Pimenov Denis. BS17-05

GitHub: https://github.com/Dapimex/DE_Assignment

Exact solution:

$$y' = \frac{y}{x} + \frac{x}{y}; \qquad y' - \frac{y}{x} = \frac{x}{y}; \qquad n = -1;\ P(x) = -\frac{1}{x};\ \ Q(x) = x$$

$$I(x) = e^{\int (1-n)\cdot P(x)\cdot dx} = e^{-2\cdot\int \frac{dx}{x}} = \frac{1}{x^2}$$

$$y^{1-n} = \frac{1}{I(x)}\cdot\left[\int (1-n)\cdot Q(x)\cdot I(x)\cdot dx + C\right]$$

$$y^2 = x^2\cdot\left(2\cdot\int x\cdot\frac{dx}{x^2} + C\right) = x^2\cdot\left(2\cdot\int \frac{dx}{x} + C\right) = x^2\cdot(2\cdot\ln(x) + C1)$$

$$y = \pm x\cdot\sqrt{2\cdot\ln(x) + C1}$$

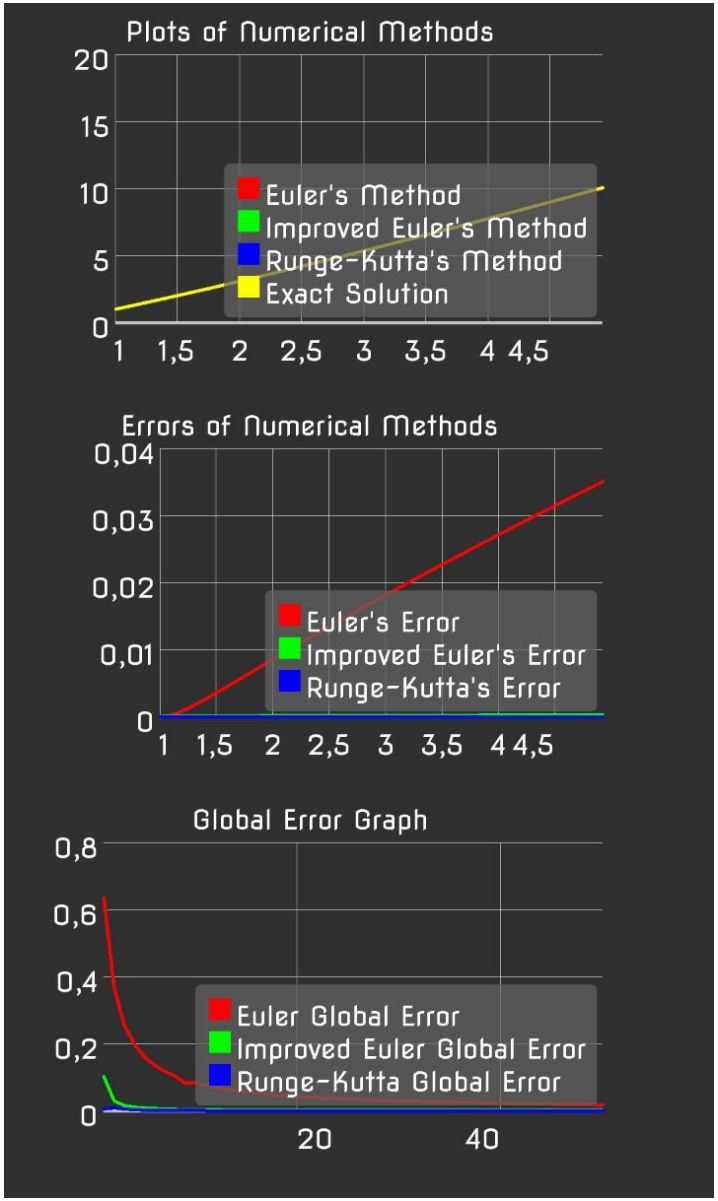
Steps: 50

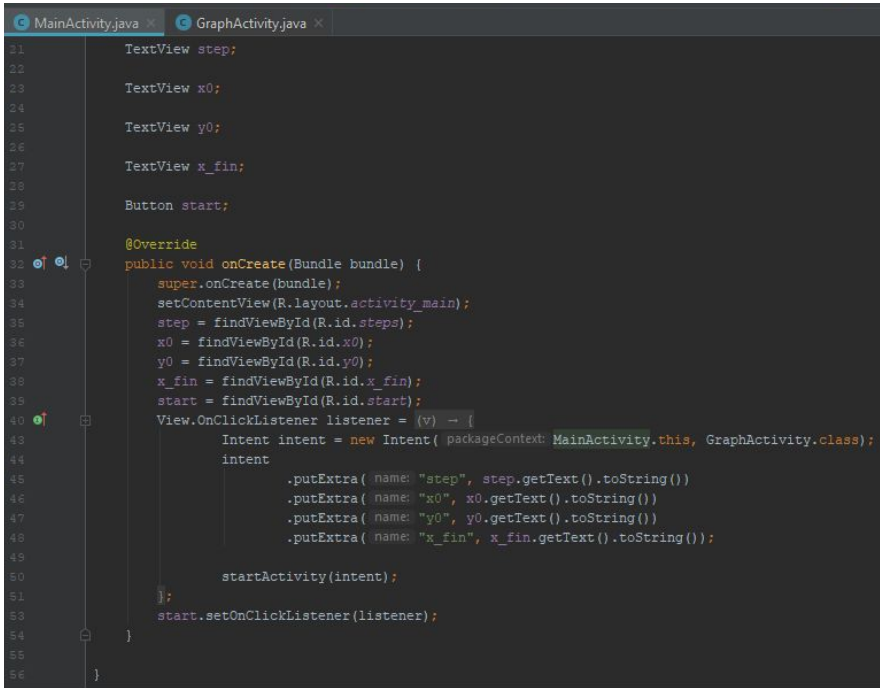
x0: 1

y0: 1

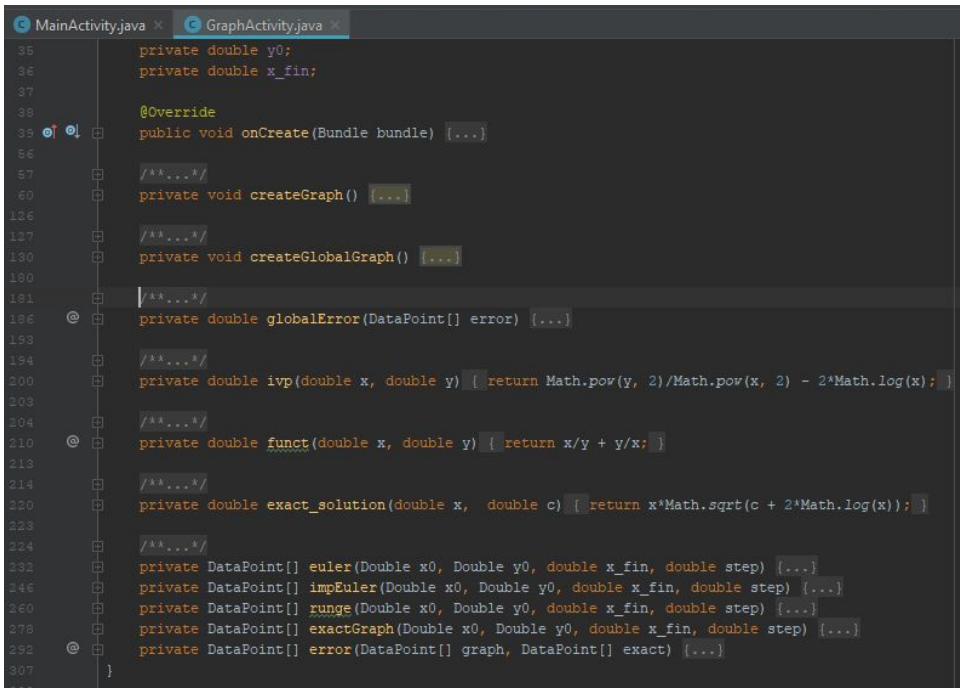
X: 5

SHOW GRAPH





```
MainActivity.java | GraphActivity.java

    TextView step;

    TextView x0;

    TextView y0;

    TextView x_fin;

    Button start;

    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_main);
        step = findViewById(R.id.steps);
        x0 = findViewById(R.id.x0);
        y0 = findViewById(R.id.y0);
        x_fin = findViewById(R.id.x_fin);
        start = findViewById(R.id.start);
        View.OnClickListener listener = (v) → {
                Intent intent = new Intent( packageContext: MainActivity.this, GraphActivity.class);
                intent
                        .putExtra( name: "step", step.getText().toString())
                        .putExtra( name: "x0", x0.getText().toString())
                        .putExtra( name: "y0", y0.getText().toString())
                        .putExtra( name: "x_fin", x_fin.getText().toString());

                startActivity(intent);
        };
        start.setOnClickListener(listener);
    }

}
```



```
MainActivity.java | GraphActivity.java

    private double y0;
    private double x_fin;

    @Override
    public void onCreate(Bundle bundle) {...}

    /**...*/
    private void createGraph() {...}

    /**...*/
    private void createGlobalGraph() {...}

    /**...*/
    private double globalError(DataPoint[] error) {...}

    /**...*/
    private double ivp(double x, double y) { return Math.pow(y, 2)/Math.pow(x, 2) - 2*Math.log(x); }

    /**...*/
    private double funct(double x, double y) { return x/y + y/x; }

    /**...*/
    private double exact_solution(double x,  double c) { return x*Math.sqrt(c + 2*Math.log(x)); }

    /**...*/
    private DataPoint[] euler(Double x0, Double y0, double x_fin, double step) {...}
    private DataPoint[] impEuler(Double x0, Double y0, double x_fin, double step) {...}
    private DataPoint[] runge(Double x0, Double y0, double x_fin, double step) {...}
    private DataPoint[] exactGraph(Double x0, Double y0, double x_fin, double step) {...}
    private DataPoint[] error(DataPoint[] graph, DataPoint[] exact) {...}
}
```

```java
/**...*/
private DataPoint[] euler(Double x0, Double y0, double x_fin, double step) {
    dataPoints.clear();
    while (x0 <= x_fin) {
        dataPoints.add(new DataPoint(x0, y0));
        double d = step*funct(x0, y0);
        x0 += step;
        y0 += d;
        if (x0.isNaN() || y0.isNaN()) break;
    }
    DataPoint[] res = new DataPoint[dataPoints.size()];
    for (int i = 0; i < dataPoints.size(); i++) res[i] = dataPoints.get(i);

    return res;
}
```

```java
private DataPoint[] impEuler(Double x0, Double y0, double x_fin, double step) {
    dataPoints.clear();
    while (x0 <= x_fin) {
        dataPoints.add(new DataPoint(x0, y0));
        double d = step*funct( x: x0 + step/2,  y: y0 + step/2*funct(x0, y0));
        x0 += step;
        y0 += d;
        if (x0.isNaN() || y0.isNaN()) break;
    }
    DataPoint[] res = new DataPoint[dataPoints.size()];
    for (int i = 0; i < dataPoints.size(); i++) res[i] = dataPoints.get(i);

    return res;
}
```

```java
private DataPoint[] runge(Double x0, Double y0, double x_fin, double step) {
    dataPoints.clear();
    while (x0 <= x_fin) {
        dataPoints.add(new DataPoint(x0, y0));
        double k1 = funct(x0, y0);
        double k2 = funct( x: x0 + step/2,  y: y0 + step/2*k1);
        double k3 = funct( x: x0 + step/2,  y: y0 + step/2*k2);
        double k4 = funct( x: x0 + step,  y: y0 + step*k3);
        double d = step/6*(k1+2*k2+2*k3+k4);
        x0 += step;
        y0 += d;
        if (x0.isNaN() || y0.isNaN()) break;
    }
    DataPoint[] res = new DataPoint[dataPoints.size()];
    for (int i = 0; i < dataPoints.size(); i++) res[i] = dataPoints.get(i);

    return res;
}
```

```java
private DataPoint[] exactGraph(Double x0, Double y0, double x_fin, double step) {
    dataPoints.clear();
    double c = ivp(x0, y0);
    while (x0 <= x_fin) {
        dataPoints.add(new DataPoint(x0, y0));
        x0 += step;
        y0 = exact_solution(x0, c);
    }
    DataPoint[] res = new DataPoint[dataPoints.size()];
    for (int i = 0; i < dataPoints.size(); i++) {
        res[i] = dataPoints.get(i);
    }
    return res;
}
```

```
private DataPoint[] error(DataPoint[] graph, DataPoint[] exact) {
    if (!(graph.length == exact.length)) {
        Toast.makeText(getApplicationContext(), getResources().getString(R.string.not_match_length), Toast.LENGTH_SHORT).show();
        return null;
    }
    DataPoint[] res = new DataPoint[graph.length];
    for (int i = 0; i < graph.length; i++) {
        if (graph[i].getX() != exact[i].getX()) {
            Toast.makeText(getApplicationContext(), getResources().getString(R.string.not_match_x), Toast.LENGTH_SHORT).show();
            return null;
        }
        res[i] = new DataPoint(graph[i].getX(), Math.abs(graph[i].getY() - exact[i].getY()));
    }
    return res;
}
```

```
private void createGlobalGraph() {
    DataPoint[] globalEuler = new DataPoint[steps];
    DataPoint[] globalImpEuler = new DataPoint[steps];
    DataPoint[] globalRunge = new DataPoint[steps];

    for (int i = 0; i < steps; i++) {
        double init_step = (x_fin - x0)/(i+1);

        DataPoint[] euler = euler(x0, y0, x_fin, init_step);
        DataPoint[] impEuler = impEuler(x0, y0, x_fin, init_step);
        DataPoint[] runge = runge(x0, y0, x_fin, init_step);
        DataPoint[] exact = exactGraph(x0, y0, x_fin, init_step);

        DataPoint[] errorEuler = error(euler, exact);
        DataPoint[] errorImpEuler = error(impEuler, exact);
        DataPoint[] errorRunge = error(runge, exact);

        globalEuler[i] = new DataPoint((i + 1), globalError(errorEuler));
        globalImpEuler[i] = new DataPoint((i + 1), globalError(errorImpEuler));
        globalRunge[i] = new DataPoint((i + 1), globalError(errorRunge));
    }

    LineGraphSeries<DataPoint> glEuler = new LineGraphSeries<>(globalEuler);
    LineGraphSeries<DataPoint> glImpEuler = new LineGraphSeries<>(globalImpEuler);
    LineGraphSeries<DataPoint> glRunge = new LineGraphSeries<>(globalRunge);
```