Assignment #10

Due Thursday, 30 November at the start of class ⟵ *revised!*

Please read sections 6.1–6.5 of the textbook (J. Epperson, *An Intro. to Numerical Methods and Analysis*, 2nd edition).

Section 6.1, pages 334–335:

- **Exercise 1.** Do parts **(a),(c)** only.
- **Exercise 3.** Do parts **(a),(b)** only.
- **Exercise 4.** Do parts **(a),(b)** only.

Section 6.2, page 338:

- **Exercise 1.** Do parts **(c),(d)** only. ("Hand calculator" includes "the MATLAB command line," in my opinion.)

Section 6.3, page 342:

- **Exercise 1.**

Section 6.4, pages 357–359:

- **Exercise 4.** Do parts **(a),(b)** only.
- **Exercise 7.** Do parts **(c),(d)** only. Do problem **P14** before this exercise, so that you can use `predcorr.m` to do the computations. Generate a table like Table 6.3 on page 351, using $h^{-1} = 2, 4, \dots, 1024$ and where $E(h) = \max_{t_k \le 1} |y(t_k) - y_k|$. Regarding "theoretical accuracy", check whether the error goes down by a factor of 4 each time h is halved.

P14. Write a code which implements the trapezoid rule predictor-corrector method:

```
function [t,Y] = predcorr(f,t0,y0,tf,N)
```

It does N steps using formulas (6.32) and (6.33). The ODE IVP is in the form

$$y' = f(t, y), \qquad y(t_0) = y_0.$$

Assume you are seeking $y(t)$ for $t \in [t_0, t_f]$, where t_f is the "final time." Note $y(t) \in \mathbb{R}$; do not worry about systems of ODEs.

The input `f` is a function of two variables. For example, if the ODE is $y' = (t+1)y^2$ then we define `f = @(t,y) (t+1).*y.^2`.

The function `predcorr()` returns arrays `t` $= \{t_0, t_1, t_2, \dots, t_N\} = \{t_0, t_0 + h, t_0 + 2h, \dots, t_f\}$, with step size $h = (t_f - t_0)/N$, and the approximate solution values `Y` $= \{y_0, y_1, y_2, \dots, y_N\}$. Note that plotting the solution is then easy: `plot(t,Y)`.

Test your code by doing **Exercise 7** in Section 6.4, above. No need to test here.

You can compare to an already-posted Euler method code, `eulermethod.m`*. It has a slightly-different signature,* `function [t,Y] = eulermethod(f,t0,y0,N,h)`*, but the inputs are equivalent because $t_f = t_0 + Nh$ and $h = (t_f - t_0)/N$.*