktów wychodzi poza początkowy przedział – funkcja **`nowy_przedzial_sieczny`** chroni przed pochodną o zbyt małym nachyleniu zgłaszając błąd w niedozwolonym przypadku.

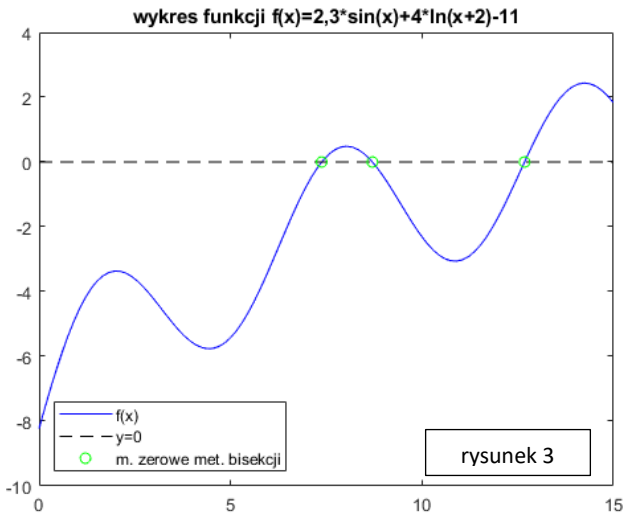
Sprawdzenie

3. Do sprawdzenia poprawności wyznaczonego wykresu użyty został generator wykresów ze strony: matemaks.pl/program-do-rysowania-wykresow-funkcji.html
   Wygenerowany został rysunek 2 pokrywający się z rysunkiem 1.



rysunek 2

4. Z kolei sprawdzenie poprawności wyznaczonych miejsc zerowych następowało na dwóch drogach – analityczne wyliczenie za pomocą funkcji `wartość_funkcji(x)` oraz graficznie- wszystkie 3 metody wygenerowały wykres z rysunku 3 co potwierdza poprawność ich działania.



rysunek 3

Komentarz

b. W celu zautomatyzowania wyboru przedziałów startowych dla metody bisekcji oraz siecznych można napisać funkcję **`wybierz_przedziały`** działającą wg. listy kroków:
   1) Podziel badany przedział na 100 części – małych przedziałów. Przedziały te powinny nachodzić na siebie z dokładnością do epsilona, tak aby nie było przypadku, w którym jeden kończy się na miejscu zerowym, a drugi na nim zaczyna (przypadek nieobsłużony przez metody).
   2) Dla każdego małego przedziału przeprowadź test **`sprawdzenie_przedziału`** sprawdzający warunek f(x1)*f(x2)<0.
   3) W przypadku spełnienia warunku dopisz go do wektora przedziałów, w których znajduje się pierwiastek funkcji.
   4) Na podstawie utworzonego wektora oblicz miejsca zerowe.

W projekcie metoda ta nie została zaimplementowana ze względu na wymóg doboru szerokich przedziałów startowych.

Zadanie 1a. **Metoda bisekcji**

Koncepcja rozwiązania

Zaimplementowana została klasyczna metoda bisekcji:

5. Z przedziału [x1,x2] wyznacz punkt c = (x1+x2)/2.
6. Jeśli c jest miejscem zerowym oraz przedział jest odpowiednio mały zwróć c.
7. Jeśli nie za pomocą metody `sprawdź_przedział` wybierz [x1,c] lub [c,x2], w którym znajduje się miejsce zerowe.
8. Powróć do kroku i.

Sprawdzenie

Dla parametrów `dokladnosc_zer=0.001` oraz `wielkosc_przedzialu=0.1`.

```
metoda bisekcji przedzial nr 1 ([6;8])
x=7.000000 y=-0.700033
```

Komentarz

d. Metoda bisekcji działa zgodnie z oczekiwaniami – w każdym kroku widać, że o połowę zmniejsza zadany w danej iteracji przedział. Co ciekawe jest to metoda, która w każdym kolejnym kroku nie zawsze zmniejsza wartość bezwzględną y na co przykładem jest przedział nr 1.
   W zadanych warunkach **średnia ilość iteracji** wynosiła **10,66**.

Warunki, w których metoda nie działa

d. Metoda bisekcji jest najprostszą metodą, która może najłatwiej zawieźć przez błąd programisty – zły dobór przedziału startowego (niespełniający warunku f(x1)*f(x2)<0. Gdy uruchomiłem program właśnie dla takiego przedziału zapętlił on po ok. 20 iteracjach jeden wynik.
   Przypuszczałem, że metoda bisekcji może zawieźć gdy przedziałem startowym będzie `[x1 pierwiastek_funkcji]` zaś `dokładność_zer` będzie mała. Uruchomiłem więc następującą funkcję:

```
x_bisekcja = bisekcja([0 7.3794839], 0.000001, wielkosc_przedzialu)
```

   output

```
ilosc iteracji 24 x_zero 7.379483e+00f, wartosc -3.645547e-07f
```

   Jak widać metoda bisekcji nawet w takim przypadku szybko zbiega do poprawnego wyniku.

```
norma_res = 9×2
met. rown norm    met QR
   34.3326      34.3326 – stopień 0
   24.5832      24.5832 – stopień 1
    7.3647      7.3647 – stopień 2
    1.4390      1.4390 – stopień 3
    1.3958      1.3958 – stopień 4
    0.8501      0.8501 – stopień 5
    0.7595      0.7595 – stopień 6
    0.7069      0.7069 – stopień 7
    0.6997      0.7181 – stopień 8
```

Komentarz

Obie metody dobrze dokonują aproksymacji dla zadanego zestawu danych już przy 3 stopniu wielomianu. Do 7. stopnia wielomianu różnice pomiędzy obiema metodami są wręcz niezauważalne – zarówno wykresy jak i norma residuum są takie same. Przy większym stopniu metoda QR traci nieznacznie na rzecz metody równań normalnych. Wynikać to może ze stosowania dodatkowego rozkładu, który przy większych macierzach nieco traci na

dokładności – może to dziać się np. na etapie ortogonalizacji Grama-Schmidta gdyż występuje coraz większe pole do popełnienia błędów np. w elemencie sumy.

Załącznik 1. **Kod źródłowy zadania 1.**

start

```
clc;
clear;
x = linspace(0,15,100);
```

przedzialy poczatkowe

```
przedzialy = [6 8; 8.3 10; 12 15];
dokladnosc_zer = 0.001;
wielkosc_przedzialu = 0.01;
ilosc_iteracji = 100;
for i=1:3
    %wybor i-tego przedzialu
    if sprawdzenie_przedzialu(przedzialy(i,:)) == 0
        error('Error, zostaly wybrane zle przedzialy');
    end
end
```

wykres

```
figure
y = wartosc_funkcji(x);
plot(x,y,'b-',[0 15], [0 0], 'k--', przedzialy(1,:),
wartosc_funkcji(przedzialy(1,:)), 'r*', przedzialy(2,:),
wartosc_funkcji(przedzialy(2,:)), 'c*', przedzialy(3,:),
wartosc_funkcji(przedzialy(3,:)), 'k*');
legend({'f(x)', 'y=0'},'Location','southwest');
title('wykres funkcji f(x)=2,3*sin(x)+4*ln(x+2)-11');
```

oszacowanie miejsc zerowych na podstawie rysunku (skrypt prof Tatjeskiego mowi, zeby estymowac na poodstawie rysunku)

```
% wartosc_funkcji(7.25)
% wartosc_funkcji(8)
% wartosc_funkcji(9)
% wartosc_funkcji(12.5)
```

porownanie

```
% przekroczenie przedzialu startowego
% x_styczne = met_stycznych([0 8], dokladnosc_zer, ilosc_iteracji)
x_bisekcja = bisekcja(przedzialy, dokladnosc_zer, wielkosc_przedzialu)
x_sieczne = met_siecznych(przedzialy, dokladnosc_zer, wielkosc_przedzialu)
x_styczne = met_stycznych(przedzialy, dokladnosc_zer, ilosc_iteracji)
y_bisekcja = wartosc_funkcji(x_bisekcja)
y_sieczne = wartosc_funkcji(x_sieczne)
y_styczne = wartosc_funkcji(x_styczne)
figure
plot(x,y,'b-',[0 15], [0 0], 'k--', x_bisekcja, wartosc_funkcji(x_bisekcja),
'go');
legend({'f(x)', 'y=0', 'm. zerowe met. bisekcji'},'Location','southwest');
title('wykres funkcji f(x)=2,3*sin(x)+4*ln(x+2)-11');
plot(x,y,'b-',[0 15], [0 0], 'k--', x_sieczne, wartosc_funkcji(x_sieczne),
'go');
legend({'f(x)', 'y=0', 'm. zerowe met. siecznych'},'Location','southwest');
title('wykres funkcji f(x)=2,3*sin(x)+4*ln(x+2)-11');
plot(x,y,'b-',[0 15], [0 0], 'k--', x_styczne, wartosc_funkcji(x_styczne),
'go');
legend({'f(x)', 'y=0', 'm. zerowe met. stycznych'},'Location','southwest');
title('wykres funkcji f(x)=2,3*sin(x)+4*ln(x+2)-11');
```

funkcje pomocnicze glowne

1. metoda bisekcji

```
function x = bisekcja(przedzialy, dokladnosc_zer, wielkosc_przedzialu)
    ilosc_pierwiastkow = size(przedzialy,1);
    x = zeros(ilosc_pierwiastkow);
    x = wektor(x);
    for i = 1:ilosc_pierwiastkow
        fprintf('metoda bisekcji przedzial nr %d\n', i);
        iteracje = 0;
        c = 0;
```

```
        while (abs(wartosc_funkcji(c))>dokladnosc_zer | (przedzialy(i,2)-
przedzialy(i,1))>wielkosc_przedzialu)
            iteracje = iteracje + 1;
            [c przedzialy(i,:)] = polowienie_przedzialu(przedzialy(i,:));
            fprintf('x=%f y=%f\n', c, wartosc_funkcji(c));
        end
        x(i) = c;
    end
end
```

polowienie przedzialow dla metody bisekcji

```
function [c nowy_przedzial] = polowienie_przedzialu(przedzial)
    c = (przedzial(1)+przedzial(2))/2;
    if(sprawdzenie_przedzialu([przedzial(1) c]) == 1)
        nowy_przedzial = [przedzial(1) c];
    else
        nowy_przedzial = [c przedzial(2)];
    end
end
```

2. metoda siecznych

```
function x = met_siecznych(przedzialy, dokladnosc_zer, wielkosc_przedzialu)
    ilosc_pierwiastkow = size(przedzialy,1);
    x = zeros(ilosc_pierwiastkow);
    x = wektor(x);
    for i = 1:ilosc_pierwiastkow
        fprintf('metoda siecznych przedzial nr %d\n', i);
        c = 0;
        while (abs(wartosc_funkcji(c))>dokladnosc_zer | (przedzialy(i,2)-
przedzialy(i,1))>wielkosc_przedzialu)
            [c przedzialy(i,:)] = nowy_sieczny_przedzial(przedzialy(i,:), c);
            fprintf('x=%f y=%f\n', c, wartosc_funkcji(c));
        end
        x(i) = c;
    end
end
```

zmniejszenie przedzialow dla metody siecznych

```
function [d nowy_przedzial] = nowy_sieczny_przedzial(przedzial, c)
    d = wyznacz_zero_f_liniowej(przedzial);
    if(przedzial(2) == c)
        nowy_przedzial = [d przedzial(2)];
    else
        nowy_przedzial = [przedzial(1) d];
    end
end
```

wyznaczenie zera f. liniowej wyznaczonej na podstawie punktow z koncow przedzialu na podstawie wyliczonego analitycznie wzoru

```
function c = wyznacz_zero_f_liniowej(przedzial)
    x1 = przedzial(1);
    x2 = przedzial(2);
```

```
    y1 = wartosc_funkcji(x1);
    y2 = wartosc_funkcji(x2);
    c = (x2*y1-x1*y2)/(y1-y2);
end
```

3. metoda stycznych

```
function x = met_stycznych(przedzialy, dokladnosc_zer, ilosc_iteracji)
    ilosc_pierwiastkow = size(przedzialy,1);
    x = zeros(ilosc_pierwiastkow);
    x = wektor(x);
    for i = 1:ilosc_pierwiastkow
        fprintf('metoda stycznych przedzial nr %d\n', i);
        iteracje = 0;
        c = 0;
        pierwotny_przedzial = przedzialy(i,:);
        while (abs(wartosc_funkcji(c))>dokladnosc_zer & iteracje <
ilosc_iteracji)
            iteracje = iteracje + 1;
            [c przedzialy(i,:)] = nowy_przedzial_styczny(przedzialy(i,:),
pierwotny_przedzial);
            fprintf('x=%f y=%f\n', c, wartosc_funkcji(c));
        end
        x(i) = c;
    end
end
```

zmniejszenie przedzialow dla metody stycznych

```
function [d nowy_przedzial] = nowy_przedzial_styczny(przedzial,
pierwotny_przedzial)
    d = nowy_punkt_styczny(przedzial(1));
    if(d >= pierwotny_przedzial(1) & d <= pierwotny_przedzial(2))
        nowy_przedzial = [d przedzial(2)];
    else
        disp(przedzial);
        disp(d);
        error('Error. Punkt poza przedzialem. Nalezy wybrac inny przedzial
poczatkowy');
    end
end
```

wyznaczenie miejsca zerowego na podstawie pochodnej oraz poprzedniego punktu

```
function x0 = nowy_punkt_styczny(x)
    m = wartosc_pochodnej(x);
    y = wartosc_funkcji(x);
    x0 = (m*x-y)/m;
end
```

wyznaczenie pochodnej dla podanych x-ow

```
function y = wartosc_pochodnej(x)
    y = 23*cos(x)/10+4/(x+2);
end
```

funkcje pomocnicze dodatkowe

wyznaczenie wyjsc dla podanych x-ow i zadanej funkcji

```
function y = wartosc_funkcji(x)
    [temp rozmiar_x] = size(x);
    if rozmiar_x == 1
        x = x';
        rozmiar_x = temp;
    end
    y = zeros(rozmiar_x);
    y = y(:,1);
    for i = 1:rozmiar_x
        y(i,1) = 2.3*sin(x(1,i))+4*log(x(1,i)+2)-11;
    end
end
```

sprawdzenie czy w podanym przedziale jest miejsce zerowe

```
function result = sprawdzenie_przedzialu(przedzial)
    if wartosc_funkcji(przedzial(1))*wartosc_funkcji(przedzial(2)) < 0
        result = 1;
    else
        result = 0;
    end
end
```

wartosc najwiekszego bledu

```
function y = najwieksze_zero(x)
    y = max(abs(wartosc_funkcji(x)));
end
```

funkcja wektoryzujaca macierz diagonalna

```
function w = wektor(A)
    rozmiar = size(A);
    for i = 1:rozmiar
        w(i,1) = A(i,i);
    end
end
```

Załącznik 2. **Kod źródłowy zadania 2.**