Katedra Automatyki i Robotyki **Metody optymalizacji** Ćwiczenie nr 3 – Metoda złotego podziału		
L.p.	**Imię i nazwisko**	**Data i godzina**
1.	Jakub Szczypek	04.11.22r. godz. 15:30 – 17:00 (piątek)

1. Charakterystyka metody złotego podziału

Metoda złotego podziału jest to przybliżona metoda poszukiwania lokalizacji minimum funkcji jednej zmiennej na zadanym przedziale. Jest to metoda należąca do grupy metod bezgradientowych. Warunkiem koniecznym stosowania tej metody jest unimodalność badanej funkcji na zadanym przedziale – funkcja posiada dokładnie jedno minimum w przedziale [a, b] i jest ciągła. Lokalizacja minimum pochodzi z arytmetycznie uśrednionych granic przedziału wyjściowego:

$$[a^{(n)},\ b^{(n)}]$$

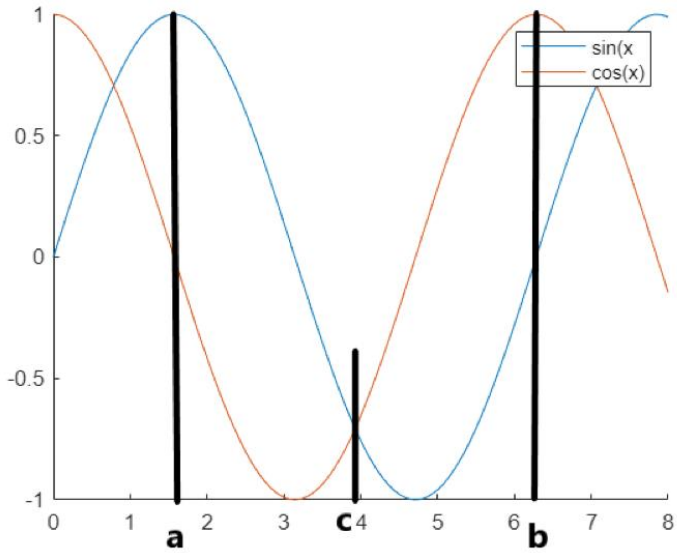
dla którego spełniony jest warunek:

$$b^{(n)} - a^{(n)} < \varepsilon$$

Gdzie:

ε – przyjęta dokładność poszukiwań, przeważnie wynosząca poniżej 0,1.

Aby dokonać redukcji przedziału nieokreśloności, musimy znać wartość funkcji w co najmniej dwóch jego punktach wewnętrznych. Znając wartość funkcji w tylko jednym punkcie wewnętrznym, nie można określić, po której stronie znajduje się poszukiwane minimum. Poniżej przedstawiam przykład dlaczego wymagamy dwóch punktów wewnętrznych.

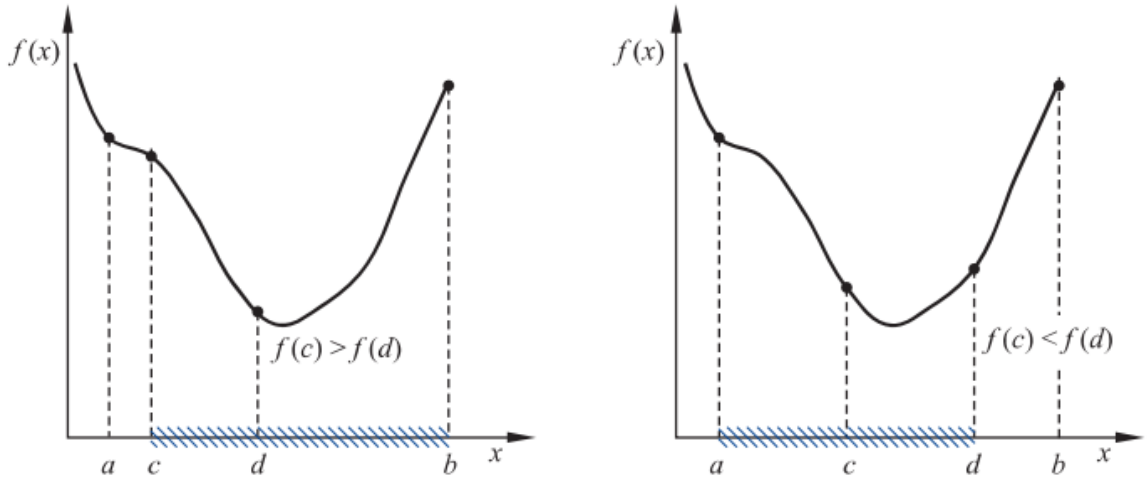


Rysunek 1. Funkcja sin(x) i cos(x)

Po pierwsze wybieramy sobie punkt C. Teraz możemy zauważyć, że dla funkcji cosinus minimum funkcji znajduje się w przedziale [a, c] i jest to cos(π), zaś dla funkcji sinus w przedziale [c, b] i jest to sin($\frac{3}{2}\pi$). Istnieje w takim razie możliwość, że utracimy unimodalność funkcji na przedziale w kolejnej iteracji metody.

2. Interpretacja dwóch punktów wewnętrznych

Jeżeli nasza funkcja f(x) jest unimodalna w przedziale [a, b], to do określenia podprzedziału, w którym leży minimum, wystarczy obliczyć wartości w dwóch punktach wewnętrznych c, d (c < d) tego przedziału. Jeżeli f(c) > f(d), to minimum funkcji znajduje się na prawo od punktu d i przedział poszukiwań zawężamy do [c, b]. Natomiast jeżeli f(c) < f(d) to minimum znajduje się po prawej stronie punktu c i zawężamy przedział poszukiwań do [a, d].



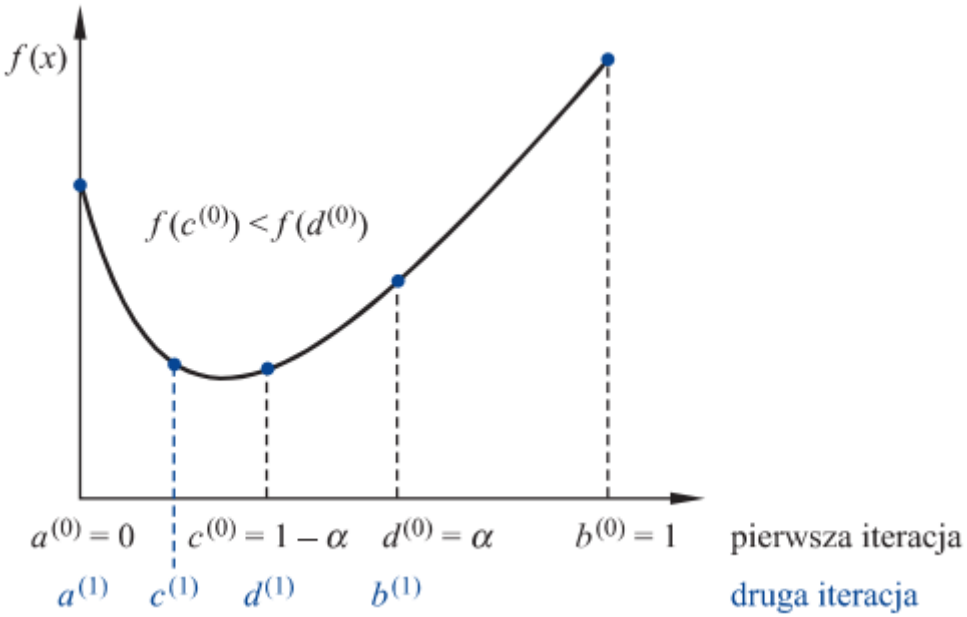
Rysunek 2. Lokalizowanie minimum w przedziale [a, b]

Znając przedział, w którym znajduje się minimum funkcji f, możemy go stopniowo zawężać, aż osiągnie długość równą zadanej dokładności obliczeń ε. Uzyskamy to, obliczając wartości funkcji w wybranych punktach przedziału, a efektywność tego podejścia zależy od kolejnego ich wyboru. Należy zauważyć, że ze względu na fakt, że jeden z punktów c, d używanych do redukowania przedziału poszukiwań leży zawsze wewnątrz nowego przedziału, może być on z powodzeniem wykorzystywany jako punkt próbny w następnym etapie poszukiwań.

3. Metoda złotego podziału

Jest to jedna z najczęściej stosowanych metod służących do zawężania przedziału poszukiwań. Idea metody oparta jest na założeniu, że w każdym kroku obliczeń długość przedziału poszukiwań zmniejszana jest w stałym stosunku α. Aby to osiągnąć należy podzielić początkowy przedział poszukiwań na trzy podprzedziały spełniające zależność:

$$\frac{b^{(i)} - c^{(i)}}{b^{(i)} - a^{(i)}} = \frac{d^{(i)} - a^{(i)}}{b^{(i)} - a^{(i)}} = \alpha$$



Rysunek 3. Zilustrowanie pierwszych dwóch iteracji metody

Przekształcając poprzednią zależność otrzymujemy postaci i-tych wartości c oraz d:

$$c^{(i)} = b^{(i)} - \alpha(b^{(i)} - a^{(i)})$$

$$d^{(i)} = a^{(i)} + \alpha(b^{(i)} - a^{(i)})$$

Zauważmy, że $f(c^{(i)}) < f(d^{(i)})$ zostało spełnione dla i = 0, zatem nowy przedział poszukiwań to $[a^{(i)}, d^{(i)}]$, a prawym punktem pośrednim $c^{(i)}$. Można zatem zapisać:

$$a^{(i+1)} = a^{(i)}$$

$$b^{(i+1)} = b^{(i)}$$

$$d^{(i+1)} = c^{(i)}$$

$$c^{(i+1)} = b^{(i+1)} - \alpha(b^{(i+1)} - a^{(i+1)})$$

Finalnie, korzystając z otrzymanych zależności, wyznaczamy wartość współczynnika α:

$$\alpha = \frac{d^{(i+1)} - a^{(i+1)}}{b^{(i+1)} - a^{(i+1)}} = \frac{c^{(i)} - a^{(i)}}{d^{(i)} - a^{(i)}} = \frac{b^{(i)} - \alpha\left(b^{(i)} - a^{(i)}\right) - a^{(i)}}{a^{(i)} + \alpha(b^{(i)} - a^{(i)}) - a^{(i)}} = \frac{1-\alpha}{\alpha}$$

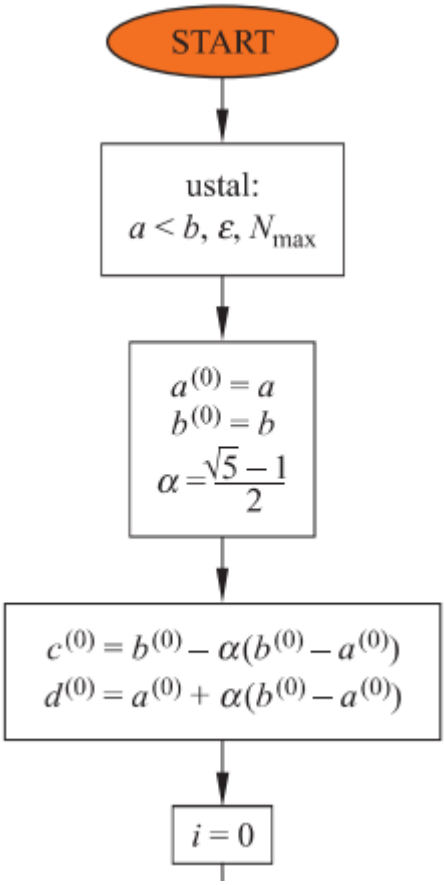
Równoważnie, możemy zapisać wynik w postaci:

$$\alpha^2 + \alpha - 1 = 0$$

Którego jedynym dodatnim rozwiązaniem jest $\alpha = \frac{\sqrt{5}-1}{2}$, jest to tak zwana złota liczba.

4. Implementacja metody w Matlabie

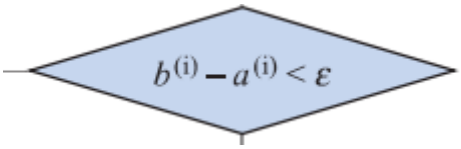
Określamy przedział poszukiwań, dokładność oraz maksymalną liczbę iteracji, by przekazać je do funkcji realizującej metodę złotego przedziału. W utworzonej funkcji określamy punkty początkowe, nadajemy wartość alfa oraz określamy punkty wewnętrzne i zaczynamy iterowanie




```
function [x, i] = gold_divide(start, stop, func, eps, Nmax)
    i = 0;
    alpha = (sqrt(5) - 1) / 2;
    a_i = start;
    b_i = stop;
    c_i = stop - alpha * (stop - start);
    d_i = start + alpha * (stop - start);
```

Rysunek 4. Schemat blokowy algorytmu metody złotego podziału i odpowiadający mu kod w Matlabie

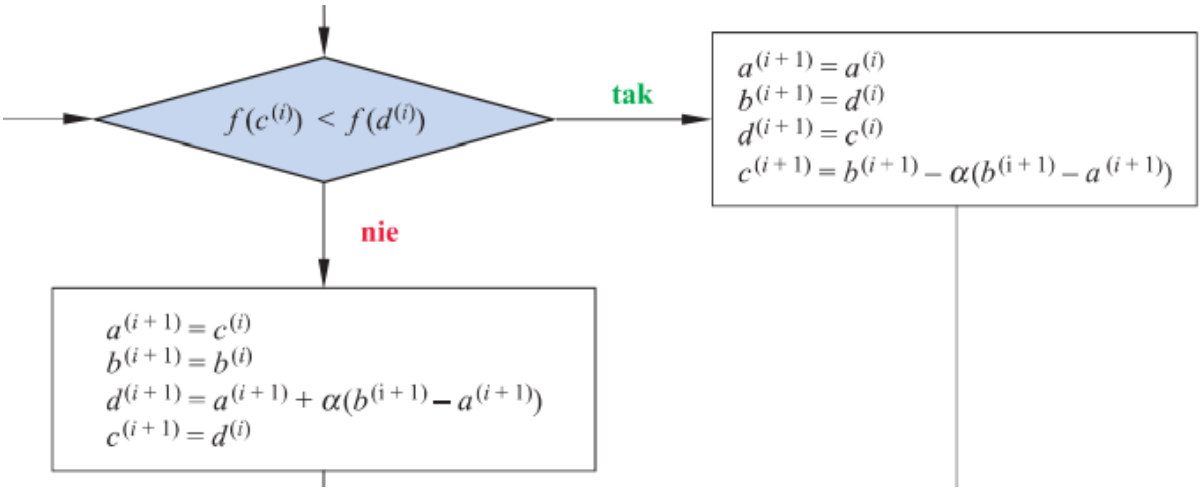
Następnie tworzymy pętlę while, która będzie wykonywana dopóki szerokość końcowego przedziału jest większa od ustalonej dokładności oraz dopóki wykonaliśmy mniej iteracji niż mamy ustalone dla maksymalnej liczby iteracji




```
while abs(b_i - a_i) > eps && i < Nmax
```

Rysunek 5. Schemat blokowy algorytmu metody złotego podziału i odpowiadający mu kod w Matlabie

W zależności od wartości funkcji w punktach wewnętrznych przesuwane są granice przedziałów oraz obliczany jest nowy punkt wewnętrzny, który stał się teraz jedną z granic przedziału.




```
if func(c_i) < func(d_i)
    a_i = a_i;
    b_i = d_i;
    d_i = c_i;
    c_i = b_i - alpha * (b_i - a_i);
```

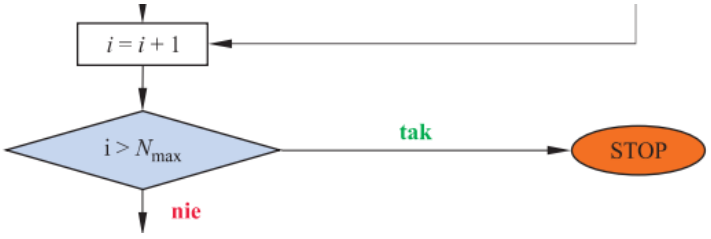
Ta funkcja wykonywana jest kiedy warunek $f(c^{(i)}) < f(d^{(i)})$ jest spełniony.

```
else
    a_i = c_i;
    b_i = b_i;
    c_i = d_i;
    d_i = a_i + alpha*(b_i - a_i);
end
```

A ta funkcja jest wykonywana kiedy $f(c^{(i)}) > f(d^{(i)})$.

Rysunek 6. Schemat blokowy algorytmu metody złotego podziału i odpowiadający mu kod w Matlabie

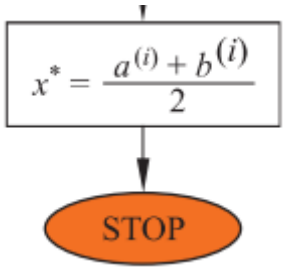
Następnie zwiększamy licznik iteracji oraz sprawdzamy, czy nie napotkaliśmy warunku stopu – przekroczenie maksymalnej liczby iteracji.




```
i = i + 1;
if i > Nmax
    fprinf("Nie udało znaleźć się rozwiązania w %d krokach", Nmax)
end
```

Rysunek 6. Schemat blokowy algorytmu metody złotego podziału i odpowiadający mu kod w Matlabie

Po otrzymaniu ostatecznego przedziału $[a^{(i)}, b^{(i)}]$ wyliczamy przewidywane położenie minimum funkcji.




```
x = (a_i + b_i)/2;
```

Rysunek 7. Schemat blokowy algorytmu metody złotego podziału i odpowiadający mu kod w Matlabie

5. Wykorzystanie metody w ramach laboratorium

W trakcie pracy na laboratorium należało dla dwóch podanych funkcji f(x) i g(x) wyznaczyć lokalizacje minimów z wykorzystaniem zaimplementowanej wcześniej metody złotego podziału. Po wyznaczeniu przedziałów początkowych przy użyciu metody ekspansji, którą napisaliśmy na poprzednich zajęciach, wystarczyło użyć je (nasze przedziały początkowe) w odpowiednio zaimplementowanej funkcji wykonującej algorytm złotego podziału. Poniżej przedstawiam napisaną przeze mnie funkcję pomocniczą do wyświetlania wyników oraz kod który wyświetla poszczególne funkcje.

Funkcja display do wyświetlania wyników:

```
function display(f, a, b, epsilon, Nmax, result, i)
    disp('Dla funkcji ' + string(f) +': ')
    disp(['Dla punktu startowego: ', num2str(a), ' i punktu końcowego: ', num2str(b)])
    disp(['dokładności rozwiązania: ', num2str(epsilon)]);
    disp(['maksymalnej ilości liczby iteracji: ', num2str(Nmax)])
    disp(['minimum funkcji wyznaczone metodą złotego podziału to :'])
    disp([(result)])
    disp(['Wynik otrzymano w ', num2str(i), ' iteracjach'])
    hold off
end
```

Rysunek 8. Funkcja Display służąca do opisu funkcji

```
figure()
fplot(func,[start stop])
hold on
plot( x, func(x),'r*')
hold off
```

Rysunek 9. Część kodu w funkcji gold_divide służący do plotowania wykresów.

Funkcja f:

```
function f=f(x)
    f=0.01*sin(10*x(1))+(0.07*x(1).^2-0.4).^2;
end
```

Rysunek 10. Funkcja f(x)

Funkcja g:

```
function g=g(x)
    px = [-10 -5 0 5 10];
    py = [0.0001 -0.1 0 0.1 -0.0001];
    stopien = 5;
    W = polyfit(px, py, stopien);
    g = (0.05*sin(1*x)+W(1)*x.^stopien+W(2)*x.^(stopien-1)+W(3)*x.^(stopien-2)+W(4)*x.^(stopien-3)+W(5)*x.^(stopien-4)+W(6)*x.^(stopien-5))*(0.001*sin(x/10))+0.000001*x;
end
```

Rysunek 11. Funkcja g(x)

Do prezentacji:

```
%Dla funkcji f(x) dla punktów startowych: -8, -0.5, 1, 8 z metody ekspansji:
start = [-4.1557, -3.0629, 1.7594, -0.64976];
stop = [0.64976, -1.6391, 2.7086, 4.1557];
%Dla funkcji g(x) dla punktów startowych: -8, -0.5, 1, 8 z metody ekspansji:
start1 = [-5.4371, -0.275, -0.70895, 20.9746];
stop1 = [-2.2335, 0.00625, 0.24062, 37.1929];

%Obliczamy minimum funkcji f(x) za pomocą metody złotego podziału dla wcześniej
%obliczonych punktów początkowych i końcowcych o dokładności eps = 0.05
[x, i] = gold_divide(start(1), stop(1), @f, 0.05, 10000);
display('f', start(1), stop(1), 0.05, Nmax, x, i)

[x, i] = gold_divide(start(2), stop(2), @f, 0.05, 10000);
display('f', start(2), stop(2), 0.05, Nmax, x, i)

[x, i] = gold_divide(start(3), stop(3), @f, 0.05, 10000);
display('f', start(3), stop(3), 0.05, Nmax, x, i)

[x, i] = gold_divide(start(4), stop(4), @f, 0.05, 10000);
display('f', start(4), stop(4), 0.05, Nmax, x, i)
```

Rysunek 12. Wywołanie funkcji f(x)

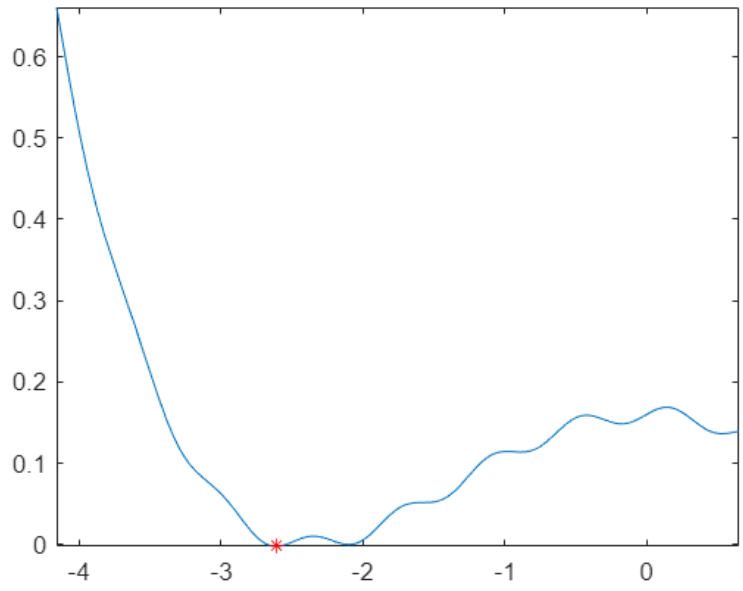
```
%Obliczamy minimum funkcji g(x) za pomocą metody złotego podziału dla wcześniej
%obliczonych punktów początkowych i końcowcych o dokładności eps = 0.001
[x, i] = gold_divide(start(1), stop(1), @g, 0.001, 10000);
display('g', start(1), stop(1), 0.001, Nmax, x, i)

[x, i] = gold_divide(start(2), stop(2), @g, 0.001, 10000);
display('g', start(2), stop(2), 0.001, Nmax, x, i)

[x, i] = gold_divide(start(3), stop(3), @g, 0.001, 10000);
display('g', start(3), stop(3), 0.001, Nmax, x, i)

[x, i] = gold_divide(start(4), stop(4), @g, 0.001, 10000);
display('g', start(4), stop(4), 0.001, Nmax, x, i)
```

Rysunek 13. Wywołanie funkcji g(x)

6. Wyniki

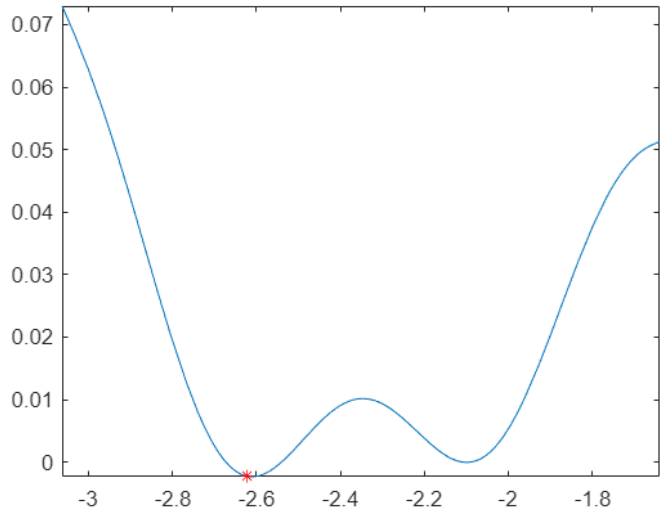
Dla wyżej przedstawionych funkcji oraz kodu wywołującego nasze wyniki dla funkcji f(x) otrzymałem następujące wykresy i informacje:

```
Dla funkcji f:
Dla punktu startowego: -4.1557 i punktu końcowego: 0.64976
dokładności rozwiązania: 0.05
maksymalnej ilości liczby iteracji: 10000
minimum funkcji wyznaczone metodą złotego podziału to :
   -2.6075

Wynik otrzymano w 10 iteracjach
```



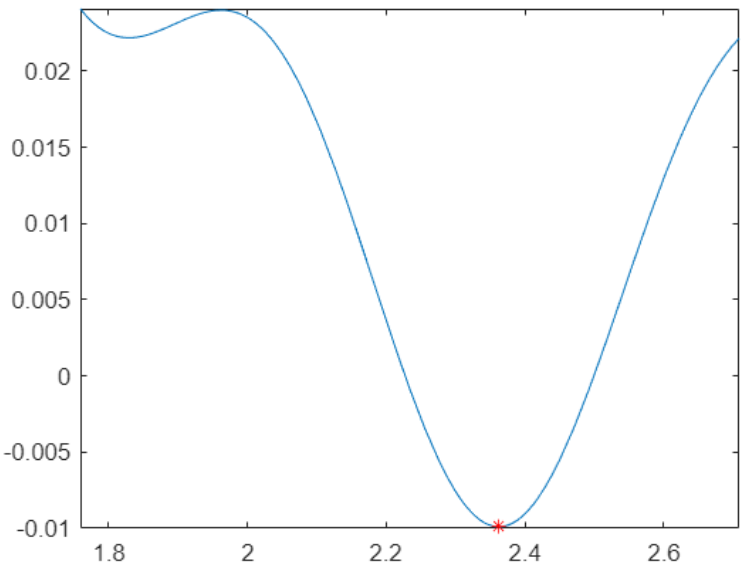
Rysunek 14. Wynik wywołania wcześniej zaimplementowanych funkcji dla f(x)

```
Dla funkcji f:
Dla punktu startowego: -3.0629 i punktu końcowego: -1.6391
dokładności rozwiązania: 0.05
maksymalnej ilości liczby iteracji: 10000
minimum funkcji wyznaczone metodą złotego podziału to :
   -2.6229

Wynik otrzymano w 7 iteracjach
```



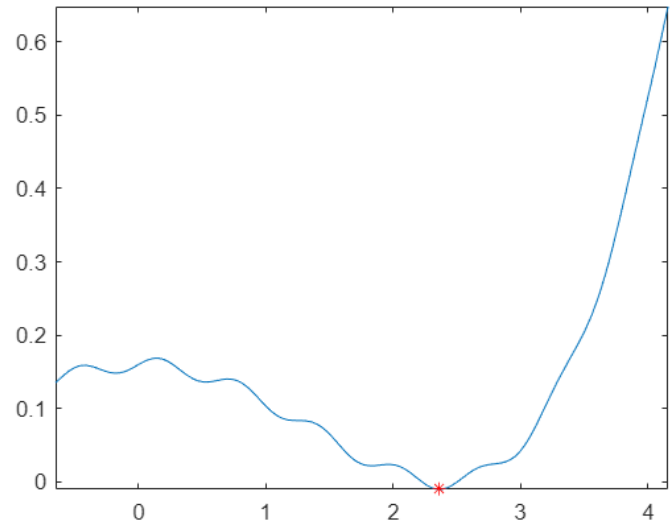
Rysunek 15. Wynik wywołania wcześniej zaimplementowanych funkcji dla f(x)

```
Dla funkcji f:
Dla punktu startowego: 1.7594 i punktu końcowego: 2.7086
dokładności rozwiązania: 0.05
maksymalnej ilości liczby iteracji: 10000
minimum funkcji wyznaczone metodą złotego podziału to :
    2.3624

Wynik otrzymano w 7 iteracjach
```



Rysunek 16. Wynik wywołania wcześniej zaimplementowanych funkcji dla f(x)

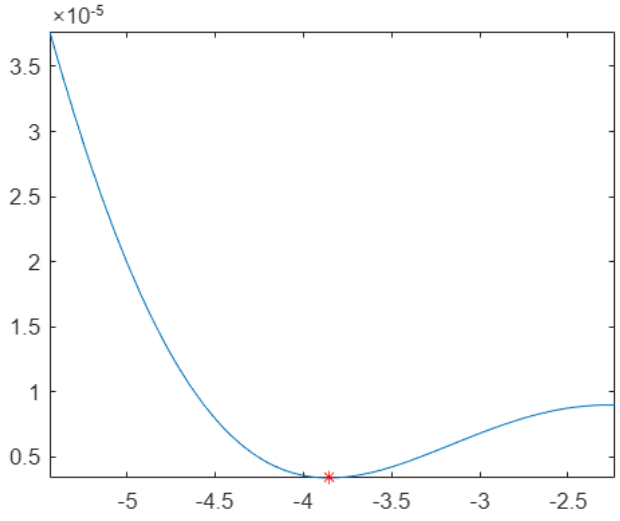
```
Dla funkcji f:
Dla punktu startowego: -0.64976 i punktu końcowego: 4.1557
dokładności rozwiązania: 0.05
maksymalnej ilości liczby iteracji: 10000
minimum funkcji wyznaczone metodą złotego podziału to :
    2.3639

Wynik otrzymano w 10 iteracjach
```



Rysunek 17. Wynik wywołania wcześniej zaimplementowanych funkcji dla f(x)

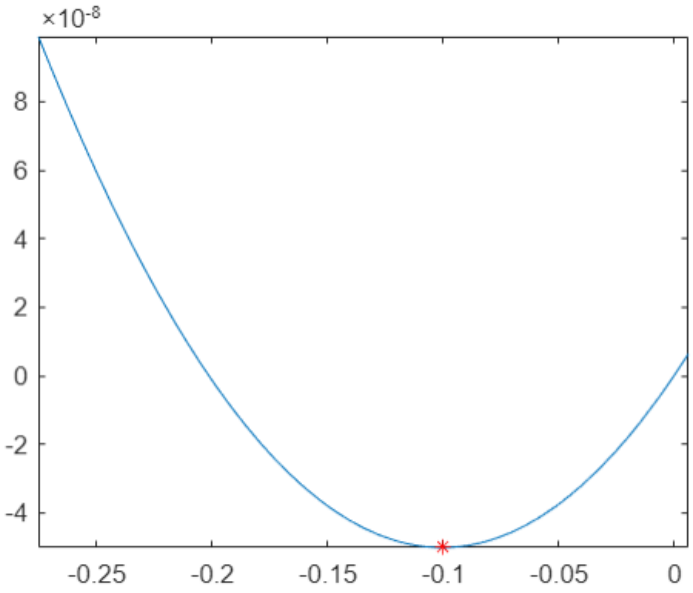
Dla wyżej przedstawionych funkcji oraz kodu wywołującego nasze wyniki dla funkcji g(x) otrzymałem następujące wykresy i informacje:

```
Dla funkcji g:
Dla punktu startowego: -5.4371 i punktu końcowego: -2.2335
dokładności rozwiązania: 0.001
maksymalnej ilości liczby iteracji: 10000
minimum funkcji wyznaczone metodą złotego podziału to :
   -3.8536

Wynik otrzymano w 17 iteracjach
```



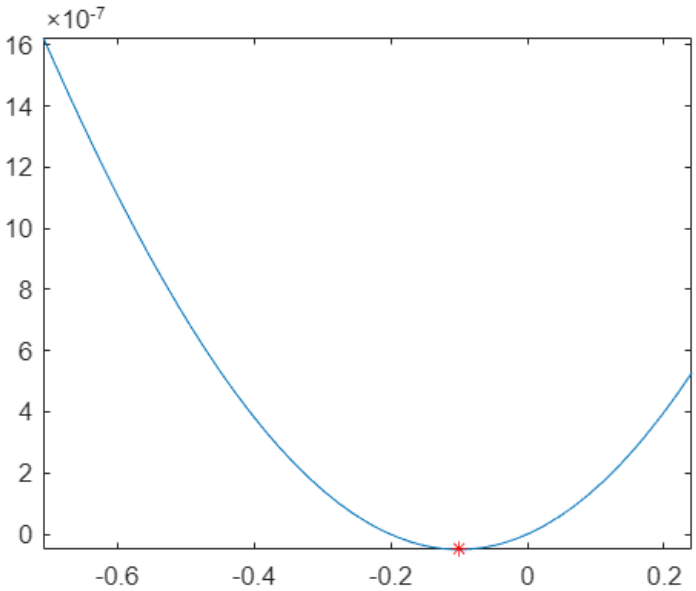
Rysunek 18. Wynik wywołania wcześniej zaimplementowanych funkcji dla g(x)

```
Dla funkcji g:
Dla punktu startowego: -0.275 i punktu końcowego: 0.00625
dokładności rozwiązania: 0.001
maksymalnej ilości liczby iteracji: 10000
minimum funkcji wyznaczone metodą złotego podziału to :
   -0.1002

Wynik otrzymano w 12 iteracjach
```



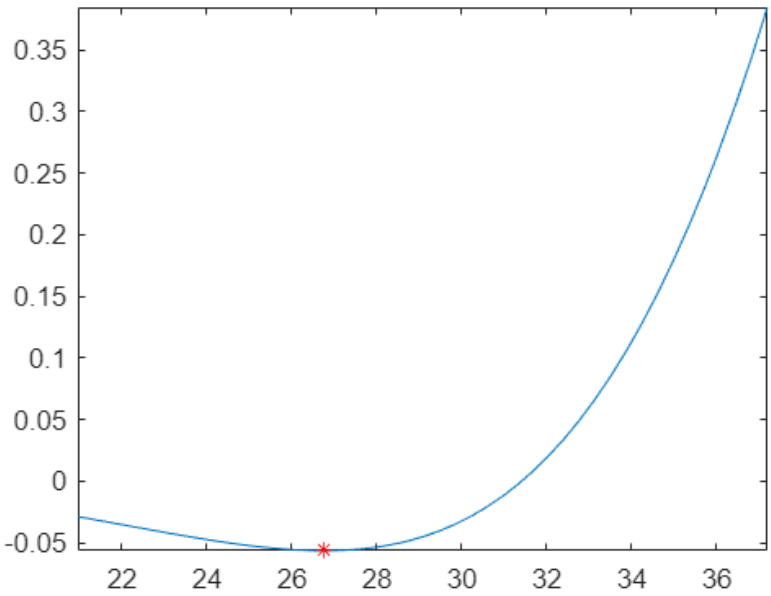
Rysunek 19. Wynik wywołania wcześniej zaimplementowanych funkcji dla g(x)

```
Dla funkcji g:
Dla punktu startowego: -0.70895 i punktu końcowego: 0.24062
dokładności rozwiązania: 0.001
maksymalnej ilości liczby iteracji: 10000
minimum funkcji wyznaczone metodą złotego podziału to :
   -0.1004

Wynik otrzymano w 15 iteracjach
```



Rysunek 19. Wynik wywołania wcześniej zaimplementowanych funkcji dla g(x)

```
Dla funkcji g:
Dla punktu startowego: 20.9746 i punktu końcowego: 37.1929
dokładności rozwiązania: 0.001
maksymalnej ilości liczby iteracji: 10000
minimum funkcji wyznaczone metodą złotego podziału to :
   26.7606

Wynik otrzymano w 21 iteracjach
```



Rysunek 20. Wynik wywołania wcześniej zaimplementowanych funkcji dla g(x)

7. Wnioski

Jak możemy zauważyć na powyższych wynikach, pomimo tego że w przedziale znajduje się więcej niż jedno minimum to funkcja złotego podziału potrafi znaleźć to „najlepsze”. Metoda ta jest efektywna w zależności od tego jak dobry został wybrany przedział początkowy. Trzeba pamiętać, że w tej metodzie musimy mieć zawsze dwa punkty wewnętrzne. Metoda ta może się „zepsuć” jeśli w trakcie którejś iteracji wartości funkcji w punktach wewnętrznych będą równe. Trzeba także zauważyć, że metoda złotego podziału nie oblicza dokładnie wartości minimum funkcji lecz podaje jej przybliżenie.