Assignment #6

Due Wednesday, 29 March 2023, at the start of class

Please read textbook[1] sections 5.3–5.8 and 6.1–6.4.

Problem P27. **a)** In preparation for problem **P29** below, write two solvers

```
function [tt,zz] = feuler(f,eta,t0,tf,N)
function [tt,zz] = rk4(f,eta,t0,tf,N)
```

which implement schemes (5.19) and (5.33), respectively, to solve the ODE IVP in (5.1) and (5.2). The first input to these solvers is `function z = f(t,u)`.[2] The other inputs are a vector of initial values `eta` $= u(t_0)$, the initial time `t0`, the final time `tf`, and the number of equal-length steps (subintervals) `N`; the time step is $\Delta t = k = (t_f - t_0)/N$. Each solver outputs the entire trajectory, so `tt` is a 1D array of length $N+1$ starting with t_0 and ending with t_f. If $\eta \in \mathbb{R}^s$ then `zz` is a 2D array with s rows and $N+1$ columns; each column i gives the solution $u(t)$ at the ith time in `tt`.

b) Solve the following simple problem exactly:

$$x'' + x = 0, \qquad x(0) = 1, \quad x'(0) = 0.$$

Hint. You will need to find the exact solution, and also write this as a first order system for setting-up a numerical solution in part **c)**.

c) The problem in **b)**, for example on the interval $[t_0, t_f] = [0, 2]$, makes a good test case. Demonstrate that the final-time numerical error of each solver in **(a)** converges at the expected rate as the timestep $k \to 0$.

Hint. What is the expected rate is for each method? There is no need to compute local truncation errors yourself, but you must know their orders.

Problem P28. Compute the leading term in the local truncation error of the following methods. In particular, please follow the style of Example 5.9,[3] so that you know the coefficient in the leading order term.

a) the 2-step BDF method (5.25).

Hint. Expand around t_{n+1}, to get τ^{n+1}.

b) the trapezoidal method (5.22).

Hint. Expand around the "half-way" time $t^* = t_n + \frac{1}{2}k$, to get τ^*.

---

[1] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Diff. Eqns.*, SIAM Press 2007

[2] Use the MATLAB and `scipy.integrate.ode` variable order here not the book order "$f(u,t)$."

[3] For the multistep midpoint rule (5.23), Example 5.9 finds $\tau^n = \frac{1}{6}k^2 u'''(t_n) + O(k^4)$. The simpler statement $\tau^n = O(k^2)$ is also true, but I am asking for a bit more.

c) the explicit trapezoid method,

$$U^{n+1} = U^n + \frac{k}{2}\Big(f(U^n) + f\big(U^n + kf(U^n)\big)\Big).$$

Hint. Note how scheme (5.30), the explicit midpoint rule, is handled.

Problem P29. *This is a real application. Perhaps it will help you appreciate our abstract notation for ODE systems, vector data types in our languages, and higher-order explicit ODE schemes. This problem has an exact solution,*[4] *but it is not used here.*

Consider the problem of two massive bodies (particles) with masses m_1 and m_2. They are attracted by gravity only. They travel in a plane so their positions are given by vectors $\mathbf{x}_i(t) = (x_i(t), y_i(t))$ for $i = 1, 2$. Newton's second law and Newton's law of gravity combine to say:

$$\begin{aligned} m_1\mathbf{x}_1'' &= -Gm_1m_2\frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|^3} \\ m_2\mathbf{x}_2'' &= -Gm_1m_2\frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_1 - \mathbf{x}_2|^3} \end{aligned} \tag{1}$$

We will consider the Earth and the Moon in isolation as our example. Thus the constants are

$$\begin{aligned} m_1 &= 5.972 \times 10^{24}\,\text{kg}, \\ m_2 &= 7.348 \times 10^{22}\,\text{kg}, \\ G &= 6.674 \times 10^{-11}\,\text{m}^3\,\text{kg}^{-1}\,\text{s}^{-2}, \end{aligned}$$

and we measure t in seconds and x_i, y_i in meters. (*Though this will not be graded, please confirm that the units balance in equations* (1).)

a) By using notation $v_i = x_i', w_i = y_i'$ for $i = 1, 2$, write system (1) as a first-order ODE system of dimension $s = 8$, with solution column vector $u(t) \in \mathbb{R}^8$. Use the standard component ordering

$$\begin{aligned} u(t) &= \begin{bmatrix} x_1(t) & y_1(t) & x_2(t) & y_2(t) & v_1(t) & w_1(t) & v_2(t) & w_2(t) \end{bmatrix}^\top \\ &= \begin{bmatrix} u_1(t) & u_2(t) & u_3(t) & u_4(t) & u_5(t) & u_6(t) & u_7(t) & u_8(t) \end{bmatrix}^\top. \end{aligned}$$

That is, write system (1) in the form of (5.1) in the book:[5] $u'(t) = f(t, u(t))$. Then implement a single function

```
function z = fearthmoon(t,u)
```

which computes the right-hand-side function $f(t, u)$ of the ODE system.

b) For initial conditions which are vaguely like what they are in reality,[6] at least if you turned off all the gravity of other bodies and start the Earth stationary at the origin, suppose $t_0 = 0$ and $x_1(0) = 0, y_1(0) = 0, v_1(0) = 0, w_1(0) = 0$ and $x_2(0) =$

---

[4] See, for example: https://www.diva-portal.org/smash/get/diva2:630427/FULLTEXT01.pdf

[5] In fact the right side of this ODE system does not have explicit dependence on t. Please again use the MATLAB and `scipy.integrate.ode` variable ordering.

[6] I searched "earth moon distance meters" and "mean orbital velocity moon."

3.844×10^8 meters, $y_2(0) = 0$, $v_2(0) = 0$, $w_2(0) = 1.022 \times 10^3\,\mathrm{m\,s^{-1}}$. Use these initial conditions to generate approximate solutions with $t_f = 40$ days.[7]

Now use each of the solvers from problem **P27** with $N = 40$ and $N = 960$, i.e. daily and hourly time steps, respectively. Also use `ode45()`, or other black-box solver, using the default accuracy. That is, generate five numerical solutions.

Do not, of course, show me lots of numbers. Make basic plots of the computed trajectories, i.e. the x_i, y_i values. Describe in a few words what you see, and how these results relate to the local truncation error of the schemes in **P27**.

c) How long is a lunar month, if we used your computations in part **b)**?

[7]Convert to seconds!