Laboratorium **Metod Obliczeniowych** Wydział Elektrotechniki Automatyki i Informatyki Politechnika Świętokrzyska	
Studia: **Stacjonarne I stopnia**	Kierunek: **Informatyka**
Data wykonania: **11.12.2017**	Grupa: **3ID13B**
Imię i nazwisko: **Bartłomiej Osak**	
Numer ćwiczenia: **9**	Temat ćwiczenia: **Rozwiązywanie równań różniczkowych zwyczajnych**

1. Rozwiązywanie równań różniczkowych zwyczajnych metodą RK4 – program w MATLAB.

Polecenie: Stosując metodę Rungego-Kutty 4-go rzędu (RK4) wyznacz rozwiązanie równania różniczkowego:

$$y'(x) = x + \sin\left(\frac{y(x)+1}{\sqrt{13}}\right)$$

z warunkiem początkowym $y(0.2) = 1.1$ w przedziale $x \in [0.2; 1.2]$ dla $h = 0.1$.

Kod źródłowy:

```
function[wynik] = metoda_rungego(x0,xn, y0, h)
    f = inline(input('Podaj rownanie funkcji f(y,t):','s'));
    x = x0:0.1:xn;
    n = size(x,2);
    xi = zeros(n,1);
    yi = zeros(n,1);
    xi(1) = x0;
    yi(1) = y0;
    disp([xi(1),yi(1)]);
    for i = 2:n
        k1 = h * f(xi(i-1), yi(i-1));
        k2 = h * f(xi(i-1) + 0.5*h, yi(i-1) + 0.5*k1);
        k3 = h * f(xi(i-1) + 0.5*h, yi(i-1) + 0.5*k2);
        k4 = h * f(xi(i-1) + h, yi(i-1) + k3);
        xi(i) = xi(i-1) + h;
        yi(i) = yi(i-1)+(1/6) * (k1 + 2*k2 + 2*k3 + k4);
        disp([xi(i),yi(i)]);
    end
    wynik = yi(n);
    plot(xi,yi,'o',xi,yi);
    xlabel('x');
    ylabel('y');
    grid on;
end
```

Wywołanie:

- **Uruchomienie funkcji metoda_rungego z parametrami: dolna i górna granica przedziału argumentu x, wartość warunku początkowego oraz wartość skoku:**

  ```
  metoda_rungego(0.2,1.2,1.1,0.1)
  ```

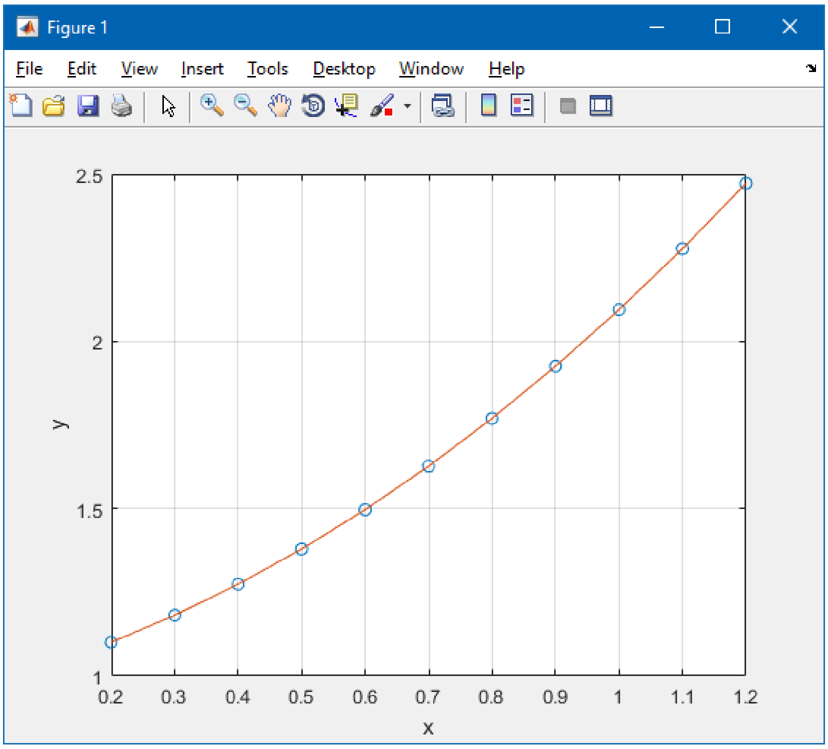
- **Wynik zadziałania:**

  ```
  Podaj rownanie funkcji f(y,t):x+sin((y+1)/sqrt(13))
      0.2000    1.1000
      0.3000    1.1809
      0.4000    1.2738
      0.5000    1.3789
      0.6000    1.4965
      0.7000    1.6267
      0.8000    1.7697
      0.9000    1.9257
      1.0000    2.0948
      1.1000    2.2770
      1.2000    2.4725
  ans =
      2.4725
  ```

Zrzut ekranu z działania programu:

```
Command Window
>> metoda_rungego(0.2,1.2,1.1,0.1)
Podaj rownanie funkcji f(y,t):x+sin((y+1)/sqrt(13))
    0.2000    1.1000

    0.3000    1.1809

    0.4000    1.2738

    0.5000    1.3789

    0.6000    1.4965

    0.7000    1.6267

    0.8000    1.7697

    0.9000    1.9257

    1.0000    2.0948

    1.1000    2.2770

    1.2000    2.4725


ans =

    2.4725
```

Wykres krzywej całkowej $y = y(x)$ wygenerowany również przez funkcję metoda_rungego(x0,xn,y0,h):



Opis programu:

Funkcja metoda_rungego przyjmuje cztery argumenty wywołania. Są to kolejno:

- x0 – dolna granica przedziału argumentów x
- xn – górna granica przedziału argumentów x
- y0 – wartość początkowa $y(x_0)$
- h – wartość skoku

Po prawidłowym wywołaniu funkcji następuje prośba o wpisanie równania różniczkowania, dla którego chcemy policzyć wartość. Następnie do wektora x zapisujemy sekwencję liczb generowaną od dolnej granicy przedziału x do górnej granicy przedziału x ze skokiem równym h. Do zmiennej n zapisywana jest długość wektora x. Wartość ta jest nam potrzebna w celu wykonania prawidłowej ilości iteracji celem wyznaczenia punktów $y = y(x)$ krzywej całkowej, a ostatecznie wyniku dla tego równania. Następnie inicjujemy wektory xi oraz yi wartościami zerowymi stosując funkcję zeros. Ich wielkość to 1 kolumna z n wierszami ($n = size(x, 2)$). Dalej następuje przypisanie do pierwszego wiersza wektora xi wartości dolnej granicy przedziału x, czyli argumentu wejściowego x0. Analogiczna śsytuacja odnosi się do wektora yi, gdzie przypisujemy wartość y0, czyli również operand wywołania funkcji. Następnie definiowana jest pętla, która wykonuje iterację od wartości 2 do wartości n ($size(x, 2)$). W pętli dążymy do obliczania kolejnych wartości y_i szukanej krzywej całkowej $y = y(x)$. Wartość y_{i+1} jest wyznaczana na podstawie wzoru:

$$y_{i+1} = y_i + \Delta y_i$$

gdzie:

$$\Delta y_i = \frac{1}{6}\left(k_1^{(i)} + 2k_2^{(i)} + 2k_3^{(i)} + k_4^{(i)}\right)$$

gdzie:

$$k_1^{(i)} = h * f(x_i, y_i)$$
$$k_2^{(i)} = h * f\left(x_i + 0.5 * h, y_i + 0.5 * k_1^{(i)}\right)$$
$$k_3^{(i)} = h * f\left(x_i + 0.5 * h, y_i + 0.5 * k_2^{(i)}\right)$$
$$k_4^{(i)} = h * f\left(x_i + h, y_i + k_3^{(i)}\right)$$

Powyższe wzory zostały zaimplementowane w pętli. Istotne jest to, iż wartości $k_n^{(i)}, n = 1,2,3,4, \dots$ są różne dla każdej iteracji i.

Wartość x_{i+1} wyznacza jest na podstawie wzoru:

$$x_i = x_0 + i * h$$

Opisując działania w pętli: na początku do zmiennych k1,k2,k3,k4 przypisywane są operacje zgodnie ze wzorami opisanymi wcześniej. Dla wszystkich k jest to iloczyn skoku, czyli wartości h oraz wywołania równania różniczkowego dla argumentu x oraz y, które dla poszczególnych k są odpowiednio rozszerzane. Następna wartość xi przechowywana jest w wektorze i wyznaczana poprzez sumę poprzedniego elementu x znajdującego się w wektorze i kroku h. Dalej do wektora yi przypisywana jest następna wartość y, która jest obliczana na podstawie wzoru powyżej. Jest to suma poprzedniej wartości y, (czyli dla tej, dla której wywoływane jest równanie różniczkowe w trakcie obliczania poszczególnych wartości k-tych) oraz delty yi, która określona jest poprzez iloraz sumy pojedynczego elementu k1, k2 i podwojonego elementu k2, k3 przez wartość 6. Na końcu wypisywany jest wynik w postaci x – y oraz rysowany jest wykres uzyskanej krzywej całkowej.

2. Rozwiązywanie równań różniczkowych zwyczajnych metodą RK4 – program w języku Java.

Polecenie: Stosując metodę Rungego-Kutty 4-go rzędu (RK4) wyznacz rozwiązanie równania różniczkowego:

$$y'(x) = x + \sin\left(\frac{y(x) + 1}{\sqrt{13}}\right)$$

z warunkiem początkowym $y(0.2) = 1.1$ w przedziale $x \in [0.2; 1.2]$ dla $h = 0.1$.

Kod źródłowy:

```java
import java.text.DecimalFormat;
import java.util.Scanner;

public class MethodRungeKutty {

    private static double[][] xi;
    private static double[][] yi;
    private static double x0;
    private static double xn;
    private static double y0;
    private static double h;
    private static DecimalFormat dfx = new DecimalFormat("#.#");
    private static DecimalFormat dfy = new DecimalFormat("#.####");

    public static double function(double x, double y) {
        return x + Math.sin((y + 1) / Math.sqrt(13));
    }
```

```java
    public static void enterData() {
        boolean end = false;
        while (!end) {
            try {
                System.out.println("\n### Program obliczający równanie różniczkowe zwyczaje metodą
                Rungego-Kuttego rzędu IV ###");
                System.out.println("### Dla: y'(x) = x + sin((y(x)+1)/sqrt(13)) ###");
                System.out.println("\n");
                System.out.print(">> Wprowadź dolną granicę przedziału zbioru argumentów x
                ([x0,xn] -> x0):");
                x0 = new Scanner(System.in).nextDouble();
                System.out.print(">> Wprowadź dolną granicę przedziału zbioru argumentów x
                ([x0,xn] -> xn):");
                xn = new Scanner(System.in).nextDouble();
                System.out.print(">> Wprowadź warunek początkowy y(x0):");
                y0 = new Scanner(System.in).nextDouble();
                System.out.print(">> Wprowadź wartość skoku (h):");
                h = new Scanner(System.in).nextDouble();
                end = true;
            } catch (Exception e) {
                System.out.println("\n[ERROR] Wystapił błąd wpisywania danych!");
                for (int i = 0; i < 500; i++) {
                    System.out.println();
                }
            }
        }
    }

    public static void algorithmRK4() throws Exception {
        double n = (xn - x0) / h;
        xi = new double[(int) n + 1][1];
        yi = new double[(int) n + 1][1];
        xi[0][0] = x0;
        yi[0][0] = y0;
        System.out.println("    x    |    y    ");
        System.out.println("---------------");
        System.out.println(dfx.format(xi[0][0]) + " \t " + dfy.format(yi[0][0]));
        for (int i = 1; i < n + 1; i++) {
            double k1 = h * function(xi[i - 1][0], yi[i - 1][0]);
            double k2 = h * function(xi[i - 1][0] + 0.5 * h, yi[i - 1][0] + 0.5 * k1);
            double k3 = h * function(xi[i - 1][0] + 0.5 * h, yi[i - 1][0] + 0.5 * k2);
            double k4 = h * function(xi[i - 1][0] + h, yi[i - 1][0] + k3);
            xi[i][0] = xi[i - 1][0] + h;
            yi[i][0] = yi[i - 1][0] + ((1.0 / 6.0) * (k1 + (2 * k2) + (2 * k3) + k4));
            System.out.println(dfx.format(xi[i][0]) + " \t " + dfy.format(yi[i][0]));
        }
        System.out.println("\nWynik: " + dfy.format(yi[(int)n][0]));
    }

    public static void main(String[] args) {
        enterData();
        try {
            algorithmRK4();
        } catch (Exception e) {
            System.out.println("\n[ERROR] Wystąpił błąd obliczeniowy dla podanych danych! Spróbuj
            ponownie!");
            enterData();
        }
    }
}
```

Zrzut ekranu z działania programu:

```
"C:\Program Files\Java\jdk-9.0.1\bin\java" "-javaagent:D:\JetBrains Student\IntelliJ IDEA

### Program obliczający równanie różniczkowe zwyczaje metodą Rungego-Kuttego rzędu IV ###
### Dla: y'(x) = x + sin((y(x)+1)/sqrt(13)) ###


>> Wprowadź dolną granicę przedziału zbioru argumentów x ([x0,xn] -> x0):0,2
>> Wprowadź górną granicę przedziału zbioru argumentów x ([x0,xn] -> xn):1,2
>> Wprowadź warunek początkowy y(x0):1,1
>> Wprowadź wartość skoku (h):0,1
   x   |   y
---------------
0,2      1,1
0,3      1,1809
0,4      1,2738
0,5      1,3789
0,6      1,4965
0,7      1,6267
0,8      1,7697
0,9      1,9257
1    2,0948
1,1      2,277
1,2      2,4725

Wynik: 2,4725

Process finished with exit code 0
```

Opis programu:

Program w języku Java składa się z jednej klasy o nazwie MethodRungeKutty. Jest to klasa publiczna, która posiada 8 pól statycznych:

- double[ ][ ] xi – tablica dwuwymiarowa pełniąca rolę wektora obliczanych wartości x w kolejnych iteracjach
- double[ ][ ] yi – tablica dwuwymiarowa pełniąca rolę wektora obliczanych wartości y w kolejnych iteracjach
- double x0 – dolna granica przedziału argumentów x
- double xn – górna granica przedziału argumentów x
- double y0 – wartość początkowa $y(x_0)$
- double h – wartość skoku
- DecimalFormat dfx – określa format wypisywania danych z wektora xi
- DecimalFormat dfy – określa format wypisywania danych z wektora yi

Ponadto klasa posiada dwie metody statyczne nie wliczając metody inicjalizacji main. Są to:

- enterData() – metoda odpowiedzialna za wpisywanie danych przez użytkownika. Podaje on dolną oraz górną granicę przedziału argumentów x, warunek początkowy, czyli $y(x_0)$ oraz wartość skoku. Podczas wpisywania dane są walidowane i zapisywane do pól statycznych, kolejno: x0,xn,y0,h.
- algorithmRK4() – metoda odpowiedzialna za inicjalizację tablic dwuwymiarowych, obliczania ilości iteracji oraz wykonanie faktyczne iteracji i zapisywania wyników do wektorów xi oraz yi. Całość odbywa się zgodnie z algorytmem:

$$y_{i+1} = y_i + \Delta y_i$$

gdzie:

$$\Delta y_i = \frac{1}{6}\left(k_1^{(i)} + 2k_2^{(i)} + 2k_3^{(i)} + k_4^{(i)}\right)$$

gdzie:

$$k_1^{(i)} = h * f(x_i, y_i)$$
$$k_2^{(i)} = h * f\left(x_i + 0.5 * h, y_i + 0.5 * k_1^{(i)}\right)$$
$$k_3^{(i)} = h * f\left(x_i + 0.5 * h, y_i + 0.5 * k_2^{(i)}\right)$$
$$k_4^{(i)} = h * f\left(x_i + h, y_i + k_3^{(i)}\right)$$

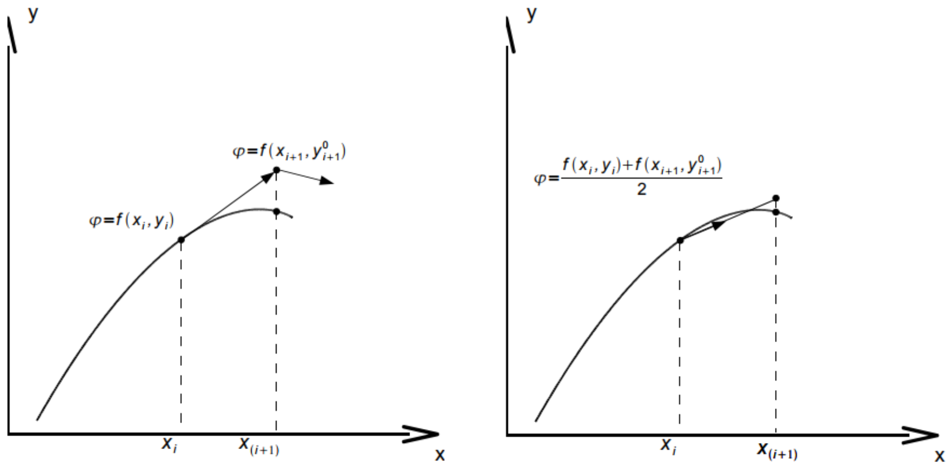
Wartość x_{i+1} wyznacza jest na podstawie wzoru:

$$x_i = x_0 + i * h$$

Na końcu wypisywane są poszczególne wartości x_i oraz $f(x_i)$ oraz ostateczny wynik.

3. Modyfikacja metody Eulera - metoda Heun'a.

W metodzie tej zamiast stałej wartości pochodnej obliczonej na początku przedziału, jak to było w metodzie Eulera, oblicza się pochodną również na końcu przedziału. Pierwsze oszacowanie wyniku nazywamy **predyktorem**, a następnie **korektorem.** Metoda ta, dzięki zabiegowi numerycznemu, daje sporą zmianę w dokładności wyniku i jest **znacznie dokładniejsza niż klasyczna metoda Eulera.**



Z lewej – predyktor (pierwszy strzał); z prawej – korektor

Predyktor wyrażamy stosując metodę Eulera:

$$\frac{dy}{dx} = f(x_i, y_i)$$
$$y^0_{i+1} = y_i + f(x_i, y_i) * h$$

Można oznaczyć również przez wzór:

$$y'_{i+1} = f(x_{i+1}, y^0_{i+1})$$

Przekształcając uzyskujemy:

$$y' = \frac{y_i + y_{i+1}}{2} = \frac{f(x_i, y_i) + f(x_{i+1}, y^0_{i+1})}{2}$$

Korektor określony jest wzorem:

$$y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y^0_{i+1})}{2} * h$$