Computational Practicum Report

Differential Equations

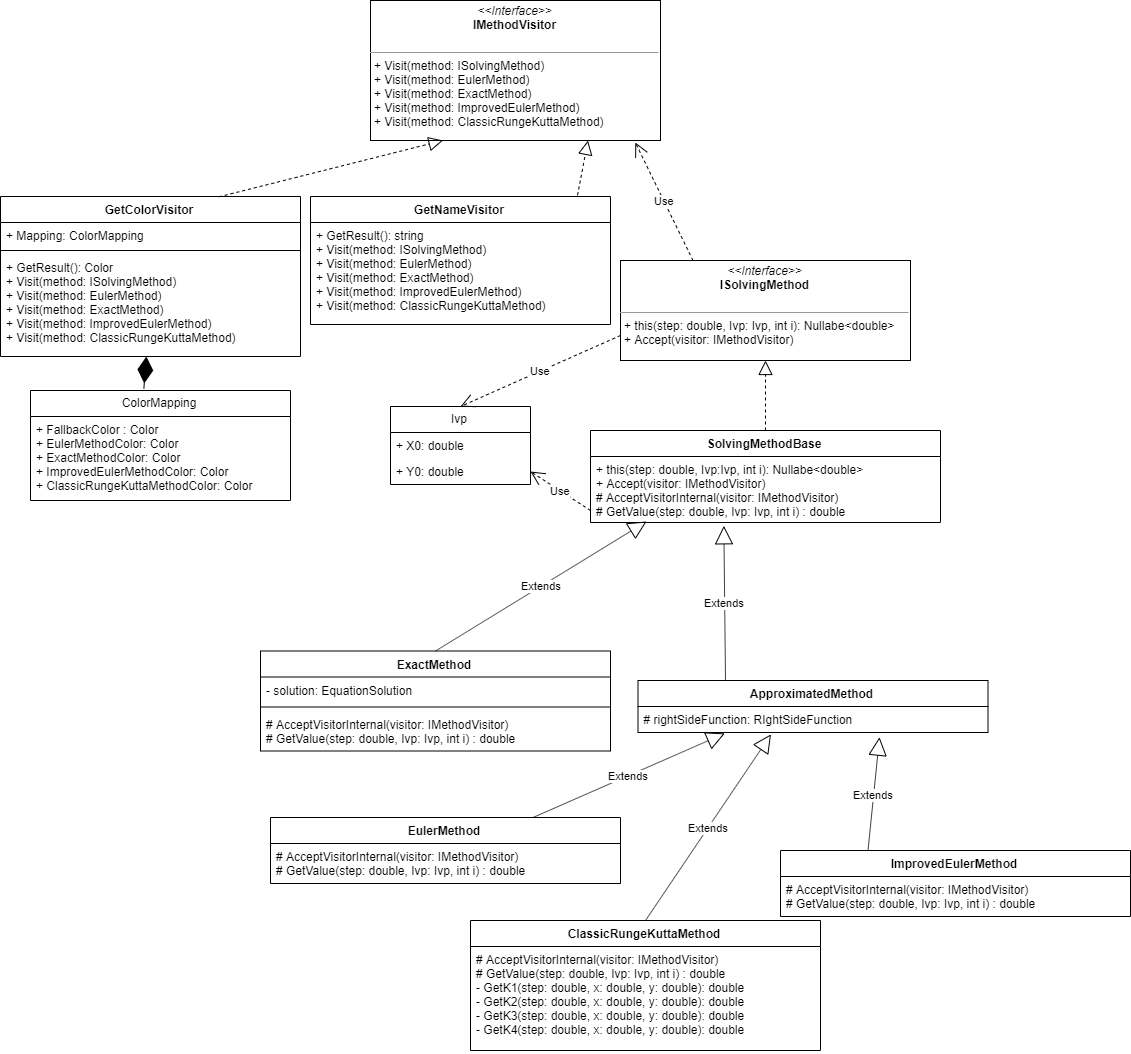
Vladislav Kantaev. Bs18-06. Variant 11

2019

Part 1

Part 2

Here is the link to GitHub repository: <https://github.com/Delt06/differential-equations>

Software is implemented via C# 8 and Windows Forms (.NET Core).  
Classes above encapsulate methods to solve differential equations. The number of methods is limited (4), thus we can apply design Visitor in order to add extra functionality to those classes. For example, GetColorVisitor is used to have graphs of every different methods have different colors.

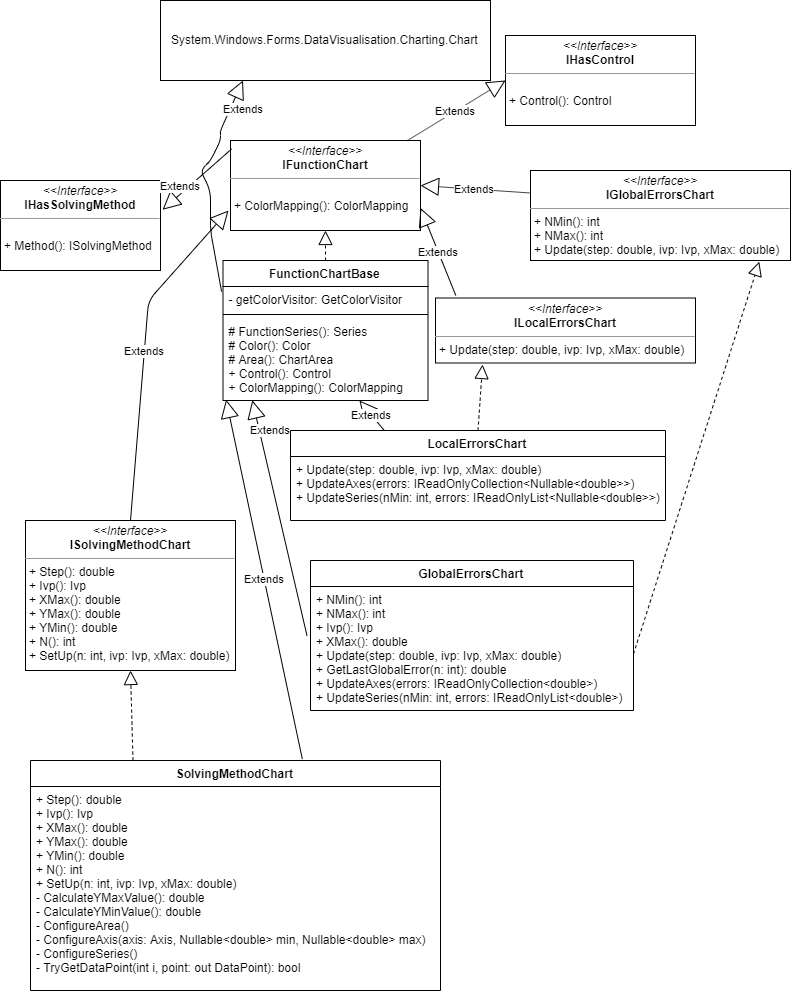
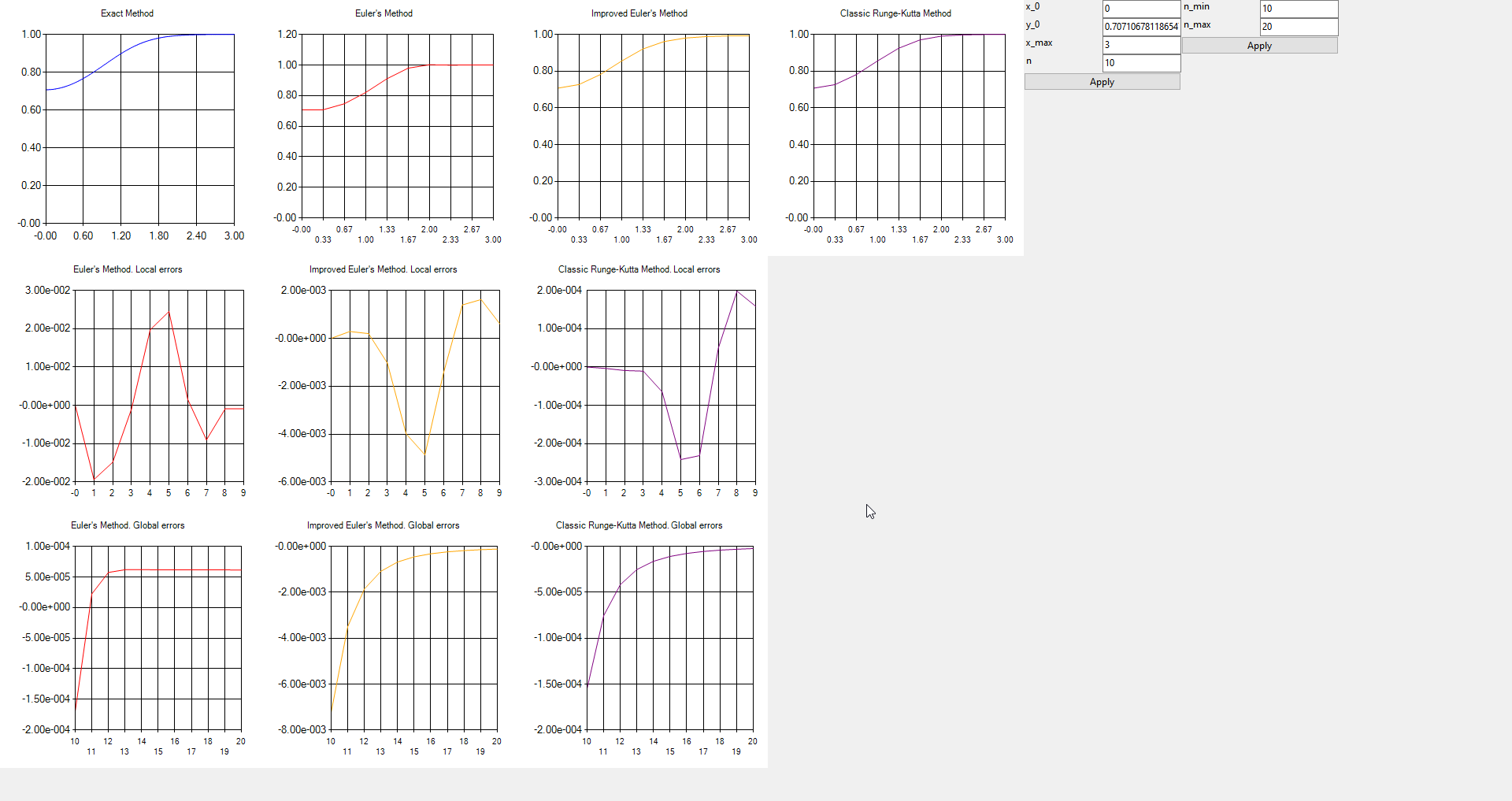


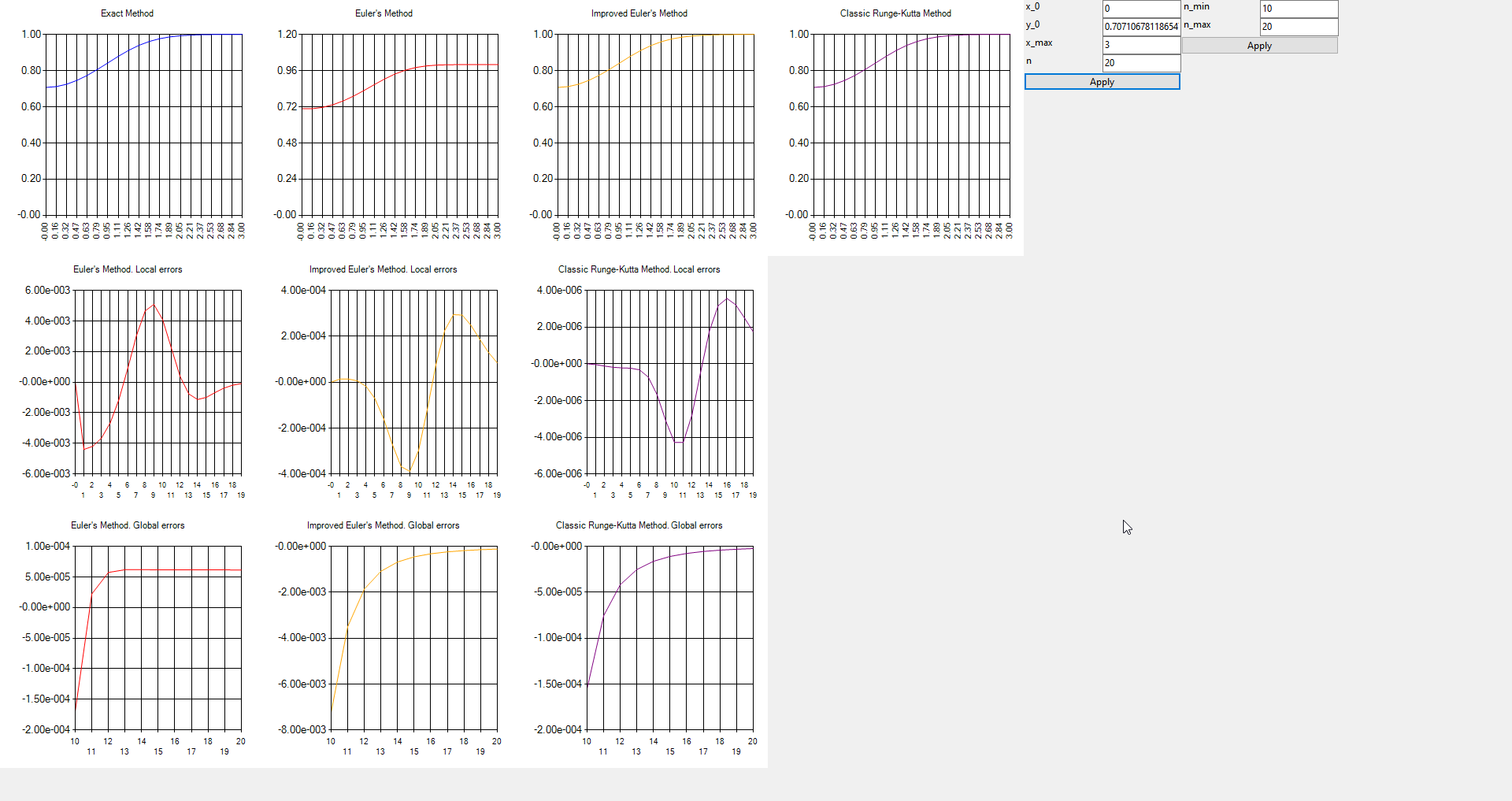
Diagram above shows the hierarchy of charts. There are 3 different chart types: for equation solutions, global, and local errors.

This is a screenshot of the implemented program. In the top row there are 4 charts of equation solutions. In the middle row there are charts of local errors, in the bottom we have global errors. The middle and the bottom rows do not include exact method because there is no need to investigate its convergence.

Exact method always has high enough precision in order to represent exact solution as closely as possible.

On the right side of the window there are fields that allow to modify the following parameters:

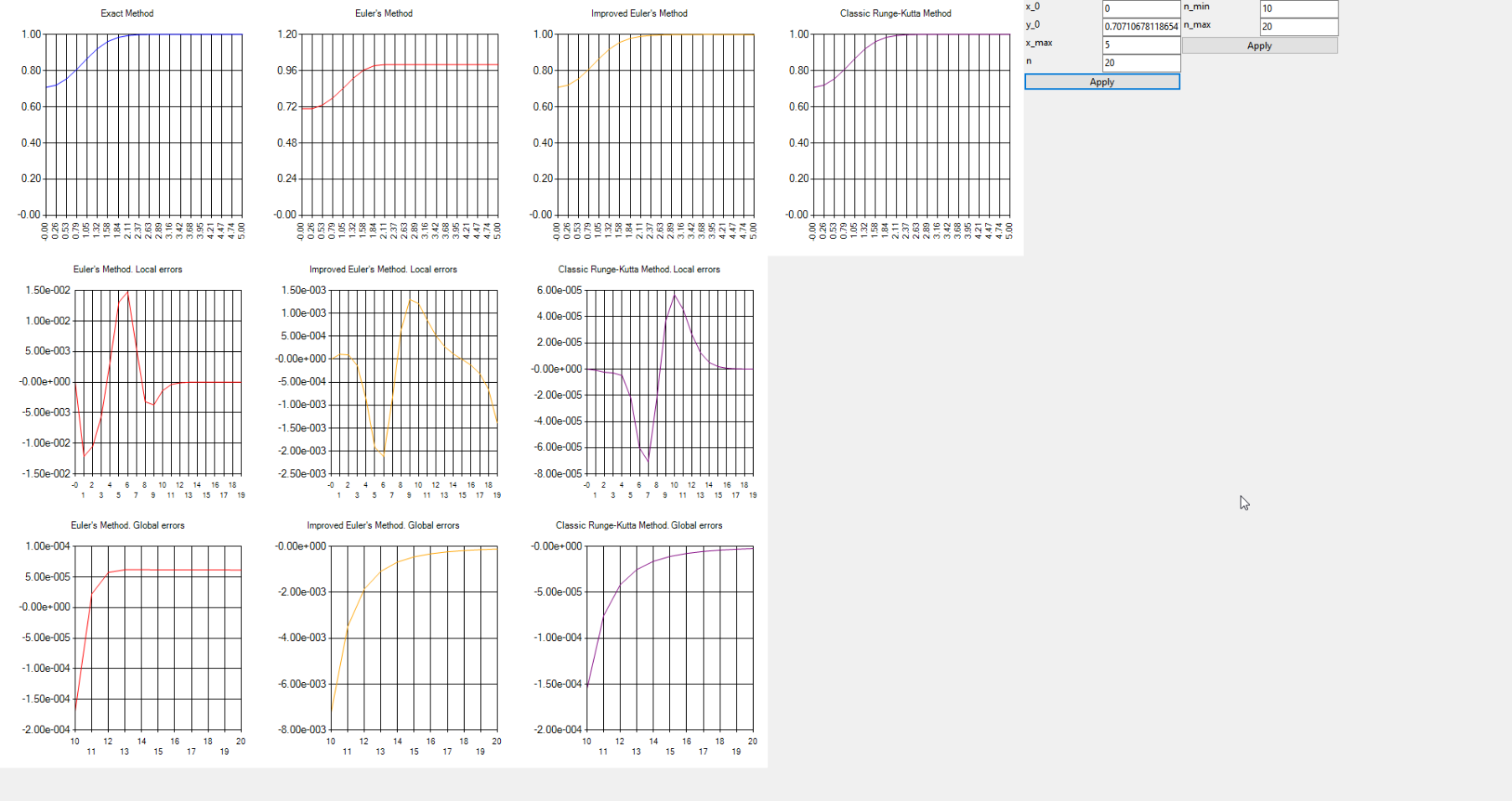
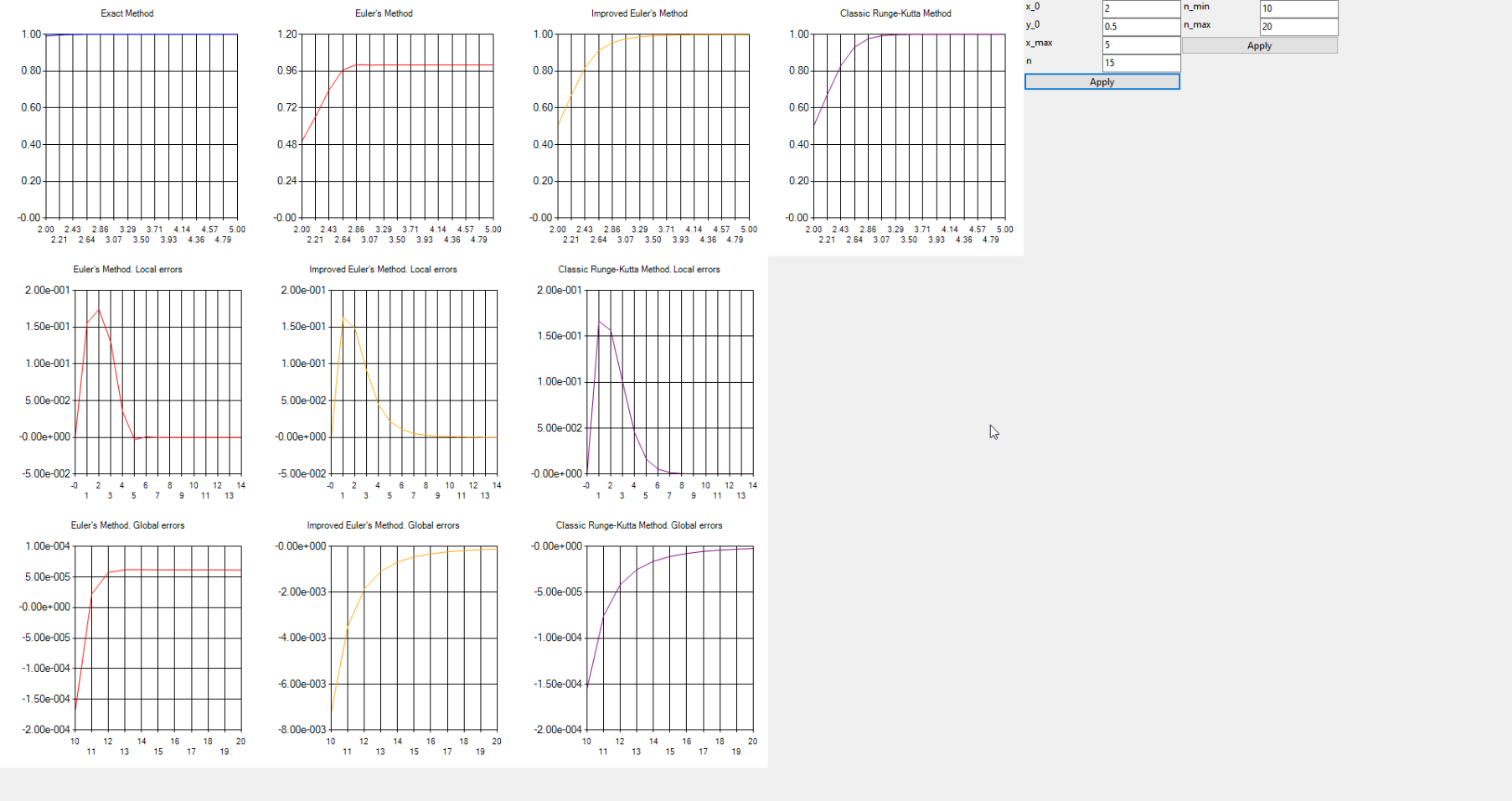
* IVP (both x0 and y0)
* X
* N
* Nmin and Nmax for global error analysis

This screenshot shows that we can vary different parameters. For example, here N is changed from 10 to 20 in order to increase accuracy. We can observe this increase in accuracy by comparing this screenshot’s local errors charts with the previous one. This is a brief explanation of how this input works:

1. Users puts values inside the fields and presses “Apply” button.
2. Program validates user input (e.g. checks that N is greater or equal than 2, because other values of N do not make sense).
3. If user input fails to pass validation procedure, errors message is displayed and nothing is updated.
4. If input is valid, every chart is updated with new values.

That is, charts are updated only if input values are changed. This becomes possibly via design pattern Observer that is implemented as language feature of C# (events and delegates).

It is worth mentioning that for some values of parameters, function’s value is either too big/small of NaN (i.e. it is not representable at the graph). In this case, point is not shown at all. Nevertheless, there are no such points with my variant’s IVP and X. They may appear only if we change IVP and X.

The following screenshots show variation of other parameters and their effect on errors:

Part 3

|  |  |
| --- | --- |
| Note: it is known that Classic Runge-Kutta method is more accurate than Improved Euler’s method, which is more accurate then Euler’s method. We will need this info in future. | |
| Let’s investigate convergence of different methods with the given initial value problem and find optimal N. We need to find an appropriate range of N. | |
| 1 |  |
| [10, 15] does not represent the whole picture: Euler’s Method is less accurate then Improved Euler’s method, but on the picture it is vice versa. Let’s increate both Nmin and Nmax. |
| 2 |  |
| [50, 60]is large enough. Every chart’s function decreases by its absolute value as argument (N) increases. |
| 3 |  |
| Range is [50, 200] (it is not visible due to the small grid size).  There are some N where graph is not smooth. The reason for them may be inaccuracy of floating-point numbers. |