Differential Equations:

Computational Practicum

Function to examine: $y' = e^{-\sin(x)} - y * \cos(x)$

The function is a first order linear non-homogeneous differential equation. There are 2 ways of knowing what it looks like on the graph: using numerical methods and analytical method of solving differential equations.

Regarding the first methods of solving differential equations, there are 3 of them in the program: Euler's method, Improved Euler's method and method of Runge-Kutta. First one provides approximate solution with truncation error of 1; Improved Euler's method provides approximation with truncation error up to 2nd level; and Runge-Kutta can provide approximation even better with truncation error of 4. In other words, method of Runge-Kutta is the most accurate out of all of them.

Exact solution of the function is: $y = e^{-\sin(x)} * (x + C)$, where **C** is some constant. It is calculated by the formula: $C = (y - e^{-\sin(x)} * x) / e^{(-\sin(x))}$, with x and y referring to initial value parameters. This is the solution provided by analytical method of solving differential equations. It is the most accurate way of solving differential equations, but it is not always possible to use it; thus, numerical methods exist and are used.

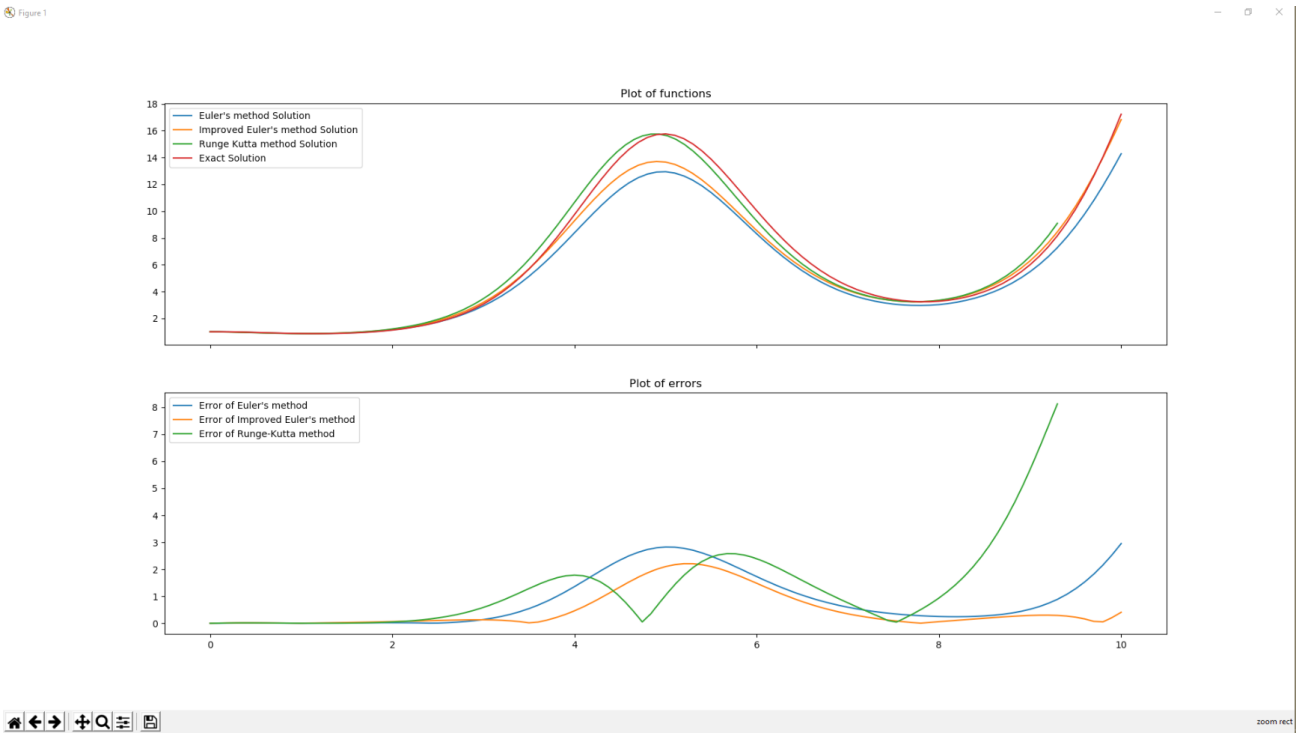
The program consists of 3 modules: one for solving differential equations using various methods, one for making a plot based on coordinates, and one for bonding them together: diffeqssolver, graph, main, respectively.

Diffeqssolver.py is a module which consists of classes and methods. Methods were added to the module for easiness of use of it; but classes are used to make it easier to edit code of the program. In other words, classes are for those who will try to edit structure of computational logic of the program, methods are for users of the module.

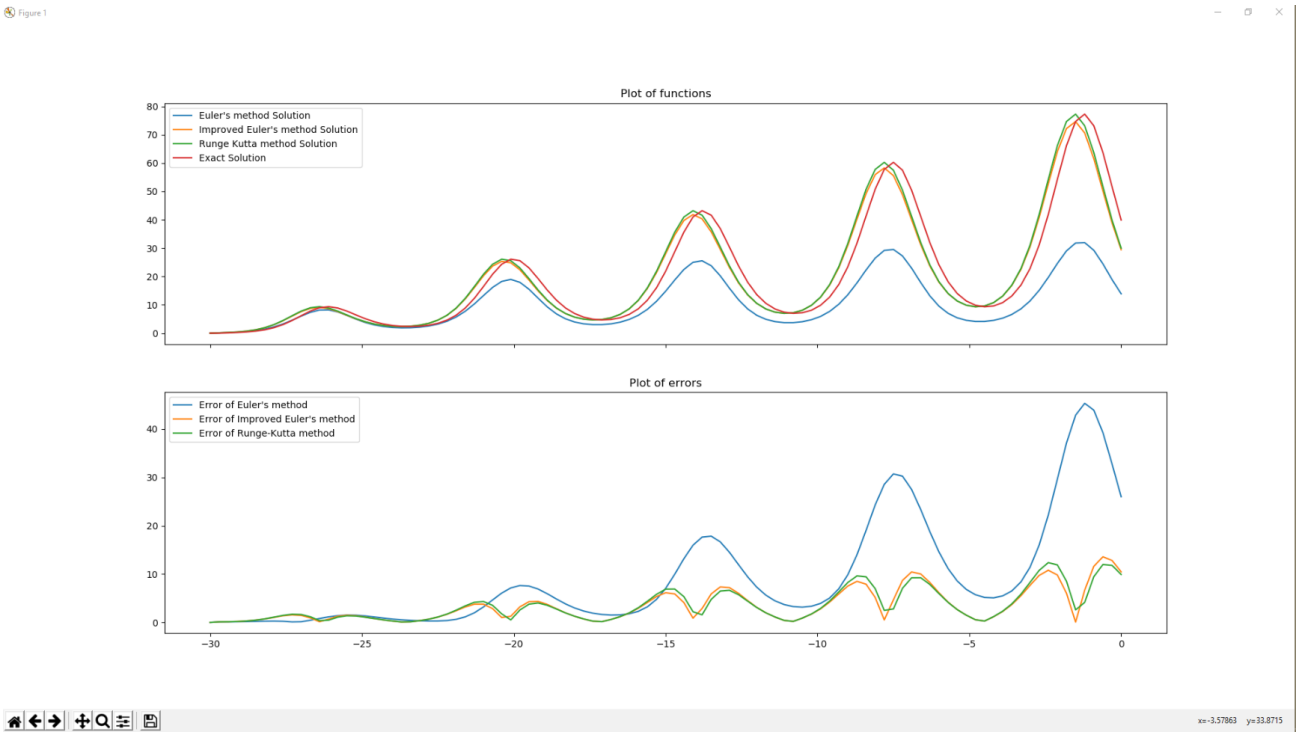
Graph.py provides methods for adding plots of functions and errors of computational methods and ability to clear the plots and show them.

Main.py uses the modules described above and provides almost infinite input to user and on every new set of data also provides plot of errors and functions.

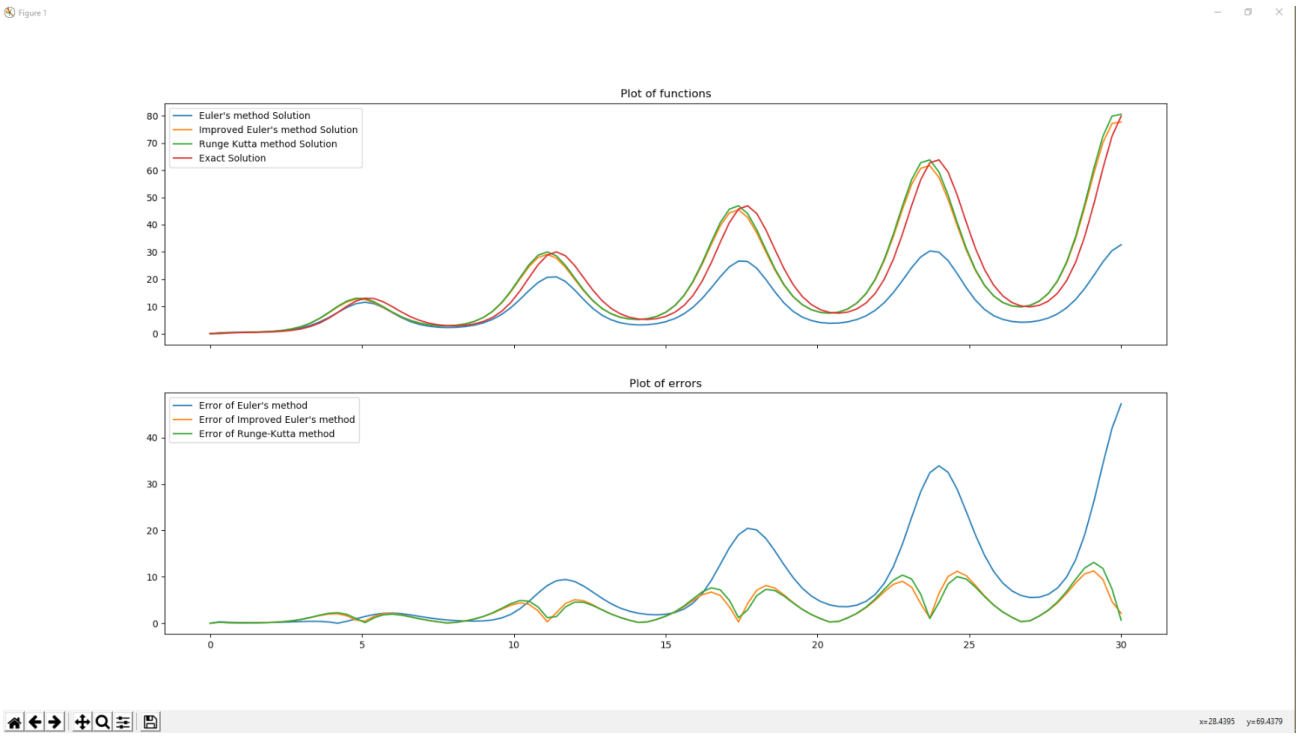
Following graphs are obtained by the program. Upper plot is for function while the lower one is a plot of local errors.



$x_0 = 0$, $y_0 = 1$, $x_{final} = 9.3$, number of steps = 101, constant for the exact solution = 1



$x_0 = -30$, $y_0 = 0$, $x_{final} = 0$, number of steps = 101, constant for the exact solution = 30



$x_0 = 0$, $y_0 = 0$, $x_{final} = 30$, number of steps = 101, constant for the exact solution = 0

Plots show that Euler's method is less accurate than method of Runge-Kutta; later one is closest of the 3 to the graph of exact solution

Kuspakov Amadey.