Computational Practicum Report

By Kamil Saitov, BS17-06

The equation:

$y' = sin^2(x) + yctg(x)$

Analytical solution

Make a substitution:

$y = y_1 * u$

Solve complementary equation:

$y_1' - y_1 ctg(x) = 0$

$\frac{dy_1}{dx} = y_1 ctg(x)$

$\int \frac{dy_1}{y_1} = \int ctg(x) dx$

$y_1 = sin(x)$

$\frac{du}{dx} = \frac{sin^2(x)}{sin(x)}$

$\int du = \int sin(x) dx$

$u = -cos(x) + C$

General solution:

$$y = Csin(x) - cos(x)sin(x)$$

Initial Value Problem:

$$x_0 = 1; \quad y_0 = 1$$

$$1 = -cos(1)sin(1) + Csin(1)$$

$$C = \frac{1 + cos(1)sin(1)}{sin(1)} = 1.728...$$

Final answer for IVP:

$$y = 1.728 * sin(x) - cos(x)sin(x)$$

Numerical solution

For numerical solution, three methods are used to compare their accuracy:

- **Euler** method
- **Improved Euler** method
- **Runge-Kutta** method

All of them are compared to the exact solution.

Euler method

It is expected to be the least accurate method of all. The formula of i^{th} term is:
$y_i = y_{i-1} + n * (f(x_{i-1}, y_{i-1}))$

Improved Euler method

It is expected to show more accuracy than the first. The formula of i^{th} term is:
$k_i = y_{i-1} + n * f(x_{i-1}, y_{i-1})$

$y_i = y_{i-1} + \frac{n}{2} * (f(x_{i-1}, y_{i-1}) + f(x_i, k))$

Runge-Kutta method

It is expected to have the best accuracy of all three methods. The formula of i^{th} term is:

$k_1 = n * f(x_{i-1}, y_{i-1})$

$k_2 = n * f(x_{i-1} + \frac{n}{2}, y_{i-1} + \frac{k_1}{2})$

$y_i = y_{i-1} + k_2$

Design decisions

The app is written in **Python**, with **Math** (for solving the equation), **Matplotlib** (for plotting the graphs) and **Tkinter** (GUI) libraries.

Classes:

main:

- Initializes Tkinter
- Adds all the buttons and entry fields
- On button press executes on of the following methods:
  compute_graphs - plots the solution graphs for given values via **Plot** class
  compute_error - plots the error function graph for given values via **Plot** class
  compute_errN - plots the “Max error for delta” function graph for given values via **Plot** class

Plot:

- Is used for plotting the graphs computed by Computation class
- Has three methods:
  plot_graphs - takes the computed values and plots four graphs: Euler, Improved Euler, Runge-Kutta and Exact (analytical solution function)
- **plot_local_error_graphs -** for each of the methods plots a difference between a method’s value and exact value
- **plot_max_error_graph -** computes plots for each of the method multiple times, each with different step size and computes the maximum error for each step. X axis is the step size, Y axis is the error

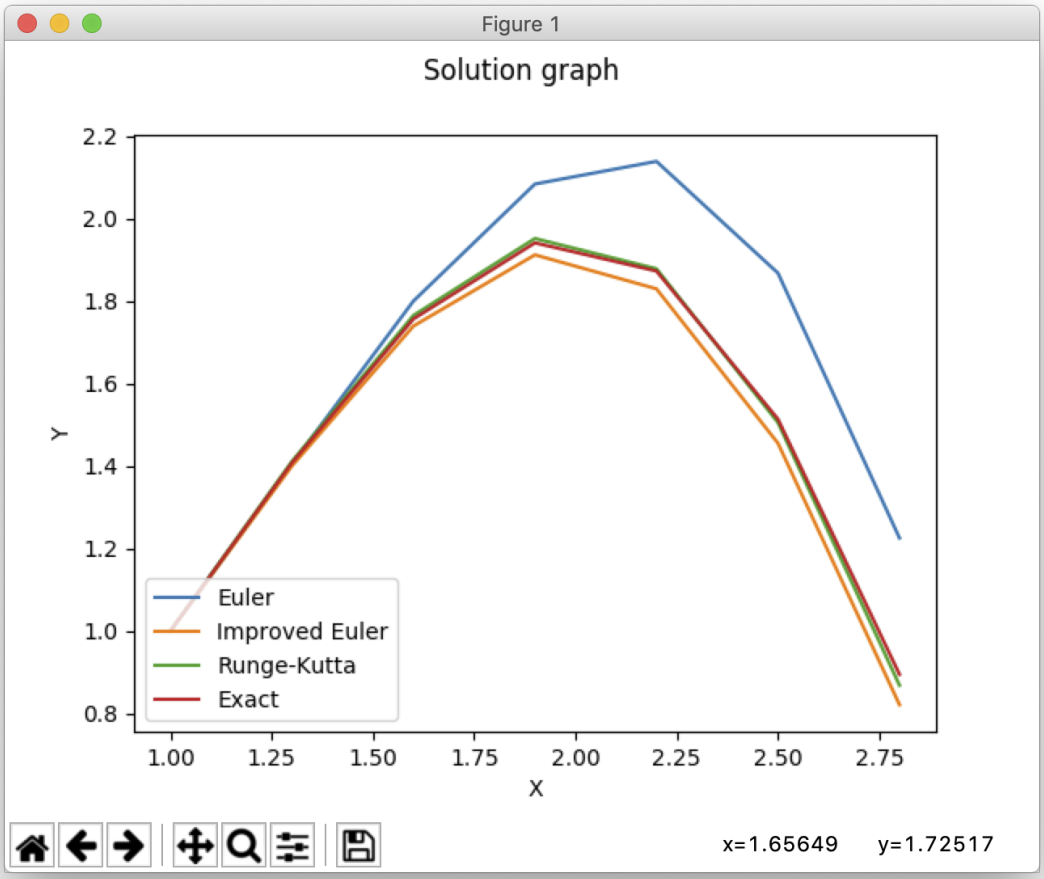
Computation:

- The method which solves and approximates everything.
- **funct -** the function to approximate
- **exact -** the exact (analytical) solution of the equation

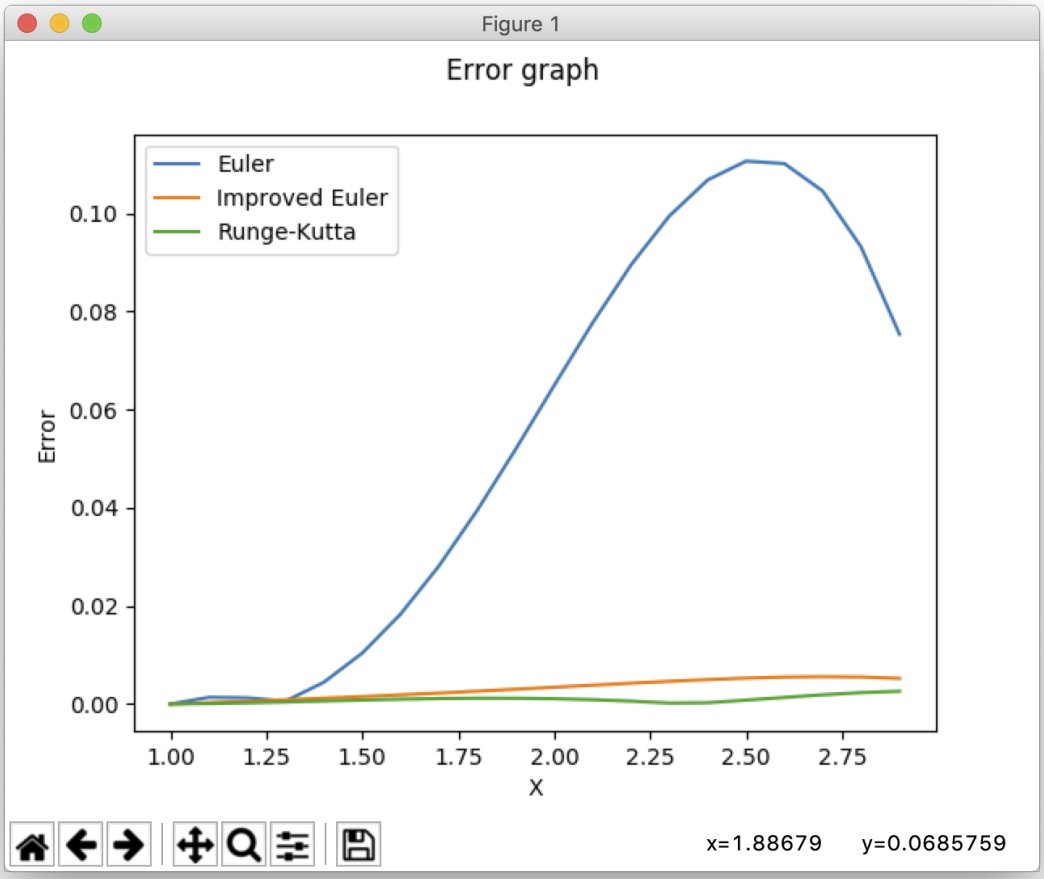
- **calculate_c -** calculating the constant as in the analytical solution (needs to be a function because we may change IVP)
- **euler, improved_euler, runge_kutta, exact_filling -** methods that fill the solution array with the solutions for each of the methods (all methods return array of X values and array of Y values)
- **compute -** creates arrays for every method solutions, forms a total solution of all methods
- **get_local_errors -** computes the error for each of the step sizes, then turns that into a pyplot-compatible arrays.

Analysis

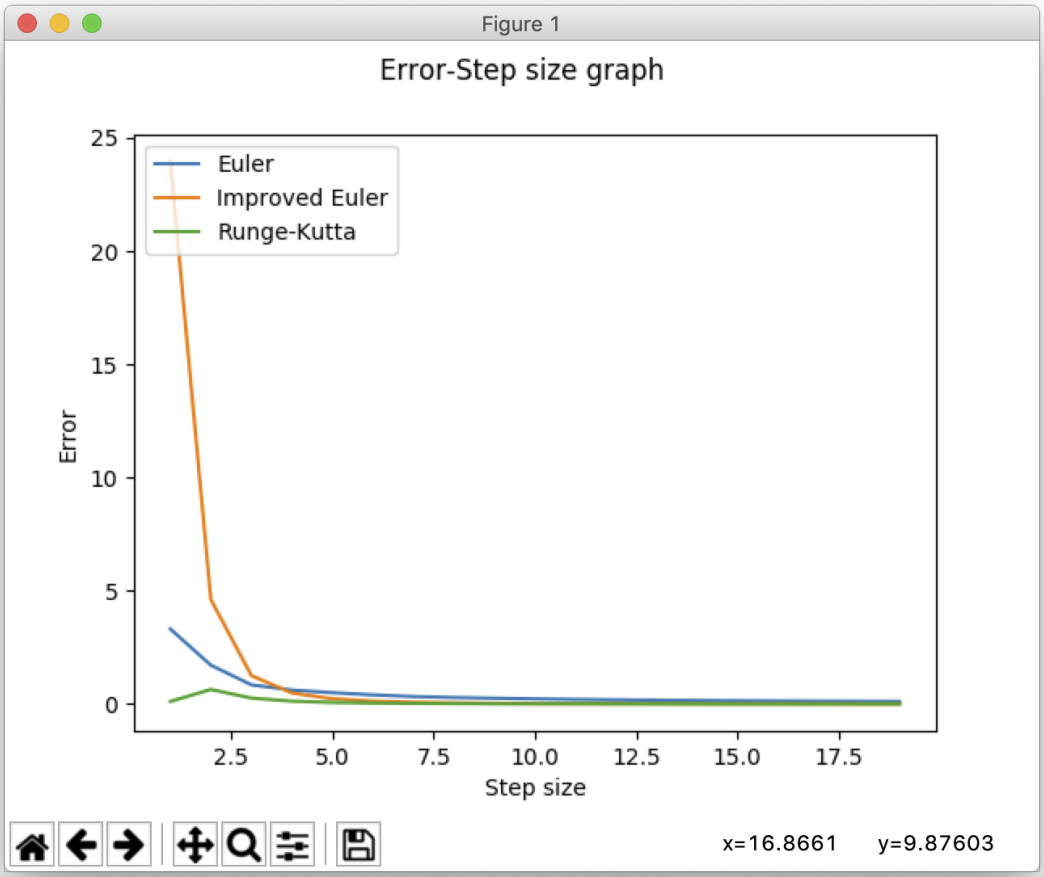
According to the solution graphs drawn by the program (step size set to 0.3 for obvious difference between the methods), the Runge-Kutta method turned out to be the most precise, Improved Euler is farther from Exact solution and the worst approximation is done by the Euler method.



Here is the local error graph for each of the three methods. While Runge-Kutta holds the least error among all, the Improved Euler method is pretty close to it. However, Euler method's error is rising rapidly until about 3/4 of the range, but despite a comparable decline, still is far from being as accurate as the other two methods.



The third graph shows that with the increasing step the error between each of the methods decreases rapidly - even the Euler method which showed the worst performance of all, started to visually merge (and thus has the comparable insignificant difference) with the other graphs. Nothing can be said about the first few points of the graph: depending on the variables set, randomly methods can show very high or very low approximation error, all because this small step cannot reach acceptable approximation even for the most vague calculations.



Conclusion

Overall, by comparing the graphs, we see that mainly the most accurate method of numerical integral solution is Runge-Kutta, then goes Improved Euler method, and lastly Euler method. If use of few steps is needed (i.e. on low-performance machines), then Runge-Kutta is the obvious choice, however, decreasing the delta makes all three methods comparable to each other (with Runge-Kutta still leader, it must be said).