Innopolis University, 2018
F18 Differential Equations
Programming Assignment Report
Gleb Petrakov
B17-03
Variant 9

Problem

Given first-order nonlinear ordinary differential equation – Bernoulli equation
`f(x, y) = x / y + y / x`
`y’ = f(x, y)`

Analytical solution
Point that `(x != 0)` and `(y != 0)`
Substract `(y / x)` from both sides and multiply both sides by `(2 * y)`:
`2 * (dy / dx) * y - 2 * (y^2 / x) = 2 * x`

Let `(u(x) = y^2)`:
(du / dx) - 2 * (du / x) = 2 * x

Let `(mu = e ^ (integral(-2 / x)dx) = 1 / (x^2)`:
`(du / dx) / (x^2) + (d(1 / (x^2)) / dx) * u = 2 / x`

Apply reverse product rule:
`d(u / (x^2)) / dx = 2 / x`

Integrate both sides with `dx`:
`integral(d(u / (x^2)) / dx)dx = integral(2 / x)dx`
`u / (x^2) = 2 * ln(x) + c`
(`c` – some constant)

`u = (x^2) * (2 * ln(x) + c)`

`y = (+/-)x * sqrt(2 * ln(x) + c)`

Evaluate `(c)` in terms of initial value problem `(x0, y0)`:
`c = (y0^2) / (x0^2) - 2 * ln(x0)`

Computational problem for this solution is trivial:
`y = (+/-)x * sqrt(2 * ln(x) + ((y0^2) / (x0^2) - 2 * ln(x0)))`

Approximation methods

Let (h) be some delta for (x) steps, (xi) and (yi) – x and y values for previous step

Euler method

```
y = yi + h * f(xi, yi)
```

Improved Euler method

```
k1 = f(xi, yi)
k2 = f(xi + h, yi + h * k1)
y = yi + h * (k1 + k2) / 2
```

Runge-Kutta method

```
xz = xi + h / 2
k1 = f(xi, yi)
k2 = f(xz, yi + h * k1 / 2)
k3 = f(xz, yi + h * k2 / 2)
k4 = f(xi + h, yi + h * k3)
y = yi + h * (k1 + 2 * k2 + 2 * k3 + k4) / 6
```

Technical implementation

https://github.com/imajou/F18-DE-Assignment

Platform: Android Native
Pattern: MVC
Language: Kotlin
IDE: Android Studio 3.2.1
Minimal SDK: 21 (Android 5.0)
Third-party libs: com.jjoe64:graphview (plotting lib)

Built apk provided in GitHub repo, if one wants to build it from sources, it is recommended to use Android Studio for building purposes&

The application provides single activity with plotting view with ability to control showed graphic and control panel to adjust problem parameters.

activities/MainActivity.kt

Activity logic, listeners binding, plotting information receiver.

model/EquationVariant.kt

Contains parameters x0 and y0 for initial value problem
Interface to specify needed functions to solve a problem:

<u>getExactDifferentialSolution(x: Double): Double</u>
Returns exact analytical solution for the initial value problem

<u>getFunctionValue(x: Double, y: Double): Double</u>
Returns f(x, y) function result.

<u>getEulerSolutionY(xi: Double, yi: Double, h: Double): Double</u>
Returns next point y-value corresponding to Euler method approximation

<u>getEulerImprovedSolutionY(xi: Double, yi: Double, h: Double): Double</u>
Returns next point y-value corresponding to improved Euler method approximation

<u>getRungeKuttaSolutionY(xi: Double, yi: Double, h: Double): Double</u>
Returns next point y-value corresponding to Runge-Kutta method approximation

model/EquationVariant9.kt

EquationVariant interface implementation for variant 9.

model/Equation.kt

Main computational power of the application stores and computes solutions and approximations/local errors of particular equation variant

model/Solution.kt

Object to compute points list for different approximation approaches.

model/ErrorLocal.kt

Object to compute points list for different approximation approaches local errors.

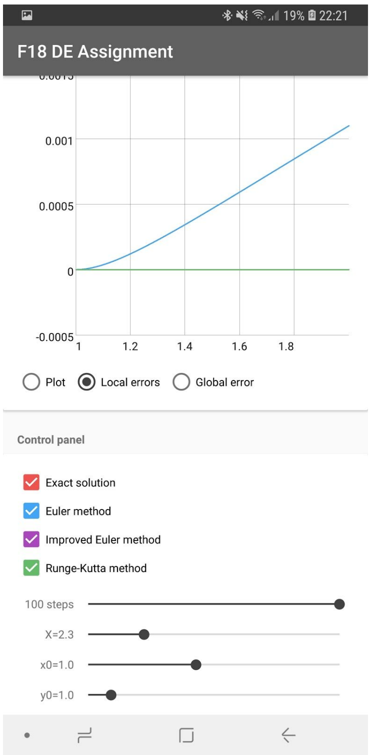
model/ErrorGlobal.kt

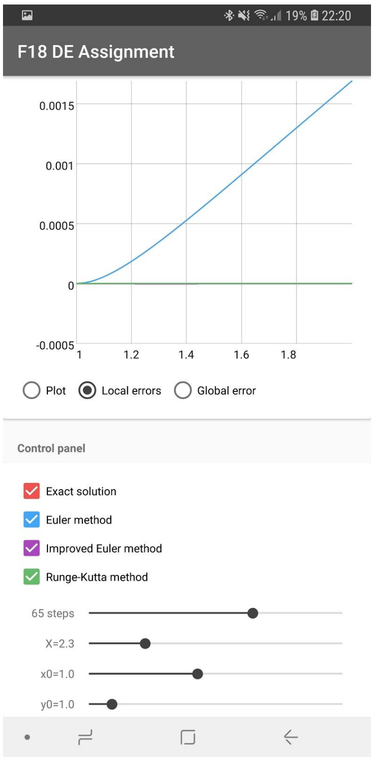
Object to compute points list for different approximation approaches global errors.

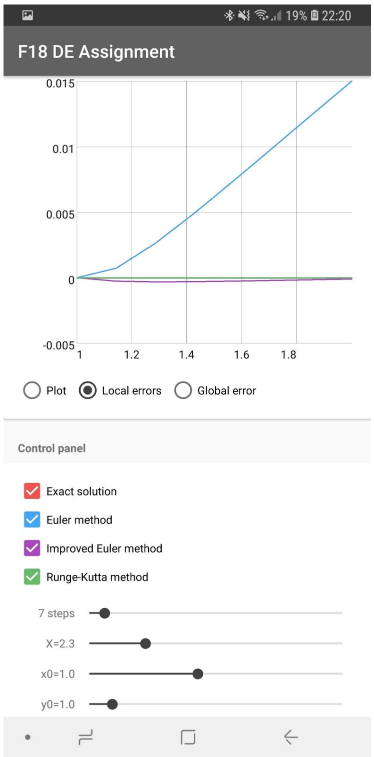
res/layout/activity_main.xml

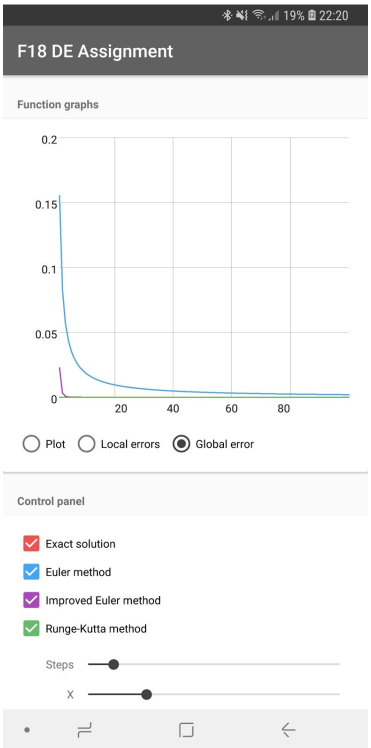
Main activity layout

Graphics


F18 DE Assignment
0.001
0.0005
0
-0.0005
1
1.2
1.4
1.6
1.8
Plot
Local errors
Global error
Control panel
Exact solution
Euler method
Improved Euler method
Runge-Kutta method
100 steps
X=2.3
x0=1.0
y0=1.0


F18 DE Assignment
0.0015
0.001
0.0005
0
-0.0005
1
1.2
1.4
1.6
1.8
Plot
Local errors
Global error
Control panel
Exact solution
Euler method
Improved Euler method
Runge-Kutta method
65 steps
X=2.3
x0=1.0
y0=1.0


F18 DE Assignment
0.015
0.01
0.005
0
-0.005
1
1.2
1.4
1.6
1.8
Plot
Local errors
Global error
Control panel
Exact solution
Euler method
Improved Euler method
Runge-Kutta method
7 steps
X=2.3
x0=1.0
y0=1.0


F18 DE Assignment
Function graphs
0.2
0.15
0.1
0.05
0
20
40
60
80
Plot
Local errors
Global error
Control panel
Exact solution
Euler method
Improved Euler method
Runge-Kutta method
Steps
X

Graphics analysis

Let us analyze graphics for basic given initial value problem – (1.0, 1.0)

The function is steadily increasing on the given interval – no special points

As one can possibly see, that local error for Euler method is maximal one, improved Euler method going next and Runge-Kutta is the best of all. Overall, local error tends to increase while we decreasing steps count.

Euler method uses trapeze to approximate and have average error of O(n^2)
Improved Euler method uses mean values of trapeze – error O(n^3)
Runge-Kutta method error – O(n^4).

Increasing number of steps on steady increasing function improves accuracy.