


Paulina Grabowska 	Metody Numeryczne
Rozwiązywanie równań nieliniowych – metoda bisekcji, metoda Newtona-Raphsona	

## 1. Wstęp

Nieliniowość to właściwość układu, którego wartość wyjściowa nie jest wprost proporcjonalna do danych wejściowych. W algebrze z nieliniowością mamy do czynienia, gdy funkcje nie spełniają warunku addytywności i homogeniczności (warunku liniowości).

Równania nieliniowe to kluczowy obszar matematyki, w którym poszukujemy wartości nieznanymi zmiennymi, przy założeniu, że relacje między nimi nie są liniowe. W przeciwieństwie do równań liniowych, gdzie każda zmienna występuje tylko w pierwszej potęgze, równania nieliniowe mogą zawierać potęgi większe niż jeden, funkcje nieliniowe oraz inne złożone operacje. Równania nieliniowe mogą przybierać różnorodne formy, w zależności od struktury funkcji oraz liczby zmiennych.

## 2. Metody rozwiązywania równań nieliniowych

### 2.1. Metoda bisekcji

Metoda bisekcji jest jedną z najprostszych i najbardziej intuicyjnych metod numerycznych służących do rozwiązywania równań nieliniowych. Jej działanie opiera się na założeniu, że jeśli funkcja  $f(x)$  oraz jej pierwsza i druga pochodna są ciągłe w przedziale  $[a, b]$  oraz funkcja ta przyjmuje różne znaki na jego krańcach ( $f(a) \cdot f(b) < 0$ ), to istnieje taki punkt  $x_1$  wewnątrz tego przedziału, dla którego  $f(x_1) = 0$ . Metoda bisekcji polega na iteracyjnym dzieleniu przedziału  $[a, b]$  na połowy i wybieraniu nowego przedziału, w którym znak funkcji  $f(x)$  się zmienia.

Pierwszym krokiem jest wybór przedziału początkowego, w którym znajduje się rozwiązanie równania nieliniowego. Przedział ten musi spełniać warunek istnienia tylko jednego pierwiastka oraz musi zawierać ten pierwiastek. Następnie sprawdzane są wartości funkcji  $f(x)$  na krańcach tego przedziału.

Następnie przedział początkowy  $[a, b]$  jest dzielony na pół, aby uzyskać punkt środkowy  $x_1$ , czyli  $x_1 = \frac{a+b}{2}$ .

Obliczana jest wartość funkcji  $f(x_1)$  w punkcie środkowym  $x_1$ . Na podstawie znaku funkcji  $f(x_1)$  w punkcie środkowym  $x_1$ , przedział  $[a, b]$  jest aktualizowany. Jeśli  $f(x_1)$  ma przeciwny znak do  $f(a)$ , to nowym przedziałem staje się  $[a, c]$ ; w przeciwnym przypadku nowym przedziałem jest  $[c, b]$ .

Proces podziału i aktualizacji przedziału jest powtarzany iteracyjnie aż do uzyskania dostatecznie małej wartości przedziału lub osiągnięcia zadanej dokładności rozwiązania. Poszukiwany pierwiastek równania wynosi  $\frac{a+b}{2}$ .

Metoda bisekcji jest stosunkowo prostym i stabilnym sposobem rozwiązywania równań nieliniowych, jednak może być stosunkowo wolna w porównaniu do bardziej zaawansowanych metod, szczególnie dla dużych przedziałów początkowych lub funkcji o skomplikowanych kształtach.

## 2.2. Metoda Newtona-Raphsona

Metoda Newtona-Raphsona jest jedną z najczęściej stosowanych i najbardziej efektywnych metod numerycznych służących do znajdowania przybliżonych rozwiązań równań nieliniowych. Jej działanie opiera się na lokalnej liniowej aproksymacji funkcji wokół przybliżonego rozwiązania, co pozwala na szybkie zbliżanie się do rzeczywistego pierwiastka równania.

Pierwszym krokiem jest wybór początkowego przybliżenia rozwiązania równania nieliniowego  $x_0$ . Dobrym wyborem punktu startowego jest punkt znajdujący się w pobliżu rzeczywistego pierwiastka oraz punkt, dla którego pochodna funkcji jest różna od zera.

Dla każdej iteracji  $n$ , nowe przybliżenie rozwiązania  $n+1$  jest obliczane na podstawie poprzedniego przybliżenia  $x_n$  za pomocą wzoru rekurencyjnego:

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

Proces iteracyjny kontynuowany jest do momentu osiągnięcia zadanej dokładności lub maksymalnej liczby iteracji ograniczonej warunkiem stopu. Warunkiem stopu może być

wystarczająco mała odległość pomiędzy kolejnymi przybliżeniami lub też zbliżona do zera wartość funkcji w wyznaczonym punkcie.

Metoda Newtona-Raphsona cechuje się szybką zbieżnością w okolicach rzeczywistego pierwiastka oraz prostotą implementacji.

### 3. Implementacja numeryczna

#### 3.1. funkcje *function* oraz *first\_derivative*

Funkcja *function* przyjmuje wartość  $x$  i oblicza wartość funkcji nieliniowej, której definicję można zmieniać w zależności od potrzeb (*Fragment kodu 1*). Funkcja ta zawiera definicję funkcji problemowej, tj. badanej dla której przeprowadzane będą obliczenia równań nieliniowych.

Funkcja *first\_derivative*: przyjmuje wartość  $x$  oraz parametr  $h$ , który określa, jak daleko od siebie będą punkty, między którymi obliczana jest różnica ilorazów (*Fragment kodu 1*). Korzystając z metody różnic skończonych, oblicza ona przybliżoną wartość pierwszej pochodnej funkcji  $f(x)$  w danym punkcie  $x$ . Innymi słowy, oblicza ona, jak szybko zmienia się wartość funkcji  $f(x)$  w punkcie  $x$  i przybliża tę zmianę na podstawie wartości funkcji w dwóch pobliskich punktach.

```
1. double function(double x) {
2.     return x*x*x - x - 1;
3. }
4.
5. double first_derivative(double x, double h) {
6.     return ((function(x + h) - function(x)) / h);
7. }
```

**Fragment kodu 1:** funkcje *function* oraz *first\_derivative*

#### 3.2. funkcja *bisection\_method*

Na początku funkcja oblicza wartości funkcji  $f(a)$  i  $f(b)$  dla podanych krańców przedziału  $[a, b]$  (*Fragment kodu 2*). Następnie sprawdzane jest, czy iloczyn wartości funkcji na krańcach przedziału jest dodatni czy ujemny. Jeśli jest dodatni, oznacza to, że oba krańce przedziału znajdują się po tej samej stronie osi  $x$ , co sugeruje brak pierwiastka wewnątrz tego przedziału. W takim przypadku funkcja wyświetla komunikat o braku pierwiastka w podanym przedziale i zwraca 0.

Następnie funkcja przechodzi do pętli głównej, która będzie się powtarzać, dopóki długość przedziału  $(b-a)$  jest większa niż zdefiniowana tolerancja  $\epsilon$ . W każdej iteracji pętli obliczany jest punkt środkowy  $x$  przedziału  $[a, b]$  za pomocą wzoru  $x = \frac{a+b}{2}$ .

Następnie obliczana jest wartość funkcji  $f(x)$  w punkcie środkowym  $x$ . Jeśli wartość funkcji  $f(x)$  wynosi dokładnie 0, to oznacza, że punkt  $x$  jest rzeczywistym pierwiastkiem równania. W takim przypadku funkcja zwraca ten pierwiastek.

W zależności od znaku iloczynu  $f(a) \times f(x)$ , przedział  $[a, b]$  jest aktualizowany poprzez przesunięcie jednego z jego końców na punkt  $x$ . Jeśli iloczyn jest ujemny, oznacza to, że pierwiastek znajduje się między  $a$  a  $x$ , więc aktualizujemy  $b$  na  $x$ . W przeciwnym przypadku, pierwiastek znajduje się między  $x$  a  $b$ , więc aktualizujemy  $a$  na  $x$ .

Po aktualizacji przedziału pętla wraca do obliczania punktu środkowego przedziału i powtarza proces, aż długość przedziału będzie mniejsza lub równa zadanej tolerancji  $\epsilon$ . Po zakończeniu pętli głównej, funkcja zwraca średnią arytmetyczną końców przedziału  $\frac{a+b}{2}$ , która stanowi przybliżenie rzeczywistego pierwiastka równania.

```

1. double bisection_method(double a, double b, double epsilon) {
2.     double fa = function(a);
3.     double fb = function(b);
4.     if (fa * fb > 0) {
5.         cout << "no root in given interval. " << endl;
6.         return 0;
7.     }
8.
9.     while ((b - a) / 2 > epsilon) {
10.        double x = (a + b) / 2;
11.        double fx = function(x);
12.        if (fx == 0.0)
13.            return x;
14.        else if (fa * fx < 0)
15.            b = x;
16.        else
17.            a = x;
18.    }
19.    return (a + b) / 2;
20. }

```

**Fragment kodu 2:** funkcja *bisection\_method*

### 3.3. funkcja *Newton\_Raphson\_method*

Na początku algorytmu podawane jest przybliżenie pierwiastka równania nieliniowego  $x_n$ . Następnie algorytm oblicza kolejne przybliżenie pierwiastka równania,  $x_{n+1}$ , korzystając z metody Newtona-Raphsona (*Fragment kodu 3*). Jest to wyrażone analogicznie do wzoru (1),

gdzie  $f(x_n)$  to wartość funkcji nieliniowej w punkcie  $x_n$ , a  $f'(x_n)$  to pierwsza pochodna funkcji w tym punkcie.

Po obliczeniu nowego przybliżenia  $x_{n+1}$ , algorytm sprawdza, czy wartość bezwzględna funkcji  $f(x_{n+1})$  jest mniejsza lub równa zadanej tolerancji  $\epsilon$ . Jeśli tak, to oznacza to, że znaleźliśmy dostatecznie bliskie przybliżenie rzeczywistego pierwiastka, dlatego zwracamy to przybliżenie.

Jeśli warunek stopu nie jest spełniony, czyli wartość bezwzględna funkcji w kolejnym przybliżeniu  $x_{n+1}$  jest większa niż tolerancja  $\epsilon$ , to algorytm rekurencyjnie wywołuje się, podając jako nowe przybliżenie  $x_{n+1}$ . W ten sposób algorytm iteracyjnie zbliża się do rzeczywistego pierwiastka, obliczając kolejne przybliżenia  $x_{n+1}$  aż do spełnienia warunku stopu.

```

1. double Newton_Raphson_method(double xn, double epsilon, double h) {
2.     double xn_plus_1 = xn - function(xn) / first_derivative(xn, h);
3.
4.     if (abs(function(xn_plus_1)) <= epsilon)
5.         return xn_plus_1;
6.     else
7.         return Newton_Raphson_method(xn_plus_1, epsilon, h);
8. }

```

**Fragment kodu 3:** funkcja *Newton\_Raphson\_method*

### 3.4. funkcja *main*

Na początku programu deklarowane są zmienne:

- $a$  i  $b$  określające początek i koniec przedziału dla metody bisekcji,
- $x_0$  określająca punkt początkowy dla metody Newtona-Raphsona,
- $h$  określająca krok używany do obliczenia pierwszej pochodnej dla metody Newtona-Raphsona,
- $\epsilon$  określająca pożądaną dokładność dla obu metod.

Następnie program wywołuje funkcje *bisection\_method* i *Newton\_Raphson\_method*, przekazując im odpowiednie parametry, tj. przedział dla metody bisekcji, punkt początkowy i dokładność dla metody Newtona-Raphsona (*Fragment kodu 4*). Wyniki zwracane przez te funkcje są następnie wyświetlane na standardowym. Na końcu program zwraca wartość 0, co oznacza pomyślne zakończenie działania programu.

```

1. int main() {
2.     double a = 0.0;           // beginning of the interval for bisection_method
3.     double b = 3.0;           // ending of the interval for bisection_method
4.
5.     double x0 = 1.0;           // beginning point for N-R
6.     double h = 0.0001;         // for calculating the derivative
7.
8.     double epsilon = 0.0001;   // accuracy
9.
10.    cout << "bisection method: " << bisection_method(a, b, epsilon) << endl;
11.    cout << "Newton-Raphson method: " << Newton_Raphson_method(x0, epsilon, h) << endl;
12.
13.
14.
15.    return 0;
16. }

```

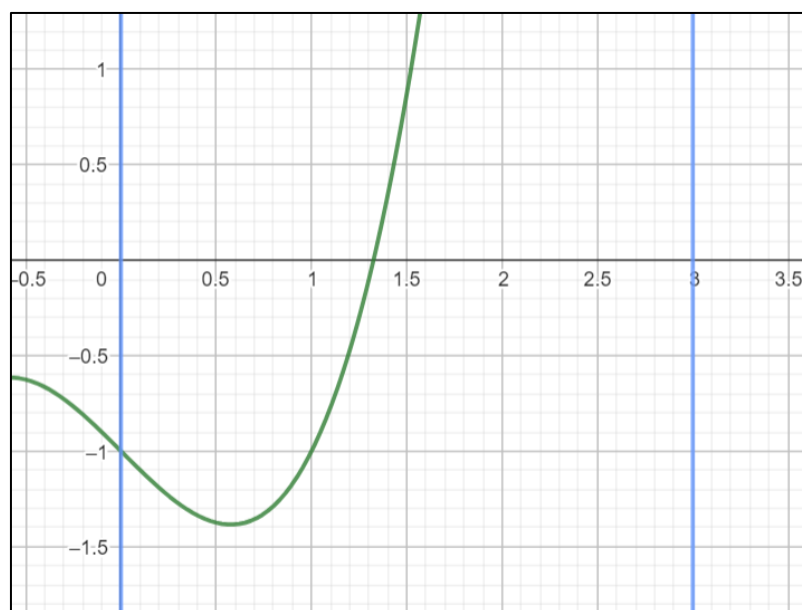
Fragment kodu 4: funkcja main

## 4. Testy jednostkowe

### 4.1. Testy dla funkcji *bisection\_method*

**4.1.1. Test na poprawność znalezienia pierwiastka w przedziale zawierającym jeden pierwiastek.**

Do tego testu obrano funkcję  $f(x) = x^3 - x - 1$  i operowano na przedziale  $[0, 3]$ , na którym funkcja ta zawiera dokładnie jeden pierwiastek. Za dokładność *epsilon* obrano wartość 0.0001. Funkcję zwizualizowano w programie Geogebra i ograniczono przedziałem (Rys. 1a), a spodziewane wyniki otrzymano w programie PlanetCalc (Rys. 1c). Wydruk z konsoli załączono na Rys. 1b.



Rys. 1a. Wizualizacja funkcji oraz przedziału w programie Geogebra

```
bisection method: 1.32468
```

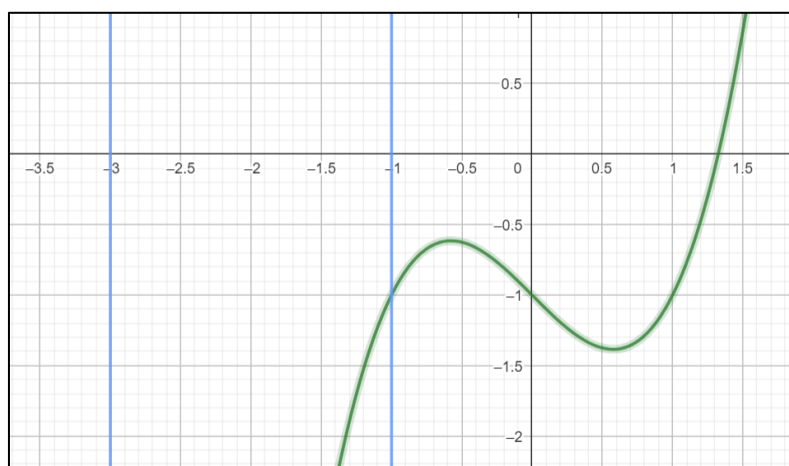
Rys. 1b. Wydruk z konsoli

x  
1.3247

Rys. 1c. Wynik metody bisekcji dla zadanych danych  
obliczony w programie PlanetCalc

#### 4.1.2. Test na obsługę braku pierwiastka w zadanym przedziale.

Do tego testu obrano funkcję  $f(x) = x^3 - x - 1$  i operowano na przedziale  $[-3, -1]$ , na którym funkcja ta nie ma żadnego pierwiastka. Za dokładność *epsilon* obrano wartość 0.0001. Funkcję zwizualizowano w programie Geogebra i ograniczono przedziałem (Rys. 2a). Wydruk z konsoli załączono na Rys. 2b.



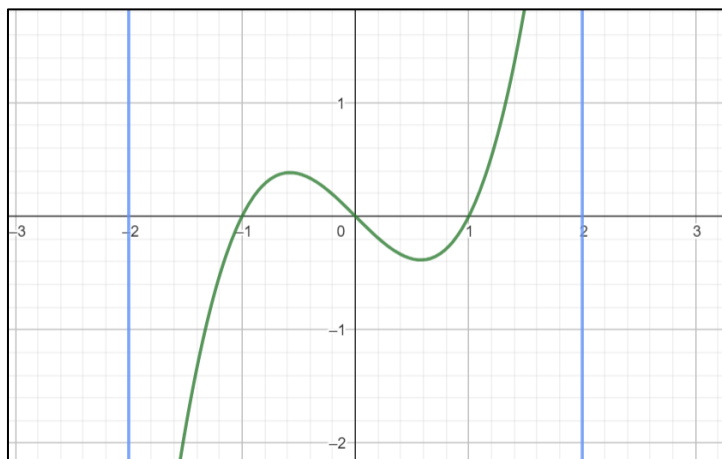
Rys. 2a. Wizualizacja funkcji oraz przedziału w programie Geogebra

```
bisection method: no root in given interval.  
0
```

Rys. 2b. Wydruk z konsoli

### 4.1.3. Test na obsługę przedziału zawierającego więcej niż jeden pierwiastek

Do tego testu obrano funkcję  $f(x) = x^3 - x$  i operowano na przedziale  $[-2, 2]$ , na którym funkcja ta zawiera dokładnie trzy pierwiastki. Za dokładność *epsilon* obrano wartość 0.0001. Funkcję zwizualizowano w programie Geogebra i ograniczono przedziałem (Rys. 3a), a spodziewane wyniki otrzymano w programie PlanetCalc (Rys. 3c). Wydruk z konsoli załączono na Rys. 3b.



Rys. 3a. Wizualizacja funkcji oraz przedziału w programie Geogebra

```
bisection method: 0
```

Rys. 3b. Wydruk z konsoli

<div>x</div> <div>0</div>
---------------------------

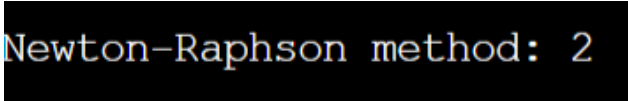
Rys. 3c. Wynik metody bisekcji dla zadanych danych obliczony w programie PlanetCalc

## 4.2. Testy dla funkcji *Newton\_Raphson\_method*

### 4.2.1. Test na poprawność znalezienia pierwiastka znanego równania

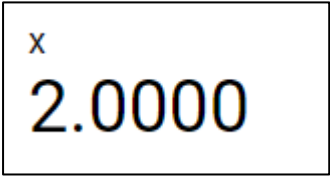
Do tego testu obrano funkcję  $f(x) = x^2 - 4$  i operowano na wartości  $x_0$  równej 1.5 oraz wartości  $h$  równej 0.0001. Za dokładność *epsilon* obrano wartość 0.0001. Spodziewane wyniki otrzymano w programie PlanetCalc (Rys. 4b). Wydruk z konsoli załączono na Rys. 4a.





```
Newton-Raphson method: 2
```

Rys. 4a. Wydruk z konsoli

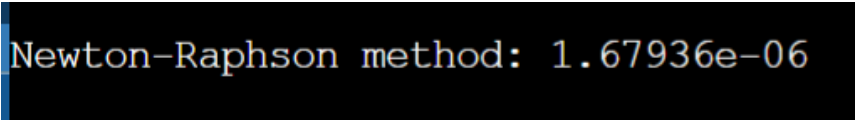


```
x
2.0000
```

Rys. 4b. Wynik metody Newtona-Raphsona dla zadanych danych  
obliczony w programie PlanetCalc

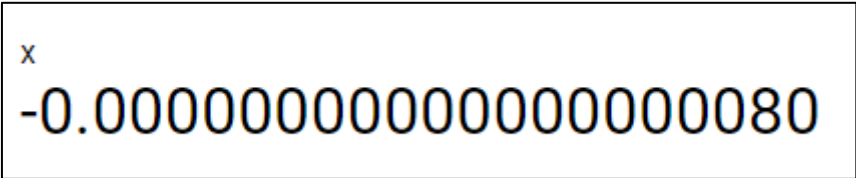
#### 4.2.2. Test na obsługę funkcji o różnych kształtach

Do tego testu obrano funkcję  $f(x) = \sin(x) + x$  i operowano na wartości  $x_0$  równej 0.5 oraz wartości  $h$  równej 0.0001. Za dokładność *epsilon* obrano wartość 0.0001. Spodziewane wyniki otrzymano w programie PlanetCalc (Rys. 5b). Wydruk z konsoli załączono na Rys. 5a.



```
Newton-Raphson method: 1.67936e-06
```

Rys. 5a. Wydruk z konsoli

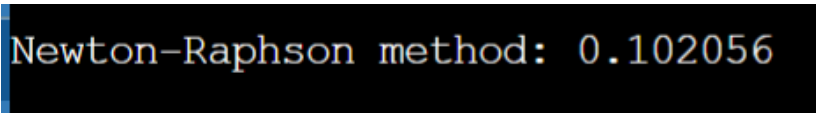


```
x
-0.00000000000000000000000080
```

Rys. 5b. Wynik metody Newtona-Raphsona dla zadanych danych  
obliczony w programie PlanetCalc

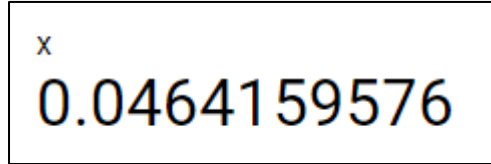
#### 4.2.3. Test na obsługę funkcji o bardzo płaskim grzbiecie

Do tego testu obrano funkcję  $f(x) = x^4 - 0,0001x$  i operowano na wartości  $x_0$  równej 1 oraz wartości  $h$  równej 0.0001. Za dokładność *epsilon* obrano wartość 0.0001. Spodziewane wyniki otrzymano w programie PlanetCalc (Rys. 6b). Wydruk z konsoli załączono na Rys. 6a.



```
Newton-Raphson method: 0.102056
```

Rys. 6a. Wydruk z konsoli



```
x
0.0464159576
```

Rys. 6b. Wynik metody Newtona-Raphsona dla zadanych danych  
obliczony w programie PlanetCalc

## 5. Opracowanie wyników

W pierwszym teście metoda bisekcji (*Tabela 1*) zwróciła wartość  $1.32468$ , która jest bardzo bliska oczekiwanej wartości  $1.3247$ , co sugeruje poprawne działanie algorytmu dla przedziału zawierającego jeden pierwiastek. W przypadku, gdy przedział zawierał tylko jeden pierwiastek, metoda bisekcji działała poprawnie, zwracając wynik bardzo bliski oczekiwanej wartości.

W drugim teście, gdzie nie ma pierwiastka w zadanym przedziale, metoda bisekcji poprawnie zwróciła komunikat *"No root in given interval"*, co potwierdza jej poprawne zachowanie w takiej sytuacji. Gdy zadany przedział nie zawierał żadnego pierwiastka, metoda bisekcji poprawnie zwróciła komunikat informujący o braku pierwiastka. Jest to istotne zachowanie, ponieważ sygnalizuje użytkownikowi, że poszukiwany pierwiastek nie istnieje w zadanym przedziale.

W trzecim teście, gdzie przedział zawiera trzy pierwiastki, otrzymana wartość  $0$  jest zgodna z oczekiwaną, co sugeruje poprawne działanie metody w tym przypadku. W przypadku przedziału zawierającego trzy pierwiastki, metoda bisekcji zwróciła wartość zerową, co jest zgodne z oczekiwaniami, ponieważ nie jest możliwe jednoznaczne określenie pierwiastka. To pokazuje, że metoda może zwrócić dowolną wartość z przedziału, w którym istnieje więcej niż jeden pierwiastek.

**Tabela 1:** Zestawienie wyników dla metody bisekcji

Test	Oczekiwana	Otrzymana
Test na przedziale z jednym pierwiastkiem	1,3247	1,32468
Test na przedziale bez pierwiastków	brak	No root in given interval
Test na przedziale z trzema pierwiastkami	0	0

Metoda Newtona-Raphsona skutecznie znalazła pierwiastek znanego dla testu pierwszego, zwracając oczekiwaną wartość 2 (*Tabela 2*). Potwierdza to poprawność działania algorytmu dla przypadków, gdzie pierwiastek jest dobrze określony.

W przypadku funkcji o różnych kształtach, metoda Newtona-Raphsona znalazła pierwiastek, ale otrzymana wartość różniła się od oczekiwanej. Może to sugerować, że algorytm może napotykać trudności w zbieżności dla funkcji o skomplikowanych kształtach lub o ekstremalnie małych wartościach.

Dla funkcji o bardzo płaskim grzbiecie, metoda Newtona-Raphsona znalazła pierwiastek, ale otrzymana wartość była nieco różna od oczekiwanej. To sugeruje, że metoda może napotykać trudności w zbieżności dla funkcji o bardzo płaskim grzbiecie, gdzie wartości pochodnych są bliskie zeru.

**Tabela 2:** Zestawienie wyników dla metody Newtona-Raphsona

Test	Oczekiwana	Otrzymana
Test dla pierwiastka znanego równania	2	2
Test dla funkcji o różnych kształtach	$8 \cdot 10^{-21}$	$1,67936 \cdot 10^{-6}$
Test dla funkcji o bardzo płaskim grzbiecie	0,0464159576	0,102056

## 6. Wnioski

Zarówno metoda bisekcji, jak i metoda Newtona-Raphsona okazały się skuteczne w znajdowaniu pierwiastków funkcji nieliniowych w odpowiednich warunkach. Obie metody potrafiły znaleźć pierwiastki znanych równań oraz funkcji o prostych kształtach.

Metoda bisekcji jest prostsza i bardziej odporna na trudne przypadki, ale jej głównym ograniczeniem jest to, że wymaga, aby funkcja miała przeciwnego znaku na krańcach

przedziału. W przypadku funkcji o oscylujących pierwiastkach lub funkcji z płaskim grzbietem, metoda bisekcji może być mniej skuteczna.

Metoda Newtona-Raphsona, z kolei, jest bardziej elastyczna i zbieżna, co pozwala na szybsze znalezienie pierwiastków, zwłaszcza w przypadku funkcji o znanych pochodnych i dobrze określonych punktach startowych. Jednakże, może ona napotkać trudności w przypadku funkcji o ekstremalnie małych wartościach lub o bardzo płaskim grzbiecie.

W przypadku metody Newtona-Raphsona wybór odpowiedniego punktu startowego  $x_0$  ma kluczowe znaczenie dla zbieżności. W przypadku funkcji o skomplikowanych kształtach lub ekstremalnych wartościach, wybór niewłaściwego punktu startowego może prowadzić do braku zbieżności.

Wybór odpowiedniej metody zależy od charakterystyki funkcji oraz wymagań dotyczących dokładności i czasu obliczeń. W przypadku funkcji o prostych kształtach lub gdy dokładność nie jest kluczowa, metoda bisekcji może być wystarczająco skuteczna i łatwa do zrozumienia. Natomiast, w przypadku funkcji o skomplikowanych kształtach lub gdy wymagana jest wysoka dokładność, metoda Newtona-Raphsona może być bardziej odpowiednia, ale wymaga starannego doboru punktu startowego i uwzględnienia potencjalnych trudności zbieżności.

## 7. Źródła

Wykłady z Metod Numerycznych autorstwa dr hab. Danuty Szeligi

Prezentacja Metody Numeryczne. Rozwiązywanie równań nieliniowych – metoda bisekcji, metoda Newtona - Raphsona autorstwa dr. Hab. Inż. Marcina Hojnego.

Wikipedia – artykuły o równaniach nieliniowych, nieliniowości, metodzie bisekcji, metodzie Newtona – Raphsona.

Program PlanetCalc: <https://planetcalc.com/3718/>

<https://planetcalc.com/7748/>

Program Geogebra: <https://www.geogebra.org/calculator>