MA50174 Advanced Numerical Methods

Assignment 3 Initial value problems

- **Set:** Monday 21st November
- **To be handed in:** Friday 2nd December, by 12.00 to Maths Department Reception Desk. You need to sign your work personally together with a signed coursework cover sheet.
- **Handing in your work.** Please write your MATLAB codes to a CD and hand them in together with the written part of your work. There will be no marks for typesetting your work (using, for example LaTeX) but it is your responsibility to make sure your work is well-presented and readable.

All work which you hand in should be your own. Cheating is a serious offence and will be dealt with under the University disciplinary procedures.

This assignment should typically take about 15 hours. If you spend significantly more than this on the assignment then you may disadvantage yourself with regard to other courses.

You have a limit of 10 pages (excluding Matlab codes).

MATLAB uses several algorithms for finding the numerical solution of initial value differential equation problems. In this assignment we will look at both the inbuilt MATLAB functions and also some other methods, for both stiff and non-stiff problems.
We start the assignment by looking at the performance of some integrators on non-stiff initial value ordinary differential equations. These are problems for which all of the components evolve simultaneously on comparable time-scales. Non-stiff problems are often solved by using *explicit* methods usually with some error control.

1. Consider the following system of non-linear first order differential equations $\dot{\mathbf{u}} = \mathbf{f}(t, \mathbf{u})$, $\mathbf{u} = [x, y]^T$ given by

$$\begin{aligned} \dot{x}(t) &= \alpha x(t) - \beta x(t) y(t) \\ \dot{y}(t) &= \delta x(t) y(t) - \gamma y(t), \end{aligned}$$

   with initial conditions $x(0) = 2$ and $y(0) = 1$ and parameters α, β, γ and δ. These equations are a simple case of the so-called *Lotka-Volterra* equations, also known as predator-prey equations. They are frequently used to describe the dynamics of biological systems in which two species interact, one a predator and one its prey. $x(t)$ is the number of prey and $y(t)$ represents the number of predators in a population at time t.

   (a) Implement the function $\mathbf{f}$ in a function file `f.m` for $\alpha = \beta = 10$ and $\gamma = \delta = 1$. Then, write a MATLAB script `method1.m` which implements the *forward Euler method* for this problem over the time interval $[0, 5]$ where the step size h should be provided by the user. [1]

   (b) Write a MATLAB script `method2.m` which implements the following method

$$\mathbf{U}^{n+1} = \mathbf{U}^n + h\mathbf{f}(t^n + \frac{h}{2}, \mathbf{U}^n + \frac{h}{2}\mathbf{f}(t^n, \mathbf{U}^n)), \quad \mathbf{U}^0 = \mathbf{u}(0) \tag{1}$$

   for this problem over the time interval $[0, 5]$ where the step size h should be provided by the user and plot the approximations to the solution $x(t)$ and $y(t)$ in the same figure. [1]

(c) Investigate the convergence behaviour of method (1) and the forward Euler method: Use step sizes 2^{-j}, $j = 4, \ldots, 10$ and calculate the norm of the error at time $t = 5$. Hence, by finding the convergence rates of both methods, compare their performance and explain your results theoretically by examining the truncation error. [2]

(d) Change the intitial conditions to $x(0) = y(0) = 1$, run Euler's method and repeat the calculation of the error norms. What do you observe? Explain your findings theoretically.[1]

2. We will investigate the problem of the angle of swing θ of a *simple pendulum* of unit length, which in a unit gravitational field, without damping, satisfies the differential equation

$$\frac{d^2\theta}{dt^2} + \sin(\theta) = 0, \quad \theta(0) = \alpha < \pi, \quad \frac{d\theta}{dt}(0) = 0. \tag{2}$$

(a) Show (analytically) that the energy (or Hamiltonian) $H(t)$ defined by

$$H(t) = \frac{1}{2}\left(\frac{d\theta}{dt}\right)^2 - \cos(\theta), \tag{3}$$

is a constant of the motion, and evaluate this constant. [1]

(b) The pendulum equation (2) is an example of a Hamiltonian system of the special form:

$$H = \frac{1}{2}p^2 + V(q), \qquad \frac{dq}{dt} = \frac{\partial H}{\partial p}, \qquad \frac{dp}{dt} = -\frac{\partial H}{\partial q}$$

Express the equation with initial conditions given in (2) using a vector differential equation and identify p and q. This can be solved by using a *symplectic integrator* with a time step $h \ll 1$. Suppose that q^n, p^n and H^n are the numerical approximations of $q(t), p(t)$ and $H(t)$ at the time $t = (n-1)h$. Two explicit symplectic integrators that can be used *when the Hamiltonian has the special form* are:

$$\text{symplectic-Euler:} \quad q^{n+1} = q^n + hp^n, \quad p^{n+1} = p^n - h\sin(q^{n+1}),$$

$$\text{Störmer-Verlet:} \quad q^* = q^n + \frac{h}{2}p^n, \quad p^{n+1} = p^n - h\sin(q^*), \quad q^{n+1} = q^* + \frac{h}{2}p^{n+1}.$$

Write a MATLAB script `pendulum.m` which implements (on request) either the *forward Euler* method, the *symplectic Euler* method or the *Störmer-Verlet* method for solving the pendulum equation (2) with a requested h and α over a time interval $[0, T]$. The code should generate sequences q^n, p^n and H^n. [1]

(c) Apply the code in the case of $T = 100$, $h = 0.1$ and $\alpha = 0.25$, comparing the results of the forward Euler, symplectic Euler and Störmer-Verlet method by plotting the points (q^n, p^n). Which do you think is the better method for this problem and why? [1]

(d) Suppose that α and hence q and p are initially *small*. By using a suitable approximation of (2), write a *linear* system in Hamiltonian form using the variables $\tilde{p}, \tilde{q}, \tilde{H}$. Then find a matrix A so that if $\mathbf{U}^n = (\tilde{q}^n, \tilde{p}^n)^T$ are the solutions of the forward Euler method applied to this system, then

$$\mathbf{U}^{n+1} = A\mathbf{U}^n.$$

Hence, or otherwise, show that in this case

$$\left((\tilde{q}^{n+1})^2 + (\tilde{p}^{n+1})^2\right) = \left(1 + h^2\right)\left((\tilde{q}^n)^2 + (\tilde{p}^n)^2\right).$$

[1]

(e) Find a corresponding matrix A for the symplectic Euler method applied to the linear system. Hence show that if $(\tilde{q}^n, \tilde{p}^n)$ is a small solution of the symplectic Euler method then there is a constant C so that

$$(\tilde{q}^n)^2 + h\tilde{q}^n\tilde{p}^n + (\tilde{p}^n)^2 = C.$$

Using this result and that in (d) give a brief explanation of the results of your computations in (c). [2]

3. We now look at *stiff* differential equations. When solving a stiff equation a numerical method has a step size which is restricted *by considerations of stability rather than accuracy.* MATLAB uses the functions `ode23t` and `ode15s` to solve stiff problems. Both of these usually involve solving nonlinear equations in order to timestep. Stiff equations are very common in applications; examples arise in the study of chemical reactions. We start with some theoretical investigations before applying the results we have derived to a problem arising in chemical engineering.

   Suppose that $\mathbf{u}$ satisfies the differential equation

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(u) \equiv \Lambda\mathbf{u}, \qquad \Lambda = \begin{pmatrix} -100 & 1 \\ 0 & -1 \end{pmatrix} \quad \mathbf{u}(0) = (2,2)^T.$$

   (a) Find the solution $\mathbf{u}(t)$ to this system. [1]

   (b) Show that if the forward Euler method is used to solve this differential equation, with $\mathbf{U}^n$ approximating $\mathbf{u}((n-1)h)$ then

$$\mathbf{U}^n = (1 + h\Lambda)^{n-1}\,\mathbf{U}^1.$$

   Let $A_h = I + h\Lambda$. Show that $\mathbf{U}^n$ will grow without bound if $h > 2/100$. This method is *unstable* if $h > 2/100 = 0.02$ and this imposes a restriction on the step size we can use. [1]

   (c) The *absolute truncation error* $\mathcal{E}$ that is made at each step of the forward Euler method between t and $t + h$ is given by

$$\mathcal{E} = \frac{h^2}{2}|\mathbf{u}''(\xi)|, \quad \xi \in [t, t+h].$$

   The relative error R is given by $R = \mathcal{E}/|\mathbf{u}|$. If we want a relative error of 0.5×10^{-2} at each step when calculating the solution, show that we could *in principle* use a step size of $h = 0.1$ in the latter part of the calculation. What step size do we in principle need to use in the initial stages of the calculation? [1]

   This shows that in the initial stages of the calculation (when $\mathbf{u}$ is varying rapidly), the step size is restricted by considerations of *accuracy* and in the latter stages by consideration of *stability.* To avoid the growth of small errors in the latter stages of the calculation the method is working too hard. This is the hallmark of using a non-stiff method on a stiff problem. The function `ode45` behaves in a very similar manner.

4. The *trapezoidal rule* is widely used to solve stiff ordinary differential equations. It combines stability with accuracy and ease of use. It is also a symmetric method which makes it especially suitable for solving differential equations which remain unchanged when time is reversed. MATLAB has a trapezoidal rule method `ode23t`. The trapezoidal rule is defined by

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \frac{h}{2}\left(\mathbf{f}(U^n) + \mathbf{f}(U^{n+1})\right).$$

   If $\mathbf{f}(u)$ is a nonlinear function, then, at each time step, a nonlinear system must be solved.

   (a) Show that for the problem in Question 3 we have

$$\mathbf{U}^{n+1} = B_h\mathbf{U}^n,$$

   for some matrix B_h which you should identify. Hence show that this method is *stable* for *all* (positive) h i.e. $\mathbf{U}_n$ does not grow without bound. [1]

   (b) The absolute truncation error at each stage of the trapezoidal rule is given by

$$\mathcal{E} = \frac{h^3}{12}|\mathbf{u}'''(\xi)|, \quad \xi \in [t, t+h].$$

   Find choices of step size h for the initial and final stages of the calculation so that the relative error $R = \mathcal{E}/|\mathbf{u}|$ is less than 10^{-2}. Compare these with the step sizes for the forward Euler method and comment. [1]

5. The HIRES problem is a stiff system of 8 nonlinear ordinary differential equations. HIRES stands for the High Irradiation Responses of photomorphogenesis on the basis of phytochrome, by means of a chemical reaction involving eight reactants. The concentration of the reactants is given by the vector $u(i)$ with $i = 1, 2, .., 8$ and each differential equation describes a chemical reaction as part of an overall process which is influenced by the presence of enzymes. It is a typical example of the sort of problems that arise in chemical engineering and/or biochemistry. Details of the problem are given in:

   E. Schäfer, *A new approach to explain the 'high irradiance responses' of photomorphogenesis on the basis of phytochrome*, J. Math. Biology, **2**, (1975), 41–56.

   We now integrate this system, using the explicit `ode45` (for which, as an explicit method, the step size is restricted by considerations of stability) and the variable order stiff solver `ode15s` (for which, as an implicit method, the step size is determined by considerations of accuracy rather than stability, but involves solving nonlinear equations at each step). The `ode15s` algorithm is very powerful and should be used for most stiff problems.

   The HIRES system takes the form

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}), \quad \mathbf{u}(0) = \mathbf{u}_0.$$

   Here

$$\mathbf{f}(\mathbf{u}) = \begin{pmatrix} -1.71u_1 + 0.43u_2 + 8.32u_3 + 0.0007 \\ 1.71u_1 - 8.75u_2 \\ -10.03u_3 + 0.43u_4 + 0.035u_5 \\ 8.32u_2 + 1.71u_3 - 1.12u_4 \\ -1.745u_5 + 0.43u_6 + 0.43u_7 \\ -280u_6u_8 + 0.69u_4 + 1.71u_5 - 0.43u_6 + 0.69u_7 \\ 280u_6u_8 - 1.81u_7 \\ -280u_6u_8 + 1.81u_7 \end{pmatrix}$$

   and the initial values are given by $\mathbf{u}_0 = (1, 0, 0, 0, 0, 0, 0, 0.0057)^T$.

   (a) *(i)* Write a MATLAB script file `hires.m` which integrates this problem over the range `t=[0 321.8122]` using either `ode45` or `ode15s` with the default parameters. Calculate the solution `[t,u]` in each case (you will have to wait for the `ode45` calculation) and plot (on the same graph) $\log(h(i))$ as a function of $t(i)$ in both cases, where h is the vector of time-steps $h(i) = t(i+1) - t(i)$. [1]

   (ii) Modify the script file so that it uses the `options = odeset(..)` function to set up a relative tolerance `'RelTol'` equal to $10^{-(4+m/4)}$ for $m = 0, 1, 2, .., 24$ and an absolute tolerance `'AbsTol'` equal to `'RelTol'`. For each of these values of m determine the *cputime* that it takes to run the calculation when using either `ode45` or `ode15s`. (See `help cputime` for details of this command). Plot (separate) graphs in each of these two cases of the cputime as a function of m. [1]

   (b) Using the results of Questions 3 and 4 give a careful qualitative and quantitative explanation of the form of each of the graphs in *(a)(i)* and *(a)(ii)*, comparing `ode45` and `ode15s`, and giving a mathematical justification where necessary. [2]

6. For the ODE and initial data given in problem 3, implement the second order Runge-Kutta scheme (problem 1, part (b)) and the trapezoidal scheme (problem 4) on the interval $[0, 1]$ with stepsize $h = 2^{-j}$ for $j = 2, 3, \ldots, 10$. Plot the error at $T = 1$ for both methods on the same graph on a log-log scale. Discuss what your graph shows. [Extra Credit 2]

Paul Milewski