AM225: Assignment 2 (due 5 PM, March 4)

Part I: ODE solution methods

*Complete at least **one out of two** problems in this section. If you submit answers for both, your grade will be calculated using the best score. Note: question 2 is harder than question 1.*

1. **Adaptive integration with a First Same As Last (FSAL) scheme.** Write an adaptive Runge–Kutta integration scheme to solve an arbitrary ODE problem $y' = f(x, y)$ for $y(x) \in \mathbb{R}^n$, using the five step FSAL scheme described in the lectures. Your code should have the following properties:

   - Use the fourth-order solution y_1 to step forward, and use the third-order $\hat{y}_1$ for step size selection.
   - Use the step size selection procedure described in the slides. The parameters *fac*, *facmax*, and *facmin* from the slide can be used. For the tolerance, you can assume *Atol* and *Rtol* are the same for all components, rather than being specified on a per-component basis. Use an initial step size of $h = 0.01$.
   - Make your program count the number of function evaluations. Your program should be parsimonious in the number of function evaluations it performs, by reusing k_5 from a succesful step to be k_1 of the next step, and by retaining k_1 when a step is rejected.

   Once your method is working, complete the following two test problems.

   (a) Test your program on the Brusselator test problem using $Atol = Rtol = \lambda$. By trying a range of λ from 10^{-3} to 10^{-13} make a precision–work plot as in the lectures. In the program files, you will find the corresponding precision–work data for the four low-order methods considered in the lectures, and you may overlay these results on your plot to compare them.

   (b) Extend your code so that it does third-order dense output with Hermite interpolation at regular intervals. Test your code using the two-component system
   $$y_1' = -xy_2, \qquad y_2' = xy_1 \tag{1}$$
   with initial conditions $y_1(0) = 1, y_2(0) = 0$. This problem has the exact solution $y_1^{\text{exact}}(x) = \cos\frac{x^2}{2}, y_2^{\text{exact}} = \sin\frac{x^2}{2}$. Simulate to $x = 8$ using $\lambda = 3 \times 10^{-3}$, saving dense output at intervals of $\frac{8}{1200}$. Plot the numerically computed solutions y_1^{num} and y_2^{num}, showing the integration steps as points, and the dense output as lines. Make a second plot showing $y_1^{\text{num}} - y_1^{\text{exact}}$ and $y_2^{\text{num}} - y_2^{\text{exact}}$.

   (c) **Optional.** The method is not sensitive to the initial step size choice of $h = 0.01$, since the adaptive procedure will automatically adjust it. However, Hairer *et al.* describe an algorithm for estimating the initial timestep. Extend your method to implement this.

2. **A high-order adaptive integrator using Richardson extrapolation.** Write an adaptive Runge–Kutta integration scheme by applying Richardson extrapolation to the fifth-order Cash–Karp scheme[1] in the lectures, thereby obtaining a sixth-order method. Starting from y_0, let y_1 and

[1]For the purposes of this question, you can ignore the lower order Cash–Karp formulae, since here the aim is to use Richardson extrapolation for step size selection.

y_2 be Cash–Karp steps with size $\frac{h}{2}$, and w be a Cash–Karp step of size h. Define the sixth-order solutions

$$\hat{y}_1 = y_1 + \frac{y_2 - w}{(2^p - 1)2}, \qquad \hat{y}_2 = y_2 + \frac{y_2 - w}{2^p - 1}. \tag{2}$$

Your program should use the same step size selection procedure as from Question 1.[2] It should count the number of function evaluations and be as parsimonious as possible. Use $y_2 - \hat{y}_2$ for step size selection, and use $\hat{y}_2$ to advance forward in x.

(a) Repeat Question 1(a) for this method.

(b) Extend your code so that it computes dense output at regular intervals, based on quintic polynomial interpolation using y_0, $f(x_0, y_0)$, $\hat{y}_1$, $f(x_0 + h, y_1)$, $\hat{y}_2$, and $f(x_0 + 2h, \hat{y}_2)$.[3] Repeat the two-component test from Question 1(b).

Part II: ODE applications and analysis

Complete at least ***three out of five*** *problems in this section. If you submit answers for more, your grade will be calculated using the three best scores.*

3. **Order condition trees.** Write a program to enumerate all trees of a given order. Provide a list of the number of trees up to order 15.[4] Extend your program so that it can visualize the trees in some format of your choice, and use it to show all trees of order 7.

4. **Error analysis of a Richardson extrapolation scheme.**

   (a) Show that Richardson extrapolation applied to the second-order Ralston method can be reformulated as a five-step, third-order Runge–Kutta method, and find its Butcher tableau.

   (b) The third-order Heun method has Butcher tableau

$$\begin{array}{c|ccc} 0 & & & \\ 1/3 & 1/3 & & \\ 2/3 & 0 & 2/3 & \\ \hline & 1/4 & 0 & 3/4 \end{array}$$

   For both the Heun method, and your method from part (a), determine the error coefficients $e(t)$ for all trees t of order 4, reporting your answers as rational numbers.

   (c) Show that one of the methods has universally smaller error magnitudes $|e(t)|$ than the other. Once the difference in the number of function evaluations is taken into account, will that method be better for practical calculations?

[2] Since Richardson extrapolation requires taking two timesteps of size $h/2$, you may encounter NaNs for a large choice of h. You code should reject that step and try again with $h \times facmin$.

[3] Note that the derivative $f(x_0 + h, y_1)$ can be used. It is not necessary to evaluate $f(x_0 + h, \hat{y}_1)$.

[4] In the lecture slides you will find the number of trees up to order 10, which you can use to check your solutions.

5. **A generalized Kuramoto model.**[5] A recent paper by O'Keeffe *et al.*[6] explores a model for swarming and synchronization behavior. In the model, we consider N agents with positions $\mathbf{x}_i(t)$ and internal phases $\theta_i(t)$, which move according to the differential equations

$$\dot{\mathbf{x}}_i = \mathbf{v}_i + \frac{1}{N}\left[\sum_{j\neq i}^{N} \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|}(A + J\cos(\theta_j - \theta_i)) - B\frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|^2}\right], \tag{3}$$

$$\dot{\theta}_i = \omega_i + \frac{K}{N}\sum_{j\neq i}^{N} \frac{\sin(\theta_j - \theta_i)}{|\mathbf{x}_j - \mathbf{x}_i|} \tag{4}$$

where J and K are constants, and $\mathbf{v}_i$ and ω_i can be individually controlled for each agent. By rescaling time and space, we can restrict attention to the case when $A = B = 1$.

(a) By making use of your favorite adaptive integrator with dense output[7] solve Eqs. 3 & 4 using $N = 1250$ agents. Set $\mathbf{v}_i = \omega_i = 0$ for all agents. Simulate from $t = 0$ to $t = 200$, and use dense output to save the positions at n equally-spaced intervals, where $N \geq 401$. Use $Atol = Rtol = 10^{-6}$ in your adaptive integration routine.

Use initial conditions of random positions in the unit disk, $\|\mathbf{x}\| \leq 1$, and random phases over $[0, 2\pi)$. Visualize the agents as dots that are colored according to their phase. A suggested color palette is

$$(R, G, B) = (f(\theta), f(\theta - {}^{2\pi}/_{3}), f(\theta + {}^{2\pi}/_{3}))$$

where $f(\theta) = 0.45(1 + \cos\theta)$. Simulate the model with the following parameters:

i. $J = 0.5, K = 0.5$,
ii. $J = 0.3, K = -0.2$,
iii. $J = 1, K = -0.2$.

For each case, state the total number of timesteps taken. Either

- include snapshots after $t = 10, 20, 50, 200$,
- or make a movie of the snapshots.

(b) Simulate at least one possible variation. Examples include: (i) changing $\mathbf{v}_i$ and ω_i, (ii) simulating two systems to steady state and then making a new initial condition with both superimposed, and (iii) implementing the method in 3D.[8]

(c) **Optional.** Extend your code to calculate right hand sides of Eqs. 3 & 4 using OpenMP. Note that the influence of actor A on actor B is equal and opposite to the influence of actor B on actor A. Structure your code so that it only considers each pair once.

[5] This question was suggested by Nick Boffi (`boffi@g.harvard.edu`), and could be the basis for a final project.

[6] K. P. O'Keeffe, H. Hong, and S. H. Strogatz, *Oscillators that sync and swarm*, Nat. Commun. **8**, 1504 (2017). doi:10.1038/s41467-017-01190-3

[7] You could use your code from Q1 or Q2. You could use the DOP853 implementation found in the course example codes.

[8] See O'Keeffe *et al.* to see how they alter the strengths of the terms in 3D.

6. **Symplectic integration for galactic dynamics.** The following fifth-order IRK method due to Geng is symplectic, meaning that it exactly preserves the Hamiltonian $H(p,q)$ for a Hamiltonian system:

$$
\begin{array}{c|ccc}
\frac{4-\sqrt{6}}{10} & \frac{16-\sqrt{6}}{72} & \frac{328-167\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{450} \\
\frac{4+\sqrt{6}}{10} & \frac{328+167\sqrt{6}}{1800} & \frac{16+\sqrt{6}}{72} & \frac{-2-3\sqrt{6}}{450} \\
1 & \frac{85-10\sqrt{6}}{180} & \frac{85+10\sqrt{6}}{180} & \frac{1}{18} \\
\hline
 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9}
\end{array}
$$

A simple model for the movement of star in a galaxy is described by the Hamiltonian

$$H(\mathbf{p},\mathbf{q}) = \frac{p_1^2+p_2^2+p_3^2}{2} + \Omega(p_1q_2 - p_2q_1) + V(\mathbf{q}), \tag{5}$$

where the star's position is $\mathbf{q}(t) = (q_1(t), q_2(t), q_3(t))$ and its momentum is $\mathbf{p}(t) = (p_1(t), p_2(t), p_3(t))$. Here, Ω is the galaxy's velocity, and V is the gravitational potential, which is approximated as

$$V(\mathbf{q}) = A\log\left(C + \frac{q_1^2}{a^2} + \frac{q_2^2}{b^2} + \frac{q_3^2}{c^2}\right). \tag{6}$$

We use non-dimensionalized parameters $a = 1.25$, $b = 1$, $c = 0.75$, $A = 1$, $C = 1$, $\Omega = 0.25$. The Hamiltonian differential equation system is given by

$$\dot{p}_i = -\frac{\partial H}{\partial q_i}, \qquad \dot{q}_i = \frac{\partial H}{\partial p_i} \tag{7}$$

for $i = 1,2,3$. Initial conditions are given by $q_2(0) = q_3(0) = p_1(0) = 0$, $q_1(0) = 2.5$, and $p_3(0) = 0.2$. The remaining momentum coordinate is chosen to be the larger of the two roots that yields $H = 2$.

   (a) Implement Geng's method, and test it on an ODE of your choice to verify that it is fifth-order accurate.[9] Make a convergence plot demonstrating fifth-order accuracy.

   (b) Simulate the galaxy ODE system up to $t = 2000$ using a step size of $1/20$ and make a 3D plot of the trajectory. Plot the Hamiltonian up to $t = 2000$.

   (c) Simulate up $t = 10^5$ and make a Poincaré map by tracking all intersections with the half-plane $q_1 > 0$, $q_2 = 0$, where $\dot{q}_2 > 0$. You can find the intersection points by approximating the trajectory as a linear segments between successive timesteps.

7. **Integrating ODEs with discontinuities.** Consider the two-component ODE system for functions $x(t)$ and $y(t)$ given by

$$\frac{dx}{dt} = \begin{cases} 0 & \text{if } |x| \geq |y|, \\ -y & \text{if } |x| < |y|, \end{cases} \tag{8}$$

[9] You do not need to test the method on a symplectic ODE system.

and

$$\frac{dy}{dt} = \begin{cases} x & \text{if } |x| \geq |y|, \\ 0 & \text{if } |x| < |y|. \end{cases} \tag{9}$$

Use the initial condition $x(0) = 1$ and $y(0) = 0$.

(a) Calculate the analytical solutions of $x(t)$ and $y(t)$. Show that they are periodic, and find the period.

(b) Simulate the ODE system in Eqs. 8 and 9 to $t = 48 + e^{-1}$, using the classic fixed-step fourth-order Runge–Kutta (RK4) method. Make a work–precision plot using a range of total step numbers from 10^3 to 10^7. Your calculation of precision should be based on the difference between the numerical solution and the exact solution from part (a). Is the convergence data consistent with RK4 being fourth-order accurate? If not, why not?

(c) Repeat part (b) with your favorite adaptive integrator, using $Rtol = 0$, and a range of absolute tolerances of $Atol \in [10^{-12}, 10^{-2}]$. Overlay the results on the work–precision plot from part (b).

(d) **Optional.** Consider the variant ODE system

$$\frac{dx}{dt} = \begin{cases} 0 & \text{if } |x| \geq |y|, \\ -\operatorname{sign}(y) & \text{if } |x| < |y|, \end{cases} \tag{10}$$

and

$$\frac{dy}{dt} = \begin{cases} \operatorname{sign}(x) & \text{if } |x| \geq |y|, \\ 0 & \text{if } |x| < |y|, \end{cases} \tag{11}$$

with initial conditions $x(0) = 1$ and $y(0) = 0$. Here

$$\operatorname{sign}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases} \tag{12}$$

Find the exact solution for this ODE system. Repeat the convergence analysis from parts (b) and (c), and compare the work–precision plots.