


Paulina Grabowska 	Metody numeryczne
<p style="text-align: center;">Sprawozdanie 10</p> <p style="text-align: center;">Rozwiązywanie równań różniczkowych – metoda Eulera, Heuna, RK4</p>	

1. Wstęp

Równania różniczkowe to matematyczne równania, które opisują, jak zmieniają się wartości funkcji w zależności od jej pochodnych. Stanowią one fundament wielu dziedzin nauki i techniki, służąc do modelowania dynamicznych zjawisk w przyrodzie, inżynierii, ekonomii, biologii i innych obszarach. Dzięki równaniom różniczkowym można przewidywać i analizować zmiany zachodzące w systemach w czasie lub przestrzeni.

Podstawowe typy równań różniczkowych to równania różniczkowe zwyczajne i równania różniczkowe cząstkowe. Równania różniczkowe zwyczajne dotyczą funkcji jednej zmiennej i ich pochodnych. Przykładem może być równanie opisujące ruch obiektu pod wpływem siły. Z kolei równania różniczkowe cząstkowe dotyczą funkcji wielu zmiennych i ich pochodnych cząstkowych. Są stosowane do opisu bardziej złożonych zjawisk, takich jak przepływ ciepła czy ruch fal.

2. Opracowania metod numerycznych

2.1. Metoda Eulera

Metoda Eulera to jedna z najprostszych i najstarszych metod numerycznych stosowanych do rozwiązywania równań różniczkowych zwyczajnych. Jest ona stosunkowo łatwa do zrozumienia i implementacji, co czyni ją popularnym wprowadzeniem do numerycznych metod rozwiązywania równań różniczkowych.

Metoda Eulera służy do aproksymacji rozwiązań równań różniczkowych pierwszego rzędu postaci:

$$\frac{dy}{dx} = f(x, y) \tag{1}$$

gdzie $y = y(x)$ jest szukaną funkcją, a $f(x, y)$ jest znaną funkcją określającą pochodną y względem x .

Metoda Eulera polega na przybliżeniu rozwiązania równania różniczkowego przez dyskretyzację zmiennej x na małe kroki h i przybliżeniu wartości y w tych punktach. Główna idea to wykorzystanie wartości pochodnej, aby oszacować wartość funkcji w następnym kroku.

Równanie iteracyjne metody Eulera można zapisać jako:

$$y_{n+1} = y_n + h \cdot f(x_n, y_n) \quad (2)$$

gdzie:

- y_n jest wartością funkcji y w punkcie x_n
- h jest krokiem dyskretyzacji,
- $f(x_n, y_n)$ jest wartością pochodnej w punkcie (x_n, y_n)

Krok N oblicza się ze wzoru:

$$N = \frac{b - x_0}{h} \quad (3)$$

Zaletami metody Eulera jest jej prostota i łatwość implementacji, a dla dużych kroków h może być mało dokładna, a dla źle dobranego h może być niestabilna.

2.2. Metoda Heuna

Metoda Heuna, znana również jako metoda poprawiona Eulera, jest jedną z metod numerycznego rozwiązywania równań różniczkowych zwyczajnych. Jest to metoda drugiego rzędu, co oznacza, że jest dokładniejsza niż podstawowa metoda Eulera. Metoda Heuna wykorzystuje korekcję przewidywania dokonanej na podstawie metody Eulera, aby uzyskać lepsze przybliżenie rozwiązania.

$$y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_n + h, y_n + hf(x_n, y_n))] \quad (4)$$

Polega na podziale każdego kroku czasowego na dwa etapy. Na pierwszym etapie, zw. "krokiem przewidywania", używamy metody Eulera, aby oszacować wartość funkcji w następnym punkcie. Następnie, w "kroku poprawki", wykorzystujemy średnią wartość nachylenia na

początku i na końcu przedziału czasowego, aby skorygować to przybliżenie. Liczbę kroków w N dla tej metody oblicza się ze wzoru (3).

2.3. Metoda RK4

Metoda Rungego-Kutty czwartego rzędu jest jedną z najpopularniejszych metod numerycznego rozwiązywania równań różniczkowych. Jest to bardziej zaawansowana technika niż metoda Eulera lub Heuna, która zapewnia jeszcze większą dokładność kosztem nieco większej złożoności obliczeniowej.

W metodzie RK4, wartość funkcji w kolejnym punkcie obliczana jest na podstawie czterech gradientów (pochodnych) w różnych punktach w danym przedziale czasowym. Pierwszy gradient służy do wyznaczenia wstępnego przybliżenia wartości funkcji w połowie kroku czasowego. Na podstawie wstępnego przybliżenia obliczany jest drugi gradient w połowie kroku czasowego. Kolejny gradient obliczany jest również w połowie kroku, wykorzystując drugie przybliżenie. Na podstawie trzeciego przybliżenia obliczana jest ostateczna wartość funkcji w następnym punkcie.

$$\begin{aligned}k_1 &= hf(x_i, y_i) \\k_2 &= hf(x_i + 0,5h, y_i + 0,5k_1) \\k_3 &= hf(x_i + 0,5n, y_i + 0,5k_2) \\k_4 &= hf(x_i + n, y_i + k_3)\end{aligned}\tag{5}$$

Wyliczone gradienty wstawiane są do wzoru:

$$y_{i+1} = y_i + (k_1 + 2k_2 + 2k_3 + k_4) \frac{1}{6}\tag{6}$$

gradienty są wykorzystywane do iteracyjnego przemieszczania się wzdłuż krzywej rozwiązania równania różniczkowego, umożliwiając coraz bardziej dokładne przybliżenie wartości funkcji w kolejnych punktach czasowych. Dzięki temu metoda RK4 oferuje wysoką dokładność przy stosunkowo niewielkim nakładzie obliczeniowym.

3. Implementacja numeryczna

3.1. funkcje ogólne

3.1.1. funkcja *main* (Fragment kodu 1)

W funkcji głównej inicjalizowane są wartości początkowe parametrów x_0 , y_0 , h oraz b (końcowy punkt). Następnie inicjalizowana jest wartość N będąca konwersją typu *static_cast*, która obliczana jest na podstawie długości przedziału ze wzoru (3). Następnie deklarowane są tablice x_arr oraz y_arr dla każdej metody z osobna, oraz wywoływane są funkcje *euler*, *heun* oraz *rk4*. Po wywołaniu funkcji za pomocą pętli wszystkie wartości x i y dla każdego punktu są wypisywane na ekran konsoli. Dla każdej z metod.

```
1. int main() {
2.     double x0 = 0;
3.     double y0 = 2;
4.     double h = 0.1;
5.     double b = 0.3;
6.     int N = static_cast<int>((b - x0) / h);
7.
8.     double x_arr_euler[N + 1];
9.     double y_arr_euler[N + 1];
10.    double x_arr_heun[N + 1];
11.    double y_arr_heun[N + 1];
12.    double x_arr_rk4[N + 1];
13.    double y_arr_rk4[N + 1];
14.
15.    euler(x0, y0, h, N, x_arr_euler, y_arr_euler);
16.    heun(x0, y0, h, N, x_arr_heun, y_arr_heun);
17.    rk4(x0, y0, h, N, x_arr_rk4, y_arr_rk4);
18.
19.    cout << "Euler:\n";
20.    cout << "x\t\t y\n";
21.    for (int i = 0; i <= N; ++i) {
22.        cout << x_arr_euler[i] << "\t " << y_arr_euler[i] << endl;
23.    }
24.
25.    cout << "\nHeun:\n";
26.    cout << "x\t\t y\n";
27.    for (int i = 0; i <= N; ++i) {
28.        cout << x_arr_heun[i] << "\t " << y_arr_heun[i] << endl;
29.    }
30.
31.    cout << "\nRK4:\n";
32.    cout << "x\t\t y\n";
33.    for (int i = 0; i <= N; ++i) {
34.        cout << x_arr_rk4[i] << "\t " << y_arr_rk4[i] << endl;
35.    }
36.
37.    return 0;
38. }
```

Fragment kodu 1: funkcja *main*

3.1.2. funkcja *function* (Fragment kodu 2)

Definicja funkcji *function*, która przyjmuje dwa argumenty typu *double* i zwraca *double*. W tej funkcji definiowana jest funkcja, którą będziemy różniczkować w dalszej części programu.

```
1. double function(double x, double y) {  
2.     return y;  
3. }
```

Fragment kodu 2: Funkcja *function*

3.2. funkcja *euler* (Fragment kodu 3)

Definicja funkcji *euler*, która implementuje metodę Eulera do rozwiązywania równań różniczkowych. Funkcja ta nie zwraca żadnej wartości i przyjmuje sześć argumentów:

- *double x0*: Początkowa wartość zmiennej niezależnej, np. początkowy punkt w przedziale czasu.
- *double y0*: Początkowa wartość funkcji zależnej, np. wartość funkcji w punkcie początkowym.
- *double h*: Krok krokowy, czyli odległość między kolejnymi punktami na osi niezależnej.
- *int N*: Liczba kroków, określająca ile razy metoda Eulera ma zostać zastosowana.
- *double* x_arr*: Wskaźnik do tablicy, w której będą przechowywane wartości zmiennej niezależnej.
- *double* y_arr*: Wskaźnik do tablicy, w której będą przechowywane wartości funkcji zależnej.

Deklarowane są tablice *x_arr* oraz *y_arr* oraz ich wartości początkowe zostają ustawione odpowiednio na *x0* oraz *y0*. Wartości *x_arr* i *y_arr* są uzupełniane przez funkcję *euler* w trakcie działania, aby zawierały kolejne przybliżone wartości zmiennej niezależnej i odpowiadające im wartości funkcji zależnej na tych punktach.

Następnie pętla *for* iteruje od 1 do *N* i w niej obliczane są kolejne wartości *x* poprzez dodanie kroku *h* do poprzedniej wartości *x*. Obliczane są również kolejne wartości *y*, korzystając ze wzoru (2) i odwołując się do funkcji *function*.

```

1. void euler(double x0, double y0, double h, int N, double* x_arr, double* y_arr) {
2.     x_arr[0] = x0;
3.     y_arr[0] = y0;
4.
5.     for (int i = 1; i <= N; ++i) {
6.         x_arr[i] = x_arr[i - 1] + h;
7.         y_arr[i] = y_arr[i - 1] + h * function(x_arr[i - 1], y_arr[i - 1]);
8.     }
9. }

```

Fragment kodu 3: Funkcja *euler*

3.3. funkcja *heun* (Fragment kodu 4)

Funkcja *heun* implementuje metodę Heuna, która jest jedną z metod numerycznych używanych do rozwiązywania równań różniczkowych. W skrócie, ta metoda polega na iteracyjnym obliczaniu kolejnych wartości funkcji dla różnych punktów czasowych.

Najpierw, na podstawie warunków początkowych x_0 i y_0 , ustawiane są pierwsze wartości x i y w odpowiednich tablicach x_arr i y_arr .

Następnie, pętla rozpoczyna iterację po kolejnych krokach czasowych, zaczynając od drugiego punktu. Dla każdego kroku czasowego obliczana jest wartość y za pomocą metody Eulera, czyli dodając do poprzedniego y iloczyn kroku czasowego h i wartość funkcji $function(x_prev, y_prev)$. Ta wartość jest zapisywana jako y_euler .

Obliczana jest poprawiona wartość y za pomocą metody Heuna. Wykorzystuje się średnią wartość nachylenia na początku i na końcu przedziału czasowego. Ta poprawiona wartość y jest zapisywana w tablicy y_arr . Następnie aktualizowany jest x dla kolejnego kroku, dodając do poprzedniego x wartość kroku czasowego h . Cały proces jest powtarzany, aż zostaną osiągnięte wszystkie punkty czasowe.

```

1. void heun(double x0, double y0, double h, int N, double* x_arr, double* y_arr) {
2.     x_arr[0] = x0;
3.     y_arr[0] = y0;
4.
5.     for (int i = 1; i <= N; ++i) {
6.         double x_prev = x_arr[i - 1];
7.         double y_prev = y_arr[i - 1];
8.         double y_euler = y_prev + h * function(x_prev, y_prev);
9.         x_arr[i] = x_prev + h;
10.        y_arr[i] = y_prev + (h / 2) * (function(x_prev, y_prev) + function(x_arr[i],
y_euler));
11.    }
12. }

```

Fragment kodu 4: funkcja *heun*

3.4. funkcja *rk4* (Fragment kodu 5)

Funkcja *rk4* implementuje metodę Rungego-Kutty czwartego rzędu (RK4) do rozwiązywania równań różniczkowych numerycznie.

Najpierw, dla warunków początkowych x_0 i y_0 , ustawiane są pierwsze wartości x i y w odpowiednich tablicach x_arr i y_arr . Następnie, dla każdego kroku czasowego, pętla wykonuje następujące kroki: obliczane są cztery gradienty, zgodnie ze wzorem (5) na podstawie funkcji opisującej równanie różniczkowe oraz wartości x i y z poprzedniego kroku, a także aktualizowana jest wartość x dla kolejnego kroku czasowego.

Nowa wartość y jest obliczana na podstawie czterech gradientów według wzoru (6), który uwzględnia ich średnią ważoną. Proces ten powtarzany jest dla każdego kolejnego kroku czasowego. Po zakończeniu pętli, tablice x_arr i y_arr zawierają wartości x i y dla kolejnych punktów czasowych.

```
1. void rk4(double x0, double y0, double h, int N, double* x_arr, double* y_arr) {
2.     x_arr[0] = x0;
3.     y_arr[0] = y0;
4.
5.     for (int i = 1; i <= N; ++i) {
6.         double x_prev = x_arr[i - 1];
7.         double y_prev = y_arr[i - 1];
8.         double k1 = h * function(x_prev, y_prev);
9.         double k2 = h * function(x_prev + h / 2, y_prev + k1 / 2);
10.        double k3 = h * function(x_prev + h / 2, y_prev + k2 / 2);
11.        double k4 = h * function(x_prev + h, y_prev + k3);
12.        x_arr[i] = x_prev + h;
13.        y_arr[i] = y_prev + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
14.    }
15. }
```

Fragment kodu 5: funkcja *rk4*

Wykonano testy na zajęciach celem potwierdzenia poprawności zaimplementowanych algorytmów. Wszystkie otrzymane wyniki pokrywały się z oczekiwaniami, co pozwala stwierdzić, że funkcje zostały zaimplementowane poprawnie.

4. Testy jednostkowe

4.1. Test dla funkcji kwadratowej

Wprowadzone dane:

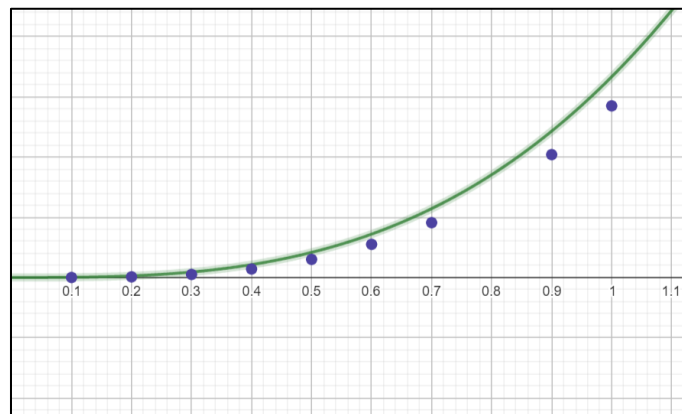
- $f(x,y) = x^2$;
- $x_0 = 0$;
- $y_0 = 0$;
- $b = 1$;
- $h = 0.2$.

4.1.1. Metoda Eulera

Wyniki sprawdzono w programie PlanetCalc specjalnie dedykowanym dla konkretnej metody. Na Rys. 1a załączono wydruk z ekranu konsoli po przeprowadzeniu testu. Sprawdzono aproksymację za pomocą programu eMathHelp i jej wyniki zwizualizowano w programie Geogebra (co przedstawiono na Rys. 1b).

Euler:		
x		y
0	0	
0.2	0	
0.4	0.008	
0.6	0.04	
0.8	0.112	
1	0.24	

Rys. 1a: Wydruk ekranu konsoli



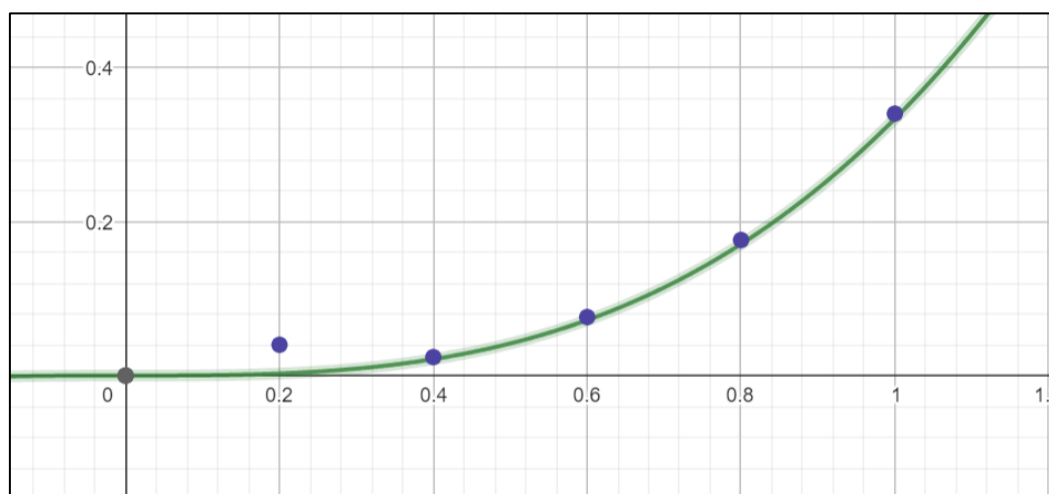
Rys. 1b. Wizualizacja rozwiązania oraz aproksymacji dla analitycznego rozwiązania $\frac{x^3}{3}$

4.2.2. Metoda Heuna

Wyniki sprawdzono w programie PlanetCalc specjalnie dedykowanym dla konkretnej metody. Na *Rys. 2a* załączono wydruk z ekranu konsoli po przeprowadzeniu testu. Sprawdzono aproksymację za pomocą programu eMathHelp i jej wyniki zwizualizowano w programie Geogebra (co przedstawiono na *Rys. 2b*).

Heun:		
x		y
0	0	
0.2	0.004	
0.4	0.024	
0.6	0.076	
0.8	0.176	
1	0.34	

Rys. 2a: Wydruk ekranu konsoli



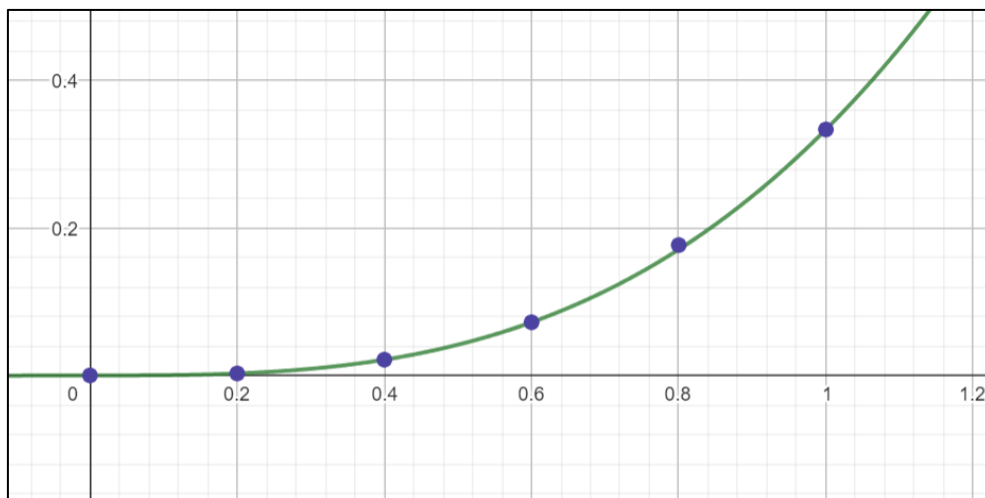
Rys. 2b. Wizualizacja rozwiązania oraz aproksymacji dla analitycznego rozwiązania $\frac{x^3}{3}$

4.2.3. Metoda RK4

Wyniki sprawdzono w programie PlanetCalc specjalnie dedykowanym dla konkretnej metody. Na *Rys. 3a* załączono wydruk z ekranu konsoli po przeprowadzeniu testu. Sprawdzono aproksymację za pomocą programu eMathHelp i jej wyniki zwizualizowano w programie Geogebra (co przedstawiono na *Rys. 3b*).

RK4:		
x	y	
0	0	
0.2	0.00266667	
0.4	0.0213333	
0.6	0.072	
0.8	0.170667	
1	0.333333	

Rys. 3a: Wydruk ekranu konsoli



Rys. 3b. Wizualizacja rozwiązania oraz aproksymacji dla analitycznego rozwiązania $\frac{x^3}{3}$

4.2. Test dla funkcji trygonometrycznej

Wprowadzone dane:

- $f(x,y) = \sin(x)$;
- $x_0 = 0$;
- $y_0 = 0$;
- $b = 6.28 (2\pi)$;
- $h = 1$.

4.2.1. Metoda Eulera

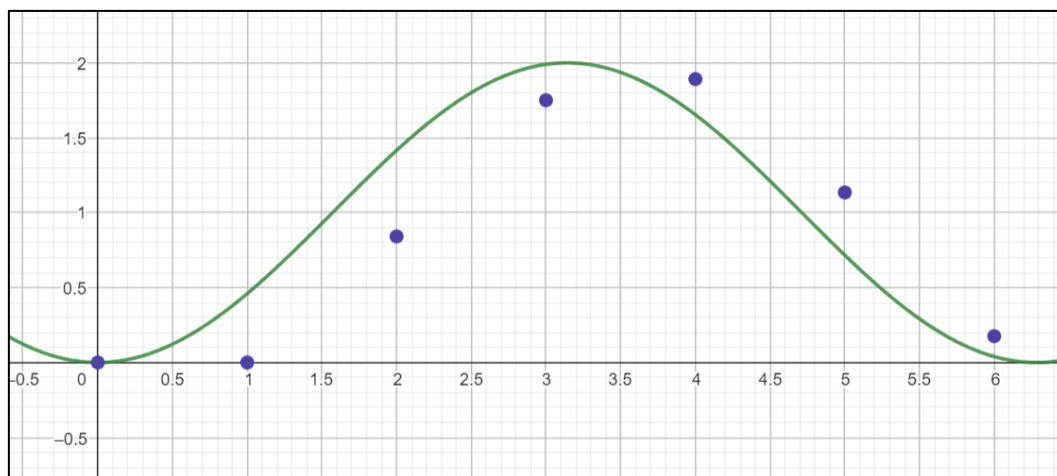
Wyniki sprawdzono w programie PlanetCalc specjalnie dedykowanym dla konkretnej metody.

Na Rys. 4a załączono wydruk z ekranu konsoli po przeprowadzeniu testu. Sprawdzono

aproxymację za pomocą programu eMathHelp i jej wyniki zwizualizowano w programie Geogebra (co przedstawiono na *Rys. 4b*).

Euler:		
x		y
0	0	
1	0	
2	0.841471	
3	1.75077	
4	1.89189	
5	1.13509	
6	0.176162	

Rys. 4a: Wydruk ekranu konsoli



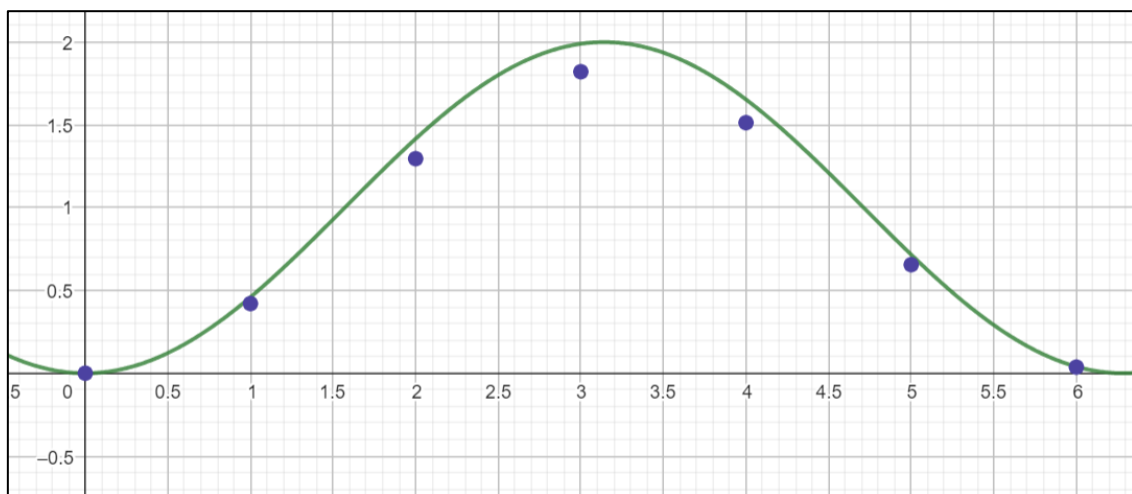
Rys. 4b. Wizualizacja rozwiązania oraz aproksymacji dla analitycznego rozwiązania $1 - \cos x$

4.2.2. Metoda Heuna

Wyniki sprawdzono w programie PlanetCalc specjalnie dedykowanym dla konkretnej metody. Na *Rys. 5a* załączono wydruk z ekranu konsoli po przeprowadzeniu testu. Sprawdzono aproxymację za pomocą programu eMathHelp i jej wyniki zwizualizowano w programie Geogebra (co przedstawiono na *Rys. 5b*).

Heun:	
x	y
0	0
1	0.420735
2	1.29612
3	1.82133
4	1.51349
5	0.655624
6	0.0364539

Rys. 5a: Wydruk ekranu konsoli



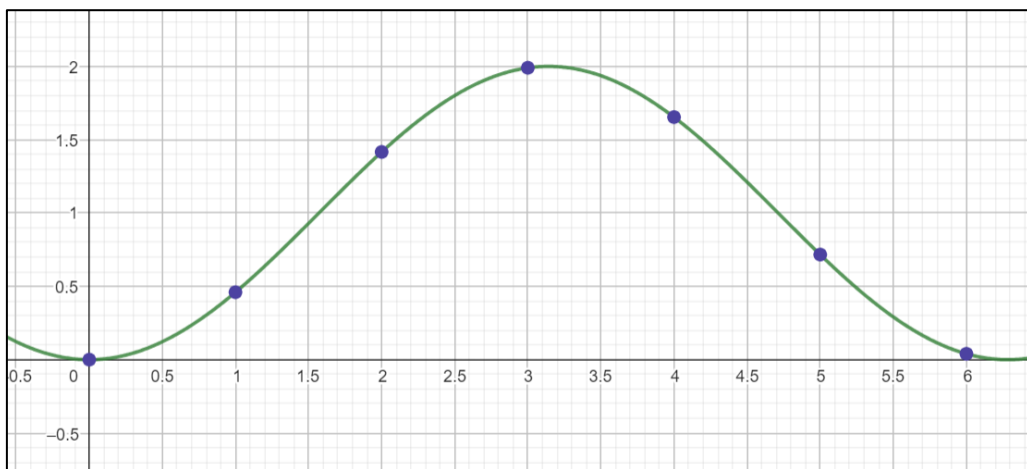
Rys. 5b. Wizualizacja rozwiązania oraz aproksymacji dla analitycznego rozwiązania $1 - \cos x$

4.2.3. Metoda RK4

Wyniki sprawdzono w programie PlanetCalc specjalnie dedykowanym dla konkretnej metody. Na *Rys. 6a* załączono wydruk z ekranu konsoli po przeprowadzeniu testu. Sprawdzono aproksymację za pomocą programu eMathHelp i jej wyniki zwizualizowano w programie Geogebra (co przedstawiono na *Rys. 6b*).

RK4:	
x	y
0	0
1	0.459862
2	1.41665
3	1.9907
4	1.65424
5	0.716594
6	0.039844

Rys. 6a: Wydruk ekranu konsoli



Rys. 6b. Wizualizacja rozwiązania oraz aproksymacji dla analitycznego rozwiązania $1 - \cos x$

4.3. Test dla funkcji logarytmicznej

Wprowadzone dane:

- $f(x,y) = \frac{1}{x}$;
- $x_0 = 0$;
- $y_0 = 1$;
- $b = 10$;
- $h = 0.1$.

4.3.1. Metoda Eulera

Wyniki sprawdzono w programie PlanetCalc specjalnie dedykowanym dla konkretnej metody. Na Rys. 7 załączono wydruk z ekranu konsoli po przeprowadzeniu testu. Sprawdzono

aproksymację za pomocą programu eMathHelp (otrzymano komunikat: *Some or all initial conditions are not in the domain of the solution*).

```
Euler:
x      y
0      1
0.5    inf
1      inf
1.5    inf
2      inf
2.5    inf
3      inf
```

Rys. 7: Wydruk ekranu konsoli

4.3.2 Metoda Heuna

Wyniki sprawdzono w programie PlanetCalc specjalnie dedykowanym dla konkretnej metody. Na Rys. 8 załączono wydruk z ekranu konsoli po przeprowadzeniu testu. Sprawdzono aproksymację za pomocą programu eMathHelp (otrzymano komunikat: *Some or all initial conditions are not in the domain of the solution*).

```
Heun:
x      y
0      1
0.5    inf
1      inf
1.5    inf
2      inf
2.5    inf
3      inf
```

Rys. 8: Wydruk ekranu konsoli

4.3.3. Metoda RK4

Wyniki sprawdzono w programie PlanetCalc specjalnie dedykowanym dla konkretnej metody. Na Rys. 9 załączono wydruk z ekranu konsoli po przeprowadzeniu testu. Sprawdzono

aproxymację za pomocą programu eMathHelp (otrzymano komunikat: *Some or all initial conditions are not in the domain of the solution*).

```

RK4:
x      y
0      1
0.5    inf
1      inf
1.5    inf
2      inf
2.5    inf
3      inf

```

Rys. 9: Wydruk ekranu konsoli

5. Opracowanie wyników

5.1. Wyniki testu dla funkcji kwadratowej

Wartości w kolumnach przedstawiają wyniki uzyskane za pomocą poszczególnych metod dla odpowiadających im wartości rzeczywistych (*Tabela 1*). Możemy zauważyć, że im bliżej wartości rzeczywistej, tym wyniki uzyskane za pomocą metod Eulera, Heuna i RK4 są bardziej zbliżone do rzeczywistych wartości. Metoda RK4 oferuje największą dokładność, co potwierdzają wyniki, które są najbardziej zbliżone do wartości rzeczywistej. Metoda Heuna osiąga lepszą dokładność niż metoda Eulera, co jest widoczne w mniejszych błędach dla większości wartości.

Tabela 1: Zestawienie wyników dla testu dla funkcji kwadratowej

	Wartość rzeczywista	Metoda Eulera	Metoda Heuna	Metoda RK4
0	0	0	0	0
0,2	0,0026666667	0,001	0,004	0,00266667
0,4	0,02133	0,014	0,024	0,0213333
0,6	0,072	0,055	0,076	0,072
0,8	0,1760667	0,140	0,176	0,170667
1	0,3333333	0,285	0,34	0,3333333

5.2. Wyniki testu dla funkcji trygonometrycznej

Wartości w kolumnach przedstawiają wyniki uzyskane za pomocą poszczególnych metod dla odpowiadających im wartości rzeczywistych (*Tabela 2*). Możemy zauważyć, że dla większości wartości, metoda RK4 osiąga największą dokładność, zbliżając się do wartości rzeczywistej. Metoda Heuna również osiąga dobre wyniki, choć nieco mniej dokładne niż metoda RK4. Metoda Eulera uzyskuje najmniejszą dokładność w porównaniu do pozostałych dwóch metod, co jest widoczne w większych błędach dla większości wartości.

Tabela 2: Zestawienie wyników dla testu dla funkcji trygonometrycznej

	Wartość rzeczywista	Metoda Eulera	Metoda Heuna	Metoda RK4
0	0	0	0	0
1	0,459696764	0,239713	0,420735	0,459862
2	1,41614683	1,1592	1,29612	1,41665
3	1,9899925	1,91308	1,82133	1,9907
4	1,6536436	1,80825	1,51349	1,65424
5	0,7163378	0,941083	0,655624	0,716594
6	0,0398297	0,10885	0,03369539	0,039844

5.3. Wyniki testu dla funkcji logarytmicznej

Wartości w kolumnach przedstawiają wyniki uzyskane za pomocą poszczególnych metod dla odpowiadających im wartości rzeczywistych (*Tabela 3*). Dla tych konkretnych wartości, wyniki wszystkich metod dążą do nieskończoności (*Inf*), co może oznaczać, że równanie różniczkowe logarytmiczne prowadzi do błędów lub niestabilności numerycznych dla tych punktów. W takich przypadkach konieczne może być zastosowanie innych metod numerycznych lub przemyslenie samego równania różniczkowego.

Tabela 3: Zestawienie wyników dla testu dla funkcji logarytmicznej.

	Wartość rzeczywista	Metoda Eulera	Metoda Heuna	Metoda RK4
0	1	1	1	1
0,5	Inf	inf	Inf	inf
1	Inf	Inf	Inf	Inf
1,5	Inf	Inf	Inf	Inf
2	Inf	Inf	Inf	Inf
2,5	Inf	Inf	Inf	Inf
3	Inf	Inf	Inf	Inf

6. Wnioski

Na podstawie opracowanych wyników dla wartości rzeczywistej oraz wyników uzyskanych za pomocą metod Eulera, Heuna i RK4, można wyciągnąć następujące wnioski:

Metoda RK4 oferuje największą dokładność. Dla większości wartości rzeczywistych, metoda RK4 osiąga najmniejsze błędy w porównaniu do metod Eulera i Heuna. Jest to zgodne z oczekiwaniami, ponieważ metoda RK4 jest bardziej zaawansowaną techniką numeryczną, która uwzględnia większą liczbę gradientów i oferuje wyższy stopień dokładności.

Metoda Heuna jest bardziej dokładna niż metoda Eulera. Porównując wyniki uzyskane za pomocą metod Heuna i Eulera, można zauważyć, że metoda Heuna daje zwykle mniejsze błędy. Metoda Heuna korzysta z korekty wstępnego przybliżenia uzyskanego za pomocą metody Eulera, co prowadzi do bardziej dokładnych wyników.

Dla pewnych wartości metody nie są stabilne numerycznie. Dla niektórych wartości, wszystkie metody prowadzą do wyników równych nieskończoności (*Inf*). Jest to sygnał, że równanie różniczkowe, które próbujemy rozwiązać, może prowadzić do błędów lub niestabilności numerycznych dla tych konkretnych punktów. W takich przypadkach konieczne może być zastosowanie innych metod numerycznych lub przemyślenie samego równania różniczkowego.

Ogólnie rzecz biorąc, dobór odpowiedniej metody numerycznej do rozwiązania równań różniczkowych zależy od oczekiwanej dokładności, stabilności numerycznej oraz złożoności obliczeniowej. Metoda RK4 jest zwykle preferowaną opcją ze względu na swoją wysoką dokładność, ale dla prostszych problemów metody Eulera i Heuna mogą być wystarczająco skuteczne przy mniejszym nakładzie obliczeniowym. Jednak należy zachować ostrożność i monitorować stabilność numeryczną, aby uniknąć błędnych wyników.

7. Źródła

Wykłady z Metod Numerycznych autorstwa dr hab. Danuty Szeli

Prezentacja Rozwiązywanie równań różniczkowych (cz.1) – metoda Eulera oraz Rozwiązywanie równań różniczkowych (cz.2) – metoda Heunego, metoda RK4 autorstwa dr. Hab. Inż. Marcina Hojnego.

Wikipedia – artykuły o równaniach różniczkowych, metodzie Eulera

Program PlanetCalc: <https://planetcalc.com/8389/>

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

Program eMathHelp: <https://www.emathhelp.net/en/calculators/differential-equations/differential-equation-calculator/?i=y%27%3Dsin%28x%29%2C+y%280%29%3D0>