HOMEWORK 5 (DUE WED. FEB. 26)

Reading: Read the `ivp_example` code and comments, which has some notes on implementing ODE methods.

- Notable suggestion: The excerpted parts of Ascher & Petzold (see Piazza).
- See the course schedule for recommended book sections.

Code to turn in:

- The code for the two ODE methods (explicit trapezoidal, AB2).

1. Problems

Q1. Derive conditions such that the multi-step method

$$\frac{1}{h}\sum_{j=0}^{m} a_j u_{n-j} = \sum_{j=0}^{m} b_j f_{n-j} \tag{1}$$

is cosistent with order at least 1 (for the ODE $y' = f(t,y)$).

Q2. Derive the Adams-Bashforth method using three previous points,

$$\frac{u_n - u_{n-1}}{h} = af_{n-1} + bf_{n-2} + cf_{n-3} + \text{LTE}$$

using a Taylor series (The answer is in the lecture notes).

Q3 (the cost of making a method explicit). Recall that the trapezoidal method is

$$u_{n+1} = u_n + \frac{h}{2}(f_{n+1} + f_n). \tag{T}$$

a) Show that the trapezoidal method is A-stable. [AC] What does the right half of the stability region tell you about the method?

b) This method is implicit. One may be tempted to get around the implicit part by approximating f_{n+1} by some other method. Show that the explicit method[1]

$$\begin{aligned} \tilde{u}_{n+1} &= u_n + hf_n, \\ u_{n+1} &= u_n + \frac{h}{2}(f(t_{n+1}, \tilde{u}_{n+1}) + f_n) \end{aligned} \tag{T2}$$

is consistent with the same order as the trapezoidal method (T). Derive an expression for the stability region. How small must h be (for both methods) when solving the IVP

$$y' = -2t^2 y + 2t - \frac{1}{t^2}, \quad y(1) = 1, \quad t \in [1, 10]$$

to ensure stability?

[1] An example of a **Runge-Kutta** method, to be covered; sometimes called the explicit trapezoidal method.

Q4 (Checking the theory). Implement the method (T2) in the form `odet2(f,I,y0,n)` where `I = [a,b]`. The code should solve the IVP

$$y' = f(t, y),\ y(a) = y_0, \quad t \in [a, b].$$

You can use h instead of n in the input if you prefer.

a) Make a log-log plot of the error for the problem in Q3b, showing that it is second order and that your answer in Q3b makes sense. (exact solution: $y(t) = 1/t$).

b) Recall that if $A(h) \approx L + Ch^p$ we can estimate p using approximations at $h, h/2, h/4$ (see HW3, Q1b). The issue here is that the solution is a grid function with N values, not a scalar. One cheap way around the issue is to test the error at a fixed time. Make a table of the error at $t = 5$ with $N = 2, 4, 8, \cdots 2^m$ points and the estimate for p (so the column for p should have $m - 2$ elements). Show that the data indicates the method is second order.

c) [AC] Suppose you wanted to check the error as in (b), but account for the whole solution (not just testing one point). What is a good way to generalize (b)?

Q5. Implement the AB2 (Adams Bashforth, two step) method

$$u_n = u_{n-1} + h\left(\frac{3}{2} f_{n-1} - \frac{1}{2} f_{n-2}\right)$$

using Euler's method as a starter. The method should solve the scalar ODE

$$y' = f(t, y), \quad y(a) = y_0, \qquad t \in [a, b].$$

a) Test your method on

$$y' = \sin(y), \quad y(0) = \pi/2$$

(exact solution: $y(t) = 2\cot^{-1}(e^{-t})$) and verify the order by plotting the error.

b) Use your code to get a plausible value of the interval of absolute stability $(-b, 0)$ for the method (the point here is that you can use the numerical code to get the answer to a theory question before doing the analysis).

Q6 [AC]. Implement the system version of the method (same formula, but with vectors) and use it to solve[2]

$$y'' = xy, \quad y(0) = 0.355, \quad y'(0) = -0.26, \qquad x \in [-10, 10]$$

Make a plot of the solution $y(x)$ in the given interval (note that the interval is two-sided here, not just $x > 0$!). (You don't need to also plot the other component).

Note: The next homework will have more on implementing systems, so Q5c is a good head start but left optional for now if you don't have time. You can also turn in Q6 as part of the next homework.

[2] This ODE is the **Airy equation**, an ODE that shows up often in applied problems; with slightly different initial conditions, the solution is the **Airy function** Ai(x). I've included rounded values instead of the exact ICs for convenience.