**Objective:** *Implementation of Euler, Improved Euler and Runge Kutta method for the given variant and comparison of error between them plotted in graph with capability of providing desired number of grids.*

**-------------------------------------------------**

$\boldsymbol{Variant\ 3}$: $y' = \frac{4}{x^2} - \frac{y}{x} - y^2$

-------------------------------------------------

$$or, \quad y' + y^2 + \frac{y}{x} = \frac{4}{x^2}$$

*We will seek for a particular solution in the form:*

$$y = \frac{c}{x} \qquad y' = -\frac{c}{x^2}$$

*Substituting this into the equation, we obtain:*

$$or, \quad -\frac{c}{x^2} + \frac{c}{x^2} + \frac{c^2}{x^2} = \frac{4}{x^2}$$

$$or, \quad c^2 = 4$$

$$\therefore \quad c = \pm 2$$

*We can take any value of* c*. For example, let* $c = 2$*. now, when the particular solution is known, we make the replacement:*

$$or, \quad z = \frac{1}{y - \frac{2}{x}} \qquad\qquad or, \quad y = \frac{1}{z} + \frac{2}{x}$$

$$and, \quad y' = -\frac{z'}{z^2} - \frac{2}{x^2}$$

*Now substitute this into the original Riccati equation:*

$$or, \quad -\frac{z'}{z^2} - \frac{2}{x^2} = \frac{4}{x^2} - \frac{\frac{1}{z} + \frac{2}{x}}{x} - \left(\frac{1}{z} + \frac{2}{x}\right)\left(\frac{1}{z} + \frac{2}{x}\right)$$

$$or, \quad -\frac{z'}{z^2} - \frac{2}{x^2} = \frac{4}{x^2} - \frac{1}{2x} - \frac{2}{x^2} - \frac{1}{z^2} - \frac{4}{zx} - \frac{4}{x^2}$$

*So cancelling out the terms,*

$$or, \quad -\frac{z'}{z^2} = \frac{5}{zx} - \frac{1}{z^2}$$

$$or, \quad z' = \frac{5z}{x} + 1$$

*Integrating for x,*

$$\therefore \quad z = c_1 x^5 - \frac{x}{4}$$

$$y = \frac{1}{c_1 x^5 - \frac{x}{4}} + \frac{2}{x} \Big\} \rightarrow (1,3)$$

$$Simpliying\ with\ x = 1\ and\ y = 3\ we\ will\ get\ c_1 = \frac{5}{4}$$

$$Putting\ back\ the\ value\ of\ c_1\ we\ will\ get,$$

$$\boldsymbol{y = \frac{4 + 2(5x^4 - 1)}{x(5x^4 - 1)}\ (analytical\ solution)}$$

**<u>Exact Solution</u>**

```
public class Exact {
    public static double function(double x) {
        double y;
        y = (4/(5*x*x*x*x*x - x)) + 2/x;
        return y;
    }

    public static XYChart.Series exact(double x0, double y0, double x, double N) {
        double currX = x0;
        double currY = y0;
        XYChart.Series ser = new XYChart.Series();
        double n = Math.abs(x - x0) / N;

        //calculation of exact solution and graph
        while (currX <= x) {
            ser.getData().add(new XYChart.Data<>(currX, currY));

            // exact solution
            currY = function(currX);
            currX += n;
        }
        return ser;
    }
}
```

# Euler Method

```java
public class Euler {
    public static XYChart.Series euler(double x0, double y0, double x, double N) {
        Double currX = x0;
        Double currY = y0;
        XYChart.Series ser = new XYChart.Series();
        Double n = Math.abs(x - x0) / N;
        ser.getData().add(new XYChart.Data<>(currX, currY));

        //euler method and graph
        while (currX <= x) {

            currY += n * Variant.funct(currX, currY);
            currX += n;

            ser.getData().add(new XYChart.Data<>(currX, currY));
        }
        return ser;
    }
}
```

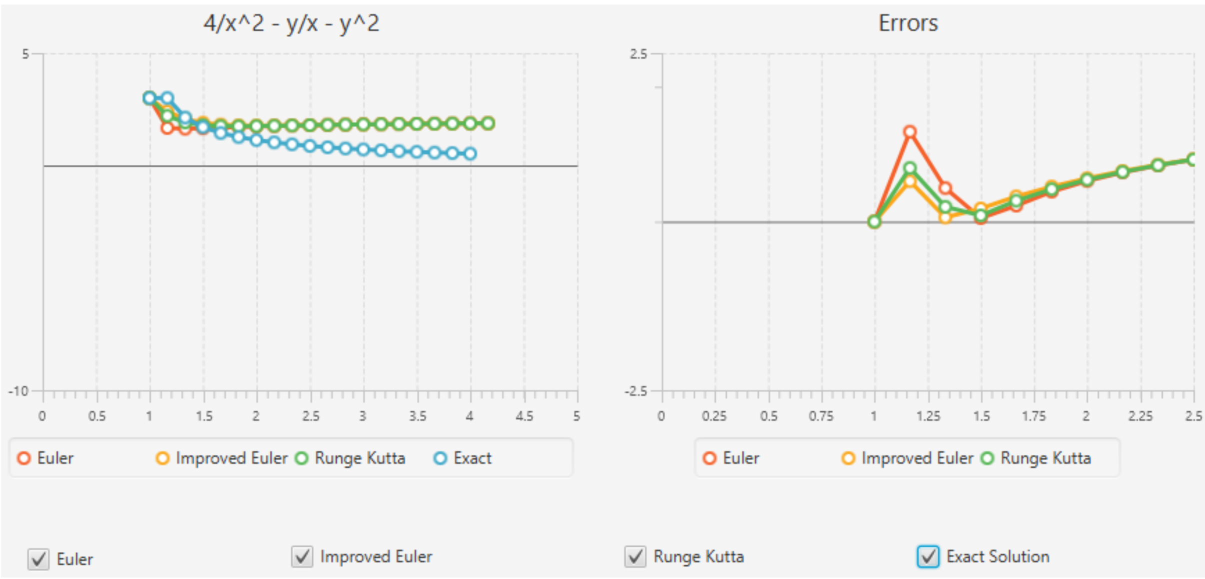
# Improved Euler

```java
class IEuler {
    public static XYChart.Series ieuler(double x0, double y0, double x, double N) {
        double currX = x0;
        double currY = y0;
        XYChart.Series ser = new XYChart.Series();
        double n = Math.abs(x - x0) / N;
        ser.getData().add(new XYChart.Data<>(currX, currY));

        double val;
        double tempo;

        //graph and improved euler method
        while (currX <= x) {

            tempo = Variant.funct(currX,currY);
            val = n*Variant.funct( x: currX+n/2,  y: currY+ n*tempo/2);
            currX += n;
            currY += val;

            ser.getData().add(new XYChart.Data<>(currX, currY));
        }

        return ser;
    }
}
```

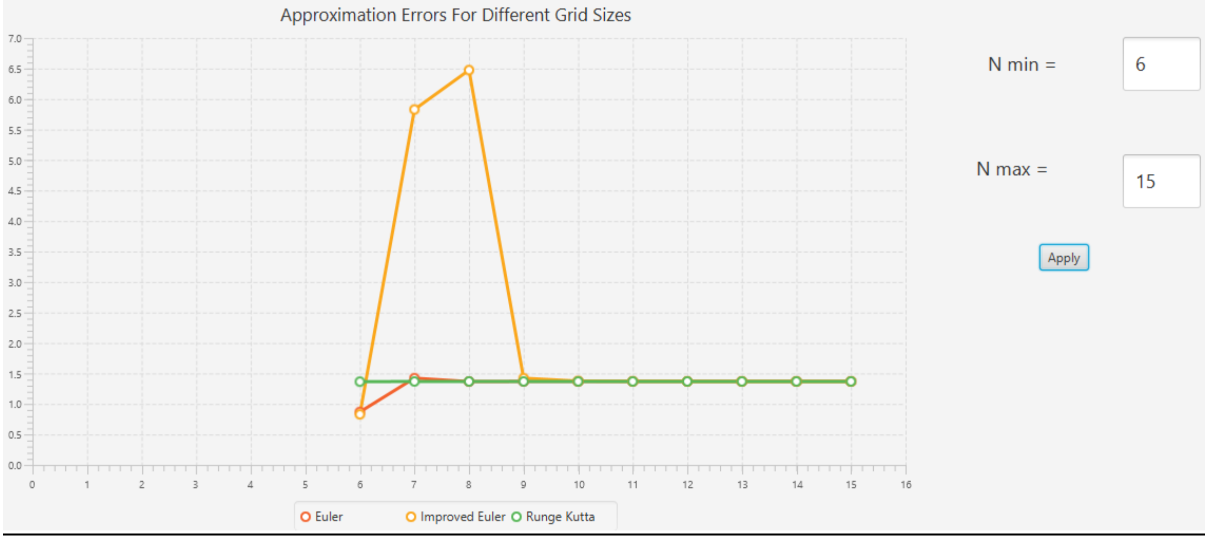
## Runge Kutta

```java
public class RKutta {
    public static XYChart.Series rKutta(double x0, double y0, double x, double N) {
        double currX = x0;
        double currY = y0;
        double k1;
        double k2;
        double k3;
        double k4;
        XYChart.Series ser = new XYChart.Series();
        double n = Math.abs(x - x0) / N;

        while (currX <= x + n) {

            ser.getData().add(new XYChart.Data<>(currX, currY));

            k1 = Variant.funct(currX, currY);
            k2 = Variant.funct(currX + n * 0.5, currY + n * k1 * 0.5);
            k3 = Variant.funct(currX + n * 0.5, currY + n * k2 * 0.5);
            k4 = Variant.funct(currX + n, currY + n * k3);

            currY += n / 6 * (k1 + 2 * k2 + 2 * k3 + k4);
            currX += n;
        }
        return ser;
    }
}
```

## Plot and Error



<u>**Approximation Errors for Different Grid Sizes**</u>




**Conclusion:** *Thus, three different methods are implemented and they are compared and their errors are plotted. This report is based on the requirements given on moodle.*