DSP HW1.

Numerical methods for ODE in SciLab

Kamil Akhmetov, B17-DS-01

Initial Value (Cauchy) Problem:

$$y' = -y + ax^2 + bx + c : \begin{cases} y(0) = 1 \\ a = 4 \\ b = 12 \\ c = 98 \\ I = [0; 4], \text{ interval of interest} \end{cases}$$

Exact solution:

$$y' = -y + 4x^2 + 12x + 98$$
$$y' + y = 4x^2 + 12x + 98$$
which is Non-Homogeneous Linear ODE of the 1st order
Solve by substitution:
$$\begin{bmatrix} y = uv \\ y' = u'v + uv' \end{bmatrix}$$
$$y + y' = uv + u'v + uv' = u(v + v') + u'v = u'v$$
$$v + v' = 0$$
$$v + \frac{dv}{dx} = 0$$
$$\frac{dv}{dx} = -v$$
$$\frac{dv}{v} = -dx$$
$$\int \frac{1}{v} dv = -\int dx$$
$$\ln v = -x + C, C \in \mathbf{R}$$
$$C = 0 \implies v = e^{-x}$$
Then
$$\begin{cases} y' + y = 4x^2 + 12x + 98 \\ y' + y = u'v = u'e^{-x} \end{cases} \implies 4x^2 + 12x + 98 = u'e^{-x} \implies$$
$$\implies \frac{du}{dx} = e^x(4x^2 + 12x + 98)$$
$$\int du = \int e^x(4x^2 + 12x + 98)dx$$
$$= \int 4x^2 e^x dx + \int 12xe^x dx + \int 98e^x dx$$
Solving by parts gives us:
$$u = (4x^2 + 4x + 94)e^x + C, C \in \mathbf{R}$$
$$y = uv = ((4x^2 + 4x + 94)e^x + C)e^{-x}$$
$$\implies y = 4x^2 + 4x + 94 + Ce^{-x}$$
Solving the Initial Value Problem
$$\begin{cases} y = 4x^2 + 4x + 94 + Ce^{-x} \\ y(0) = 1 \end{cases} \implies C = -93$$
Exact solution:
$$y = 4x^2 + 4x + 94 - 93e^{-x}$$

Numerical methods:

First, define the problem statement as the following:

$$\begin{cases} y' = f(x,y) \\ y(x_0) = y_0 \end{cases} \text{ where } \begin{cases} f(x,y) = -y + 4x^2 + 12x + 98 \\ x_0 = 0 \\ y_0 = 1 \end{cases}$$

Second, solve using different numerical methods and compare results against the exact solution:

- **Euler method**

```
 function y=solve_euler(y0,x,h,f)
    y = zeros(x) // (1xN)
    d = zeros(size(x)(2)-1)
    y(1) = y0
    for i = 1:(size(x)(2)-1)
        d(i) = h * f(x(i), y(i))
        y(i+1) = y(i) + d(i)
    end
endfunction
```

- **Improved Euler method**

```
 function y=solve_euler_improved(y0,x,h,f)
    y = zeros(x) // (1xN)
    y(1) = y0
    for i = 1:(size(x)(2)-1)
        m1 = f(x(i), y(i))
        m2 = f(x(i + 1), y(i) + h * m1)
        y(i + 1) = (y(i) + h * (m1 + m2) / 2)
    end
endfunction
```

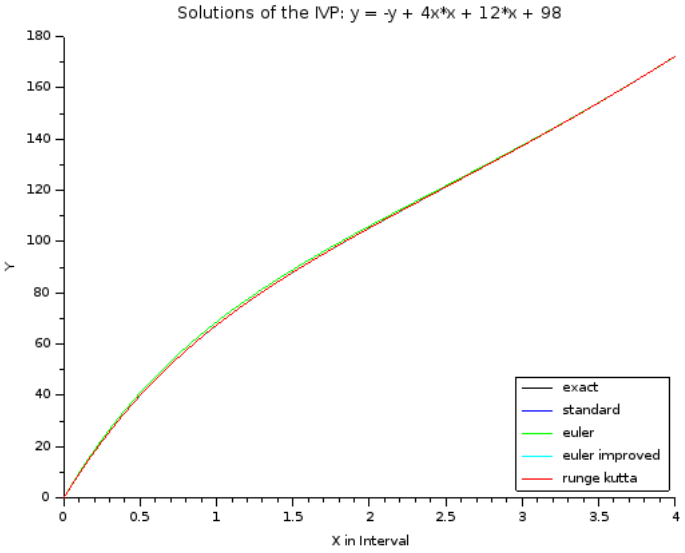
- **Runge Kutta method**

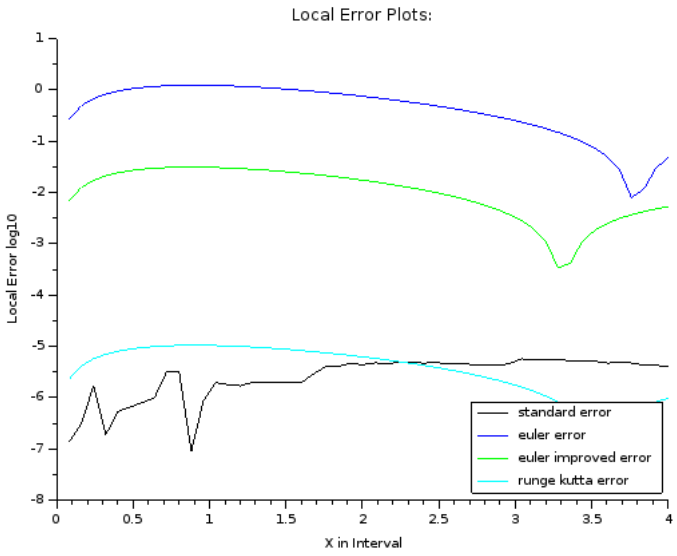
```
 function y=solve_runge_kutta(y0,x,h,f)
    y = zeros(x)
    y(1) = y0
    for i = 1:(size(x)(2)-1)
        k1 = f(x(i), y(i))
        k2 = f(x(i) + h / 2, y(i) + h / 2 * k1)
        k3 = f(x(i) + h / 2, y(i) + h / 2 * k2)
        k4 = f(x(i) + h, y(i) + h * k3)

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

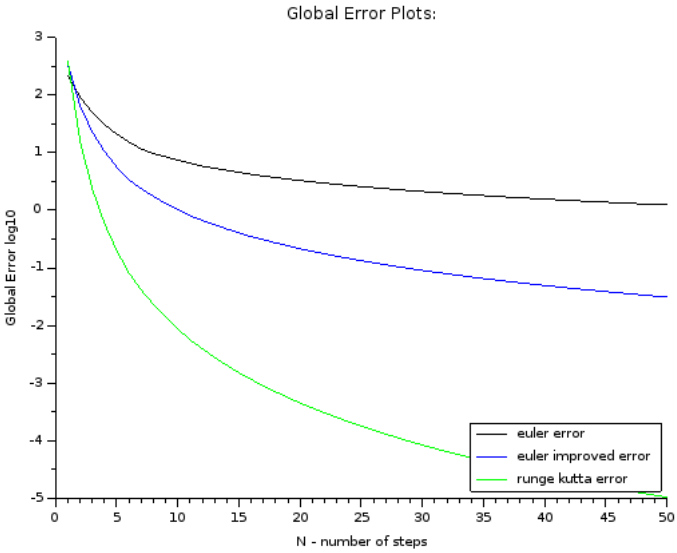
- **Rest of the code in repo:** `https://github.com/kamilkoduo/s20-dsp`

Results

We used N=50 (number of steps) for demonstration:







Method	Global error
Standard ODE	0.0000057
Euler	1.2315641
Euler Improved	0.0310862
Runge Kutta	0.0000104

Conclusions

As we can see, errors of the numerical methods in our setting are of the quite low order, that is why we used logarithmic scale to display the differences in accuracy of applied techniques.

In the setting of $N = 50$, the lowest global error was shown by the standard SciLab's ODE solution. The next nearest best solution is the `Runge Kutta` method. And both `Euler` methods have much lower accuracy.

On the last graph notice that the increase in the number of steps causes decrease in the global error for all of the methods. But `Runge Kutta` method improves accuracy significantly faster and results in error order of 10^{-5} at the step $N = 50$.