Differential Equations Task

Dragos Strugar, BS17-05

October 2018

1 Problem Statement

Use numerical methods to solve the initial value problem. Use Euler's method, Enhanced Euler's method and Runge-Kutta method. Plot the results. Explain the results. Compare your results with the exact solution. Plot the error graph. Show the implementation.

My variant: 12, i.e. $\frac{dy}{dx} = 3xy + xy^2$.

2 General Solution and Initial-Value-Problem (IVP)

First, let us solve the equation analytically. Notice that the equation is separable.

$$\frac{dy(x)}{dx} = x(y(x) + 3)y(x) \tag{1}$$

$$\frac{\frac{dy(x)}{dx}}{(y(x) + 3)y(x)} = x \tag{2}$$

$$\int \frac{\frac{dy(x)}{dx}}{(y(x) + 3)y(x)} = \int x dx \tag{3}$$

$$\frac{1}{3}\log(y(x)) - \frac{1}{3}log(y(x) + 3) = \frac{x^2}{2} + c \tag{4}$$

$$-\frac{3exp(3c1 + \frac{3x^2}{2})}{exp(3c1 + \frac{3x^2}{2}) - 1} \tag{5}$$

Therefore, our solution is:

$$y(x) = -\frac{3exp(\frac{3x^2}{2})}{exp(\frac{3x^2}{2}) - 2} \tag{6}$$

3 A few observations

Note that the solution has following properties:

- parity: even
- limit as x approaches +-inf = -3

- most importantly, has the following asymptotes:
  - horizontal: -3
  - vertical: $\pm\sqrt{\frac{2log(2)}{3}}$

Graph of the function is presented below.

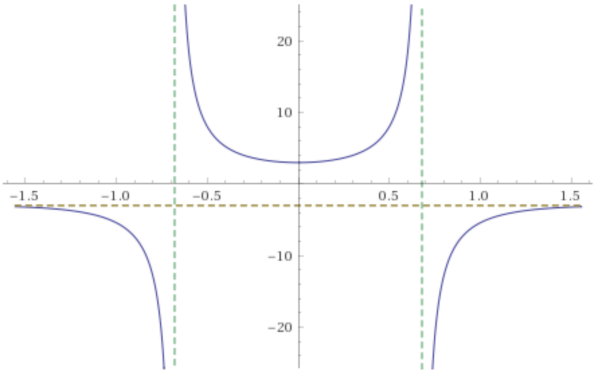

Figure 1: Plot of the solution

Please note that the program that corresponds to this problem calculates the integration constant and asymptotes on the go, based on the values of IVP.

4 Implementation

I choose to go with MATLAB programming language and its environment, as it it best-suited for numerical tasks like this. The repository of the code is available on GitHub: on this link.

5 Results

The two figures provided in this PDF are the results of running the MATLAB program available on GitHub. The green continuous line presents the exact solution of the differential equation. Red circles represent the Euler method approximation. Blue xs represent the Improved Euler Method, while magenta stars represent the Runge Kutta method. Also the error figure is provided, which shows that Runge Kutta methods offers the best approximation of the three methods.

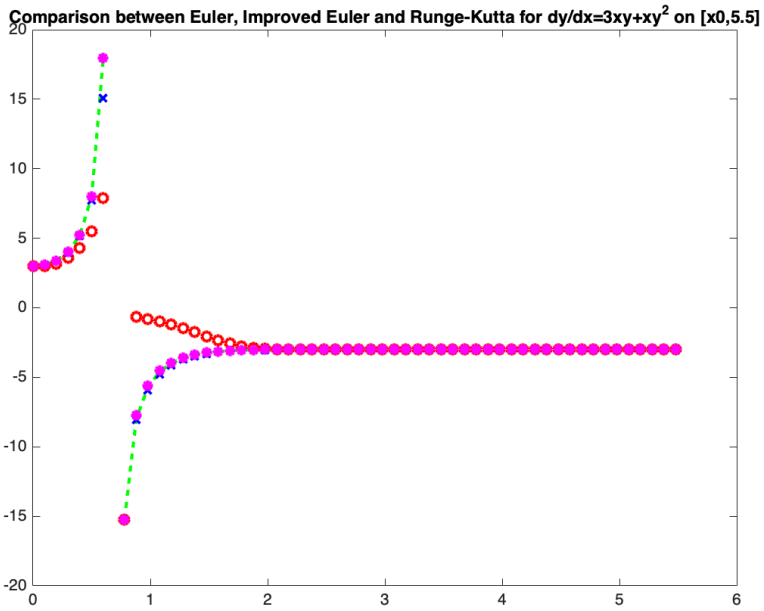

Figure 2: Results

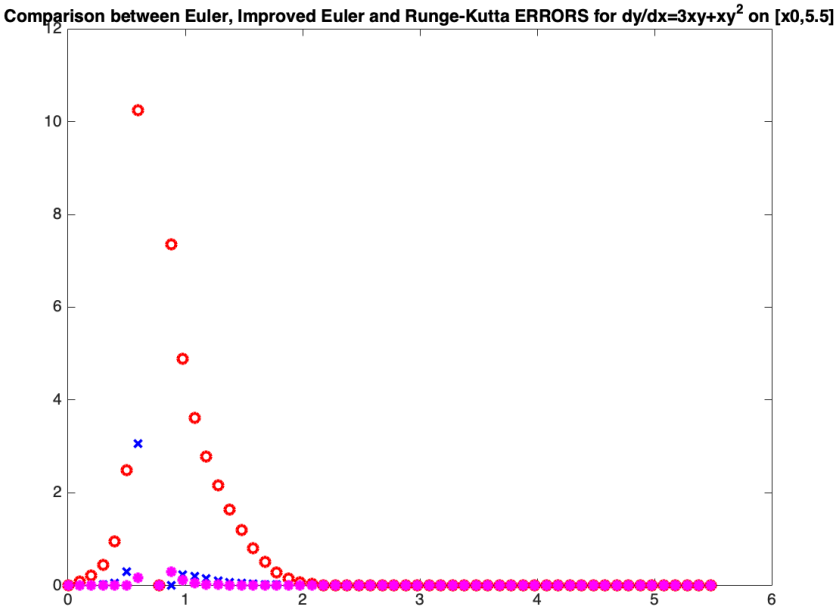

Figure 3: Errors

6 Implementation Details

6.1 Why MATLAB (Matrix Laboratory)

MATLAB has numerous advantages over other languages for performing numerical tasks like this:

- Basic data element is the matrix. A simple integer is matrix of one row and one column. Several mathematical operations that work on arrays or matrices are built-in to the MATLAB environment. For example, cross-products, dot-products, etc.
- Visually appealing plotting library
- Vectorized operations - adding two arrays together needs only one command, instead of a for or while loop

These are some of the reasons I went ahead and did this task using this specific language. My second option would be Python, as it also works very well with numbers, especially with NumPy and MatPlotLib libraries.

6.2 Numerical Methods

Our task was to use the numerical methods for solving differential equations to approximate the value of some (possibly discontinuous) solution of that particular equation. As said in the introductory part, the equation I was given is $dy/dx = 3xy + xy^2$. It can be clearly seen that this function is even, and thus has two asymptotes unless the integration constant ($c = (log(ya/(ya + 3))/3) - (0.5 * a^2)$ equals $-0.5 * a^2$).

6.3 Process

I first started by implementing the scenario in which there would be no change in the initial value problem. Then, I generalized the solution, allowing the user to enter the step, and the initial values.

I implemented all the methods as separate functions, available in 3 different external files. These files, and thereafter functions, are imported to the main.m file which acts like a main function in most languages.

6.3.1 Euler Method

The code of the Euler method is available here. It is pretty short and straight-forward. Exports the x trace and corresponding approximations at each element of the trace.

6.3.2 Improved Euler Method

The code of the Euler method is available here. This method approximates better than the standard Euler method. Exports the same values as Euler method for x trace, with the corresponding values of y using this very method.

6.3.3 Runge Kutta

The code of the Runge Kutta method is available here. Runge Kutta method performs the best out of these three methods. How much better is presented on the error graph provided in this document.

6.4 Other files

After implementing these three functions, I started other pieces of code that I needed. First of them was the differential equation itself. The source code of it is available here. It simply returns the value of $xy^2 + 3xy$; given the values of x and y.

Also, I implemented the formula for calculating the solution of the Differential equation. Code is here. It calculates the value of function at point x, with integrating factor c.

6.5 Glue Code

And finally, everything is combined in the main.m file. It starts by asking the user to input all the properties of the initial value problem. Then, based on the function, calculates the integrating factor and all possible asymptotes.

As numerical methods do not work when they come across an asymptote, it is necessary to observe that the problem can be broken down into three major cases:

1. number of continuous sections is one - has no asymptotes in range
2. number of continuous sections is two - has 1 asymptote in range
3. number of continuous sections is three - has 2 asymptotes in range

After doing this, the rest of the code is responsible for applying the numerical methods to these intervals, plotting the results and generating errors. It then saves the plots as .png files, available for user to analyze.

7 Error as function

The code also provides a graph that tells us that the error decreases as the number of steps increases, or as the step decreases. This observation is rather obvious, but still deserves to be mentioned. The graph is available on GitHub.

8 Wrap up

My solution written in MATLAB programming language satisfies all the requirements presented to us in the tutorial session of the Differential Equations course. It provides the source code, and corresponding documentation that explains all the ideas and implementation behind the scenes. Then it showcases the results. Concludes that the best numerical approximation method is Runge Kutta (out of the three we were to implement).