CAAM 453: Numerical Analysis I

Problem Set 6: ODE Methods
Due: Monday, November 13, 2018

Note: Turn in all MATLAB *code that you have written and turn in all output generated by your* MATLAB *functions/scripts.* MATLAB *functions/scripts must be commented, output must be formatted nicely, and plots must be labeled. You may freely use any* MATLAB *functions from this year's CAAM 453 website. If you do so, you do not need to include their code in your writeup unless you have modified them.*

Problem 1: LMM Accuracy and Stability (10 points)

Consider the following two linear multistep methods[†]:

$$x_{k+2} = 3x_k - 2x_{k+1} + f(x_k, t_k) + 3f(x_{k+1}, t_{k+1}); \tag{a}$$

$$x_{k+2} = \frac{1}{2}\left[x_k + x_{k+1}\right] + 2f(x_{k+1}, t_{k+1}). \tag{b}$$

For each method, decide if it is (i) zero-stable; (ii) consistent; and if it is consistent, (iii) find its order of accuracy.

Problem 2: Accuracy for Adams Methods (5 points)

By counting degrees of freedom, what is the maximum order of accuracy you would expect to achieve with an m-step explicit method where $\alpha_m = 1$, $\alpha_{m-1} = -1$? For an implicit method?

Problems 3 and 4 are on the following pages.

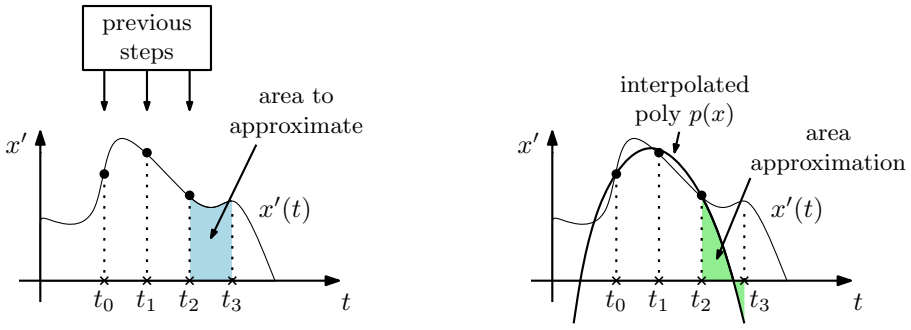
[†]Note method (b) is a slight modification of the leapfrog method, in which $x_{k+2} = x_k + 2f(x_{k+1}, t_{k+1})$. In (b) x_k is replaced by the average $(x_k + x_{k+1})/2$.

Problem 3: Return of Polynomial Interpolation (35 points)

Consider the following idea for an explicit linear n-step method for solving $x' = f(x,t)$. By the fundamental theorem of calculus,

$$x(t_{k+n}) = x(t_{k+n-1}) + \int_{t_{k+n-1}}^{t_{k+n}} x'(t)\,dt \qquad (*)$$

and of course $x'(t) = f(x(t),t)$. So the problem is to approximate $x'(t)$ over the interval $[t_{k+n-1}, t_{k+n}]$. Now given $x(t_k), \ldots, x(t_{k+n-1})$, we can interpolate the corresponding values $x'(t_k) = f(x(t_k),t_k), \ldots, x'(t_{k+n-1}) = f(x(t_{k+n-1}),t_{k+n-1})$ with a degree-$(n-1)$ polynomial $p(x)$, and then replace the integral in equation $(*)$ with $\int_{t_{k+n-1}}^{t_{k+n}} p(x)$:



So, this leads to an n-step method. Given $x_k, \ldots, x_{k+n-1}$, we first interpolate the derivatives $f(x_k,t_k), \ldots, f(x_{k+n-1},t_{k+n-1})$ with a degree-$(n-1)$ polynomial $p(x)$. Then, let

$$x_{k+n} = x_{k+n-1} + \int_{t_{k+n-1}}^{t_{k+n}} p(x)\,dx. \qquad (**)$$

For short, write $f_i = f(x_i,t_i)$.

1. (a) Let $n = 3$. Compute $p(x)$ and the integral $\int_{t_{k+2}}^{t_{k+3}} p(x)\,dx$ in terms of f_k, f_{k+1}, and f_{k+2}. What are the coefficients α and β for this method? Which method from class have you derived? *Hint: you may assume that $t_k = 0$ (so $t_{k+1} = h$, and $t_{k+2} = 2h$) since shifting t doesn't affect the value of the integral.*

2. (b) The same idea can be used to develop an implicit method. For this, to get x_{k+n}, we interpolate $f(x_k,t_k), \ldots, f(x_{k+n},t_{k+n})$ with a degree-n polynomial $p(x)$ (remember, x_{k+n} is unknown— we will eventually have to solve for it). Then we use $(**)$ to define x_{k+n} in terms of x_{k+n-1}.

   Let $n = 2$, and compute $p(x)$ and $\int_{t_{k+1}}^{t_{k+2}} p(x)\,dx$ in terms of $f_k, \ldots, f_{k+2}$. This is the 2-step *Adams–Moulton* method.

Problem 4: LMM Computation (50 points, pledged)

In this problem, you will write a MATLAB function to approximately solve an IVP using a given linear multistep method. Consider the following initial value problem:

$$\text{Find } x(t) \text{ for } t \in [t_0, t_1] \text{ such that } \begin{cases} x' = f(x,t), \\ x(t_0) = x_0. \end{cases}$$

Inputs: Your function will be given the initial conditions x_0, t_0, the final time t_1, the right-hand side $f(x,t)$ as a function handle, the method's coefficients $\alpha_0, \ldots, \alpha_n$ and $\beta_0, \ldots, \beta_{n-1}$ (as vectors), and the stepsize h.

Outputs: It should return $t_0, \ldots, t_r$, an array of time steps, and $x_0, \ldots, x_r$, the approximations to $x(t)$ at those time steps.

Initialization: To construct the first $n-1$ steps $x_1, \ldots, x_{n-1}$, use the RK4 multistage method (code on class website).

After writing your code, do the following:

(a) To check your code, apply 2-step Adams–Bashforth (AB2) to the problem $x' = 2x$, $[t_0, t_1] = [0, 1]$, $x_0 = 1$, with the step sizes $h = 0.2$, 0.1, 0.05. Create a plot of the three numerical solutions and the exact solution together.

(b) For a time-dependent problem, consider $x' = x - t$, $[t_0, t_1] = [0, 2]$, $x_0 = 1/2$; the exact solution is $x(t) = t + 1 - \frac{1}{2}e^t$. Apply the leapfrog method ($x_{k+2} = x_k + 2f(x_{k+1}, t_{k+1})$) with time steps $h = 0.4$, 0.2, 0.1. Again, create a plot with the numerical and exact solutions.

(c) An interesting problem is the logistic equation $x' = ax(1-x)$, with $a = 5$, $[t_0, t_1] = [-1, 3]$, $x_0 = 0.3$. The exact solution is $x(t) = 1 - \frac{1}{1+Ke^{at}}$, with $K = (\frac{x_0}{1-x_0})e^{-at_0}$.

(1) Apply (forward) Euler and 4-step Adams–Bashforth (AB4) with $h = 0.1$ and plot both solutions together with the exact solution. In which portions of the interval $[-1, 3]$ is each method most accurate?

(2) This ODE exhibits *stiffness*: the time step for AB4 is too large, leading to divergence. Apply AB4 with $h = 0.04, 0.08, 0.09, 0.1$, and plot the solutions together. How sensitive is the value of the numerical solution at $t = 3$ to changes in h?

(3) Now apply AB2 and RK4 with $h = 0.1$, and plot the results of each. You may use the RK4 code from the website.

(4) Finally, with $h = 0.05$, find the error between the exact and approximate solutions at $t = 3$ for each of the methods: Euler, AB2, and AB4. Is this surprising?

Compute the same error for RK4, but for fairness, use $h = 0.2$, since as a four-stage method it performs four evaluations of $f(x,t)$ per time step, whereas the other methods perform just one function evaluation per step.

(d) In this part, let's examine zero-stability for the order-3, 2-step explicit method from class: $x_{k+2} = 5x_k - 4x_{k+1} + 2f(x_k, t_k) + 4f(x_{k+1}, t_{k+1})$. We will apply this method to the problem $x' = 0$, $[t_0, t_1] = [0, 2]$, $x_0 = 1$. Modify your code so it uses 1 and $1 + \epsilon$ (ϵ = machine epsilon) as starting values x_0, x_1, instead of using RK4. Plot the results with time steps $h = 0.2$ and $h = 0.05$ (make sure to choose reasonable limits). What is happening?