Differential Equations

Computational Practicum

Pavel Vybornov BS17-07 (Variant 19)

Exact Solution

$$\begin{cases} y' = 2\sqrt{y} + 2y \\ y(0) = 1 \\ x \in [0; 9] \end{cases}$$

- $y' = 2\sqrt{y} + 2y$
- $\frac{dy}{dx} = 2\sqrt{y} + 2y$
- $\frac{dy}{2\sqrt{y}+2y} = dx$
- Integrating:
  $\int \frac{dy}{2\sqrt{y}+2y} = \int dx$
- $\int \frac{dy}{2\sqrt{y}(1+\sqrt{y})} = \int dx$
- $ln(1+\sqrt{y}) = x + C$
- $C = ln(1+\sqrt{y}) - x$
- $InitialValueProblem : C = ln(1+1) - 0 \implies C = ln(2)$
- $y = (e^{(C+x)} - 1)^2$
- $y = (e^{(ln(2)+x)} - 1)^2$

```
/**
 * Function to set Series for Exact Solution Graph
 * @param x0 - input initial conditions
 * @param X - input initial conditions
 * @param y0 - input initial conditions
 * @param N - input initial conditions
 * @return
 */
private XYChart.Series seriesGenerate(double x0, double X, double y0, double N){
    double h = (X-x0)/N;
    XYChart.Series series = new XYChart.Series();

    ObservableList<XYChart.Data> datas = FXCollections.observableArrayList();
    for(double x=x0; x<=X; x+=h){
        datas.add(new XYChart.Data(x, exactFunction(y0, x0, x)));
    }
    series.setName("Exact Solution");
    series.setData(datas);
    return series;
}
```

Euler Method

$$\text{Step:} \quad h = \frac{X - x_0}{N}$$

$$x_i = x_0 + ih$$

$$y_{i+1} = y_i + h * (-y_i - x_i)$$

```
/**
 * Function to get 'eulerY'
 * @param xPrev
 * @param yPrev
 * @return
 */
protected double nextY(double xPrev, double yPrev) {
    return (eulerY(xPrev, yPrev));
}


/**
 * Finding y according to Euler's Method
 * @param xPrev - previous value of x
 * @param yPrev - previous value of y
 * @return
 */
protected double eulerY(double xPrev, double yPrev) {
    return yPrev + h * diffur(yPrev);
}
```

Improved Euler Method

$$y_{i+1} = y_i + h\frac{f(x_i, y_i) + f(x_i + h, y_i + hf(x_i, y_i))}{2}$$

```
/**
 * Find next value of y according to Improved Euler Method
 * uses super (Euler)
 * @param xPrev
 * @param yPrev
 * @return
 */
@Override
protected double nextY(double xPrev, double yPrev) {
    return yPrev+super.h*((super.diffur(yPrev)+super.diffur(super.eulerY(xPrev,yPrev)))/2);
}
```

Runge-Kutta Method

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k1}{2})$$

$$k_3 = hf(x_i + \frac{h}{2}, y_i + \frac{k2}{2})$$

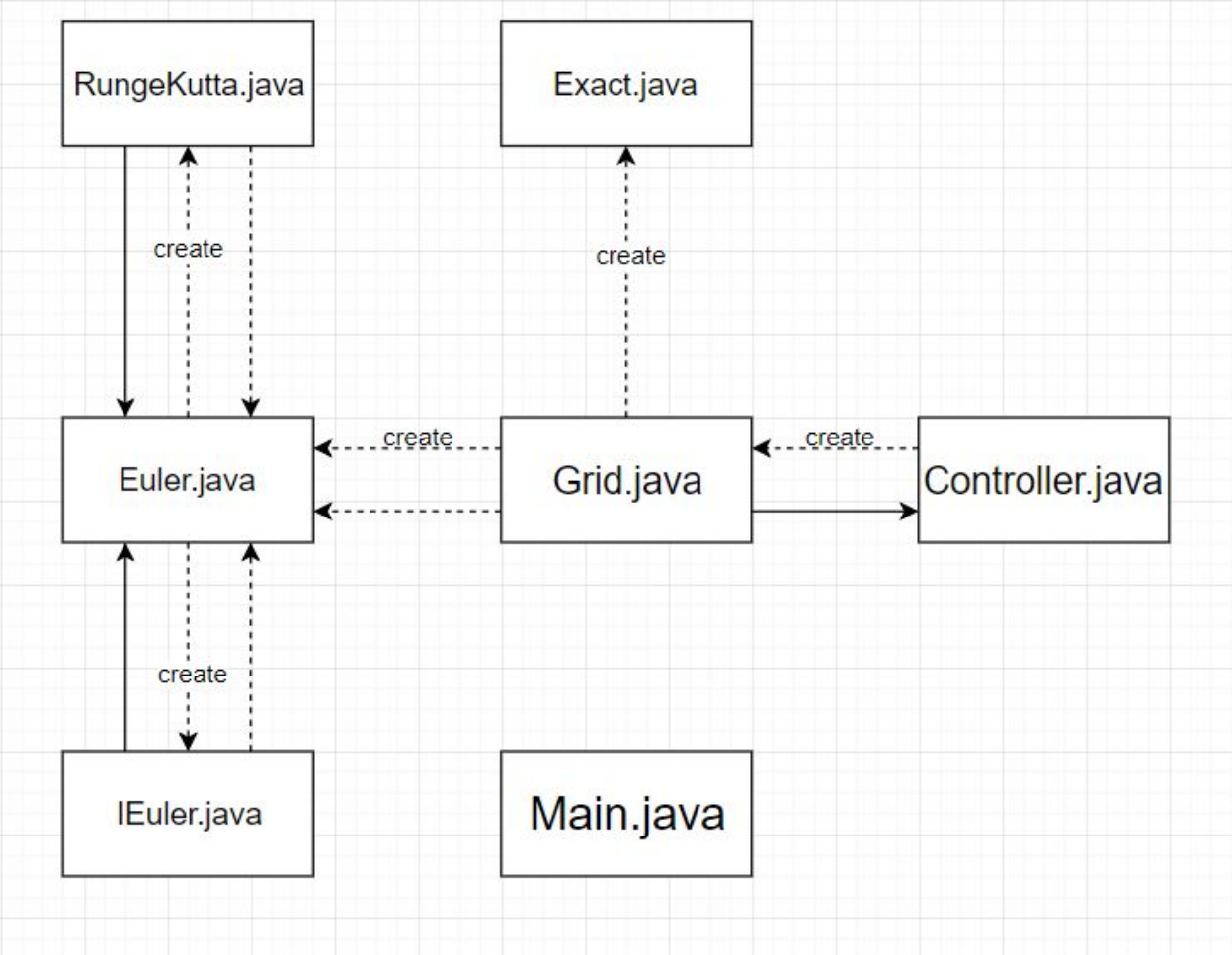
$$k_4 = hf(x_i + h, y_i + k_3)$$

$$y_{i+1} = y_i + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)$$

```java
/**
 * Function to get next y for Runge-Kutta Method
 * uses super (Euler)
 * @param xPrev - previous value of x
 * @param yPrev - previous value of y
 * @return
 */
@Override
protected double nextY(double xPrev, double yPrev) {
    double k1 = super.h*super.diffur(yPrev);
    double k2 = super.h*super.diffur( y: yPrev + k1/2);
    double k3 = super.h*super.diffur( y: yPrev + k2/2);
    double k4 = super.h*super.diffur( y: yPrev + k3);
    return (yPrev+ (k1+2*k2+2*k3+k4)*(1.0/6.0));
}
```

Code

- [Code on GitHub]
- Structure



Graphical User Interface

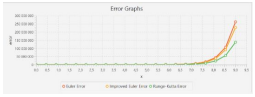
- Application has two tabs: “Graphs” and “Error Analysis”

“Graphs”

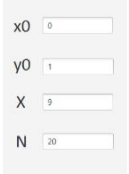
Contains:

- Canvas with Graphs of Exact Solution, Euler Method, Improved Euler Method and Runge-Kutta Method

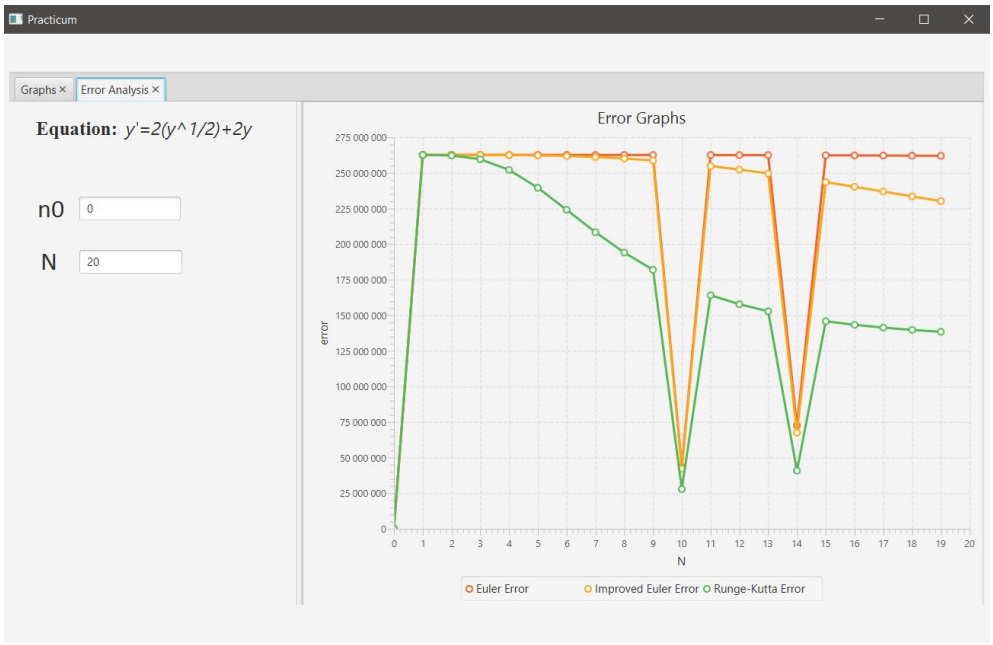
- Canvas with Graphs of errors of Euler Method, Improved Euler Method and Runge-Kutta Method



- Text-fields with Initial Values which can be changed

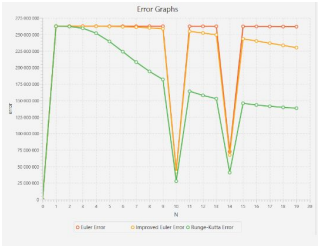


“Error Analysis”



Contains:

- Canvas with Graphs of Error Analysis of Euler Method, Improved Euler Method and Runge-Kutta Method



- Text-fields with values that can be changed