0 Instructions

This problem set is **due on Tuesday, Nov. 19 by midnight**. Submit your solutions on Canvas. Include the names of everyone you worked with on this problem set. Include any code you used to solve the problems as part of your submission.

1 Practice

Problem 1.1. Explain in your own words why local truncation errors are useful, even though they do not directly measure the error of a numerical method for solving differential equations.

Problem 1.2. The graph of a single-variable function $y = y(x)$ has *curvature* given by the formula

$$\frac{y''}{(1+y'^2)^{\frac{3}{2}}}.$$

An interesting question is whether there is a function whose value is proportional to its own curvature, which provides the differential equation

$$y'' = -y(1+y'^2)^{\frac{3}{2}}.$$

Transform this equation into a first order system consisting of two equations by introducing

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix}$$

and appropriately defining the system $\mathbf{u}' = \mathbf{f}(t, \mathbf{u})$.

Problem 1.3. Compute four timesteps of Euler's method to approximate a solution to the system you found above. Start from $x = 0$ and use the initial conditions $y(0) = 0$ and $y'(0) = \sqrt{3}$ with a timestep size of $h = 0.5$.

2 Implementing basic solvers

In this problem, you will implement several methods for solving systems of first order differential equations. Recall that such systems have the format

$$\begin{aligned} y_1' &= f_1(t, y_1(t), \ldots, y_m(t)) \\ y_2' &= f_2(t, y_1(t), \ldots, y_m(t)) \\ &\vdots \\ y_m' &= f_m(t, y_1(t), \ldots, y_m(t)) \end{aligned}$$

for $t \in [a, b]$ along with initial conditions

$$y_i(a) = \alpha_i.$$

It is often convenient to think of y and f as vectors:

$$\mathbf{y}' = \begin{pmatrix} y_1' \\ \vdots \\ y_m' \end{pmatrix} = \begin{pmatrix} f_1(t, y_1(t), \ldots, y_m(t)) \\ \vdots \\ f_m(t, y_1(t), \ldots, y_m(t)) \end{pmatrix} = \mathbf{f}(t, \mathbf{y})$$

where the boldfaced variables indicate vectors. In your code, you may find it helpful to use numpy to provide vector operations. Each of the methods you implement below will require the following input data:

- the function $\mathbf{f}(t, \mathbf{y})$ which models the differential equation $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$
- the endpoints of the interval $[a, b]$ over which to approximate the solution
- the initial value $\mathbf{y}_0 = \mathbf{y}(a)$
- the number of timesteps n to use. (This means your timestep is $h = \frac{b-a}{n}$.)

and should output the solution trajectory

$$[\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_n].$$

Problem 2.1. Implement Euler's method. Remember that Euler's method is the following:

1. begin with the initial value $\mathbf{w}_0 = \mathbf{y}_0$.
2. for each $i = 0, \ldots, n-1$, approximate $\mathbf{w}_{i+1} = \mathbf{w}_i + h\mathbf{f}(t_i, \mathbf{w}_i)$.

Problem 2.2. Implement the (explicit) Runge-Kutta method of order 4 specified by the following Butcher array

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}.$$

Remember that the Butcher array above encodes the following steps

1. begin with the initial value $\mathbf{w}_0 = \mathbf{y}_0$
2. for each $i = 0, \ldots, n-1$, approximate $\mathbf{w}_{i+1}$ as follows:

$$\mathbf{k}_1 = \mathbf{f}(t_i, \mathbf{w}_i)$$

$$\mathbf{k}_2 = \mathbf{f}(t_i + \frac{h}{2}, \mathbf{w}_i + \frac{h}{2}\mathbf{k}_1)$$

$$\mathbf{k}_3 = \mathbf{f}(t_i + \frac{h}{2}, \mathbf{w}_i + \frac{h}{2}\mathbf{k}_2)$$

$$\mathbf{k}_4 = \mathbf{f}(t_i + h, \mathbf{w}_i + h\mathbf{k}_3)$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4).$$

3 Testing your code

Now we will investigate some differential equations on which you will apply your code.

Problem 3.1. The Van der Pol equation is the following second order differential equation

$$y''(t) + \mu(y(t)^2 - 1)y' + y(t) = 0,$$

where μ is an arbitrary positive constant that we will set as $\mu = 2$ for this problem.

(a) Transform this second order differential equation into a first order system with two equations by introducing

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix}$$

and appropriately defining the system $\mathbf{u}' = \mathbf{f}(t, \mathbf{u})$.

(b) Solve your system on the interval $[0, 20]$ using your code (Euler + Runge-Kutta) for 3 different choices of initial values. You are free to choose an appropriate number of timesteps. Plot your solutions (e.g. plot y as the x-values vs y' as the y-values) and describe their qualitative behavior.

Problem 3.2. The position $(x(t), y(t))$ of a satellite or rocket in motion around the earth and moon is described by the following second order system of differential equations

$$x'' = x + 2y' - d\frac{x + c}{((x + c)^2 + y^2)^{\frac{3}{2}}} - c\frac{x - d}{((x - d)^2 + y^2)^{\frac{3}{2}}}$$

$$y'' = y - 2x' - d\frac{y}{((x + c)^2 + y^2)^{\frac{3}{2}}} - c\frac{y}{((x - d)^2 + y^2)^{\frac{3}{2}}}$$

where $c = 0.012277471$ and $d = 1 - c$. With the initial conditions

$$x(0) = 0.994, x'(0) = 0, y(0) = 0, y'(0) = -2.00158510637908252240537862224$$

there is a periodic solution with period $P = 17.06521656015796255889172062 49$.

(a) Transform this second order system of differential equations into a first order system with four equations by introducing

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}$$

and appropriately defining the system $\mathbf{u}' = \mathbf{f}(t, \mathbf{u})$.

(b) For each $n = 1, 2, \ldots, 100$, use your code for Euler's method to approximate $\mathbf{u}(P)$ using $10000n$ steps and record the error of your approximation as a function of $10000n$. To get more interpretable data, transform your results by taking logs to get log(error) as a function of $\log(10000n)$.

You should see that the relationship between log(error) and $\log(10000n)$ is approximately linear for relatively small values of n; find the slope of the best-fit line for this portion of your data. How do you interpret the slope of this line? Why is the data not linear for the entire domain?

(c) Repeat part (b) with your Runge-Kutta method. For this part, only use $n = 1, 2, \ldots, 100$ with $100n$ timesteps. Contrast the results your obtained in part (b) for Euler's method with the result you obtained in this part.

(d) The complexity of a method is often measured in terms of the number of function evaluations (i.e. the number of times the function $\mathbf{f}(t, \mathbf{y})$ needs to be evaluated). Based on this metric, the Runge-Kutta method of order 4 requires 4 times as many function evaluations as Euler's method. Does it seem like the accuracy achieved is 4 times the accuracy for Euler's method? Why or why not?

Problem 3.3. Recall the curvature differential equation from the practice section

$$y'' = -y(1 + y'^2)^{\frac{3}{2}}.$$

Using the same starting conditions $y(0) = 0$ and $y'(0) = \sqrt{3}$, use your Runge-Kutta fourth order implementation to approximate a solution $y(x)$ using 1000 steps on the interval $[0, 20]$ and plot the resulting curve you obtain.