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. $\dfrac{dy}{dx} = 3xy + xy^2$.

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

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

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

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

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

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

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

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.

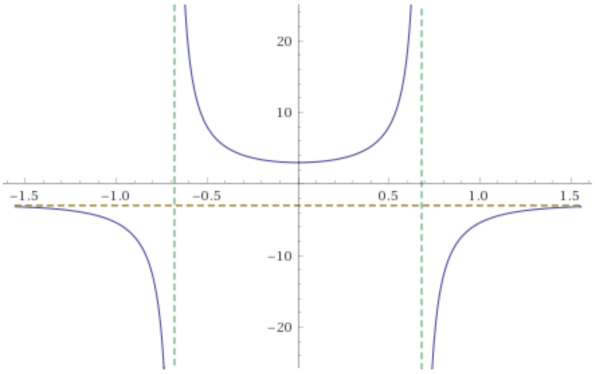

Figure 1: Plot of the solution

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:
here

Figure 2: Problem 12 translated to code

5 Results

The plot of all the methods:

```matlab
% performs euler method on function f
% on range [xinit, xend]
% y(xinit) = yinit
% h - step
function [x,y] = EulerMethod(f,xinit,xend,yinit,h)
    N = floor((xend-xinit)/h);

    x = zeros(1, N+1);
    y = zeros(1, N+1);

    x(1) = xinit;
    y(1) = yinit;

    for i=1:N
        x(i+1) = x(i)+h;
        y(i+1) = y(i) + h*feval(f,x(i),y(i));
    end

end
```

Figure 3: Euler Method

```matlab
% performs improved euler method on function f
% on range [xinit, xend]
% y(xinit) = yinit
% h - step
function [x,y] = ImprovedEulerMethod(f,xinit,xend,yinit,h)
    N = floor((xend-xinit)/h);

    x = [xinit zeros(1, N)];
    y = [yinit zeros(1, N)];

    for i=1:N
        x(i+1) = x(i)+h;
        ynew = y(i) + h*feval(f,x(i),y(i));
        y(i+1) = y(i) + h*feval(f,x(i),y(i))/2 + h*feval(f,x(i+1),ynew)/2;
    end

end
```

Figure 4: Improved Euler Method

```
% performs runge kutta method on function f
% on range [xinit, xend]
% y(xinit) = yinit
% h - step
function [ x, y ] = RungeKuttaMethod(f,xinit,xend,yinit,h)
    N = floor((xend-xinit)/h);

    x = xinit:h:xend;
    y = [yinit zeros(1, N)];

    for i=1:(length(x)-1)
        k1 = feval(f, x(i),y(i));
        k2 = feval(f,x(i)+0.5*h,y(i)+0.5*h*k1);
        k3 = feval(f,(x(i)+0.5*h),(y(i)+0.5*h*k2));
        k4 = feval(f,(x(i)+h),(y(i)+k3*h));

        y(i+1) = y(i) + (1/6)*(k1+2*k2+2*k3+k4)*h;
    end

end
```

Figure 5: Runge Kutta Method


```
% performs runge kutta method on function f
% on range [xinit, xend]
% y(xinit) = yinit
% h - step
function [ x, y ] = RungeKuttaMethod(f,xinit,xend,yinit,h)
    N = floor((xend-xinit)/h);

    x = xinit:h:xend;
    y = [yinit zeros(1, N)];

    for i=1:(length(x)-1)
        k1 = feval(f, x(i),y(i));
        k2 = feval(f,x(i)+0.5*h,y(i)+0.5*h*k1);
        k3 = feval(f,(x(i)+0.5*h),(y(i)+0.5*h*k2));
        k4 = feval(f,(x(i)+h),(y(i)+k3*h));

        y(i+1) = y(i) + (1/6)*(k1+2*k2+2*k3+k4)*h;
    end

end
```


Figure 6: Runge Kutta Method

```matlab
% MAIN DRIVER PROGRAM
% WRITTEN BY: DRAGOS STRUGAR, B17-05
% 15 OCT 2018 ONWARDS
% DIFFERENTIAL EQUATIONS COURSE
%
%
% TASK -- use numerical methods to approximate dy/dx = 3xy + xy^2
% TASK -- euler method, improved euler method, and runge kutta method
% TASK -- plot the solutions
% TASK -- give error graphs
%
%

% clears working space
close all;
clc;
clear;

% gives IVP - initial value problem, with step = 0.1
a = 0; b = 5.5; ya = 3; h = 0.1;
% as the problem has an asymptote, divide the plot, and problem into two
% parts: before and after the asymptote
asymptote = sqrt(2/3) * sqrt(log(2));

% t1 - vector, from initial x0 to asymptote, evenly distributed with step h
% t1 - vector, from asymptote to ending X, evenly distributed with step h
t1 = a:h:asymptote;
t2 = (asymptote+h):h:b;

% USE FILES:
% EulerMethod.m
% ImprovedEulerMethod.m
% RungeKutta.m
% Problem12.m specifies the equation dy/dx = 3xy + xy^2
% Can be included as ''

% EULER METHOD, before the asymptote and after
[x1eul,y1eul] = EulerMethod('Problem12', a, asymptote, ya, h);
[x2eul,y2eul] = EulerMethod('Problem12', asymptote+h, b, -15.2575, h); % -15.2575 = sol. at asymptote + h

% IMPROVED EULER METHOD, before the asymptote and after
[x1improved,y1improved] = ImprovedEulerMethod('Problem12', a, asymptote, ya, h);
[x2improved,y2improved] = ImprovedEulerMethod('Problem12', asymptote+h, b, -15.2575, h);

% RUNGE-KUTTA METHOD, before the asymptote and after
[x1rk, y1rk] = RungeKuttaMethod('Problem12', a, asymptote, ya, h);
[x2rk, y2rk] = RungeKuttaMethod('Problem12', asymptote+h, b, -15.2575, h);

% EXACT SOLUTION OF THE PROBLEM, before the asymptote and after
sq1 = x1improved.^2;
sq2 = x2improved.^2;
yi1 = -1*(3*exp(3/2*sq1))./(exp(3/2*sq1)-2);
yi2 = -1*(3*exp(3/2*sq2))./(exp(3/2*sq2)-2);

% PLOT RESULTS
hold on;

% plot euler
plot(t1, y1eul, 'ro', 'LineWidth', 2);
plot(t2, y2eul, 'ro', 'LineWidth', 2);

% plot improved euler
plot(t1, y1improved, 'bx', 'LineWidth', 2);
plot(t2, y2improved, 'bx', 'LineWidth', 2);

% plot rk
plot(t1, y1rk, 'y*', 'LineWidth', 2);
plot(t2, y2rk, 'y*', 'LineWidth', 2);

% plot exact solution
plot(t1, yi1, 'g--', 'LineWidth', 2);
plot(t2, yi2, 'g--', 'LineWidth', 2);

hold off;
% ADD A BIT OF EXTRA INFO TO THE PLOT
title('Comparison between Euler, Improved Euler and Runge-Kutta for dy/dx=3xy+xy^2 on [0,5.5]');
legend('euler', 'euler', 'improved euler', 'improved euler','runge-kutta','runge-kutta', 'exact', 'exact');
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
box on;

saveas(gcf, 'results.png')

% PLOT ALL ERRORS
% MAKE another graph

% euler method errors
eul1err = abs(yi1 - y1eul);
eul2err = abs(yi2 - y2eul);

% TOTAL EULER ERROR
eul_err = sum(eul1err) + sum(eul2err);

% euler method errors
imp1err = abs(yi1 - y1improved);
imp2err = abs(yi2 - y2improved);

% TOTAL IMPROVED EULER ERROR
imp_err = sum(imp1err) + sum(imp2err);

% runge kutta errors
rk1err = abs(yi1 - y1rk);
rk2err = abs(yi2 - y2rk);

% TOTAL RUNGE KUTTA ERROR
rk_err = sum(rk1err) + sum(rk2err);

% PLOT ERROR RESULTS
figure();
hold on;
% euler
plot(t1, eul1err, 'ro', 'LineWidth', 2);
plot(t2, eul2err, 'ro', 'LineWidth', 2);

% improved euler
plot(t1, imp1err, 'bx', 'LineWidth', 2);
plot(t2, imp2err, 'bx', 'LineWidth', 2);

% runge kutta
plot(t1, rk1err, 'y*', 'LineWidth', 2);
plot(t2, rk2err, 'y*', 'LineWidth', 2);

% add more information
hold off;
title('Comparison between Euler, Improved Euler and Runge-Kutta ERRORS for dy/dx=3xy+xy^2 on [0,5.5]');
legend('euler erorr', 'euler erorr', 'improved euler erorr', 'improved euler erorr','runge-kutta
erorr','runge-kutta erorr');
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
box on;

% save results in error_results.png
saveas(gcf, 'error_results.png')
```


Figure 7: Driver Program

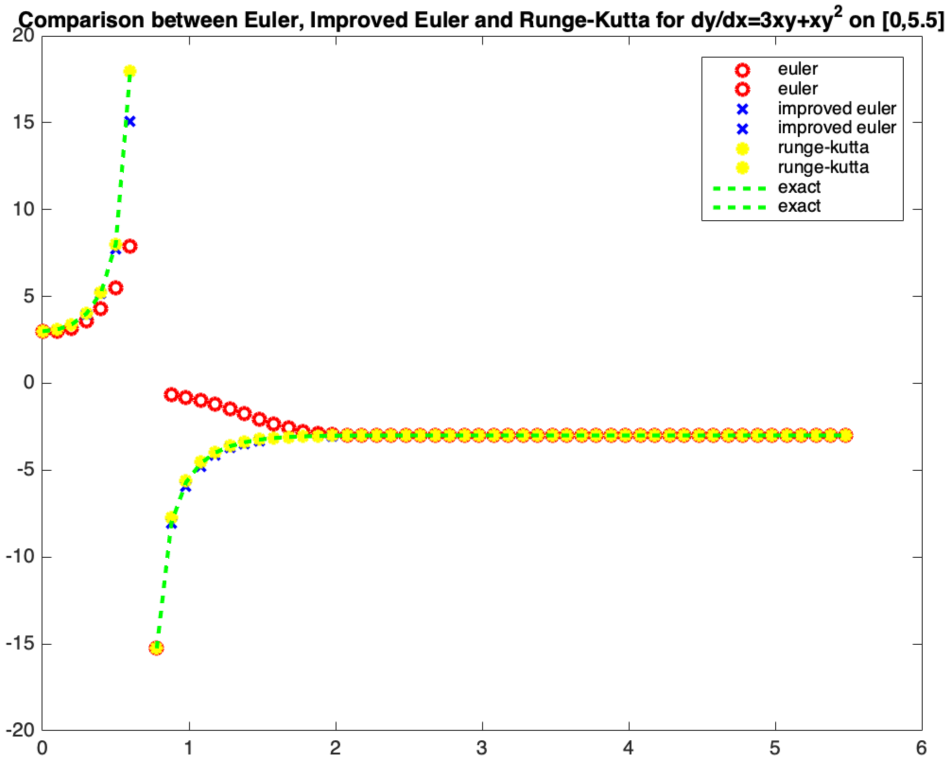

Figure 8: Results

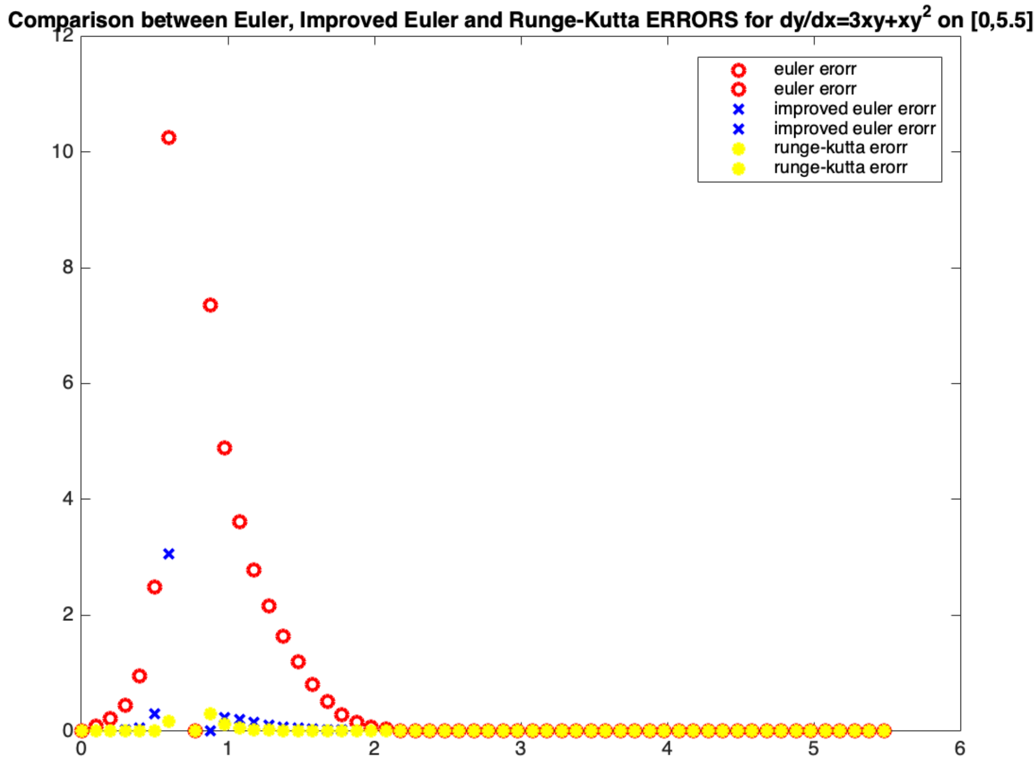

Figure 9: Errors