Phys 331 - Numerical Techniques for the Sciences I.

Homework 08: Ordinary Differential Equations

Posted November 6, 2023
Due November 20, 2023

Problem 1: Solving ODEs [10pts]
Use the analytical method discussed in class (integrating factor!) to solve the following ODE, with the initial value $y(0) = -1$:

$$\frac{dy}{dx} = -2x - y. \tag{1}$$

Mathematica, Maple, MATLAB, etc are *not* allowed. Write down the expressions for $p(x)$, $g(x)$ and $\mu(x)$, and use the formula for y. The solution is

$$y(x) = -3e^{-x} - 2x + 2, \tag{2}$$

Convince yourself that this is a solution of Eq. 1.

Problem 2: Fixed-step Integrators [50 pts]
In the following, we will compare a few fixed-step integrators, namely the Euler method, and two Runge-Kutta integrators. The goal is to build a suite of ODE integrators that can be easily exchanged, thus allowing to integrate many different types of ODEs. The easiest way to do this is to break up the problem into steps of increasing detail.

(a) *The Environment:* Use the template `problem2.ipynb` to write a calling program for the stepping routine (next task). Your code should set the vectors and the initial values, call the stepping routine (below), determine the (point-wise) relative errors

$$e_i = (y_{ode,i} - y_{true,i})/y_{true,i} \quad , \; i = 1,2,...,nstep+1 \tag{3}$$

and the *rms* error

$$e_{rms} = \sqrt{\sum_i e_i^2/nstep}. \tag{4}$$

Finally, it should plot (side by side) the integrated and analytical solution, and the relative error. The ODE is given by Eq. 1.

(b) *The Stepping Routine:* Write a stepping routine for our fixed-step integrators called `ode_fixedstep`. The calling sequence is given in the corresponding template. `ode_fixedstep` integrates an ODE using a fixed step. It first determines the stepsize (set by $(x_{end} - x_{start})/nstep$), and then updates the solution by calling one of `eulerstep, rk2step,` `rk4step`, functions which perform exactly one step. As it steps through the domain, it also fills the array of support points x (this is crucial for adaptive step integrators which we will do later). Note that n steps means $n+1$ support points x_i, so make sure that you define your output vectors sufficiently large. `ode_fixedstep` returns the support points x and the solution y (see templates).

(c) *The Euler Step:* Implement the Euler method as discussed in class. The template `eulerstep` gives the calling sequence. Test your integrator with $nstep = 10, 100, 1000, 10000$, and write down the rms errors e_{rms} for each.

(d) *Runge-Kutta 2nd order:* Implement the RK2 integrator (see template `rk2step`), repeating the tests with $nstep = 10, 100, 1000, 10000$. Write down the errors.

(e) *Runge-Kutta 4th order:* Implement the RK4 integrator (see template `rk4step`), repeating the tests with $nstep = 10, 100, 1000, 10000$. Again, write down the errors and compare to the other integrators. Discuss the improvement of the solution for each integrator and step number, and compare to your expectations.