ESAM 446-1
Numerical Solution of Partial Differential Equations

Fall 2004

Hermann Riecke

Problem Set 1

Due Friday, October 1.

1. ODE: Nonlinear Duffing Oscillator

Consider the ordinary differential equation describing the frictionless motion of a particle in a quartic potential,

$$\frac{d^2u}{dt^2} = -\partial_u V(u), \text{ with } V(u) = -\frac{1}{2}u^2 + \frac{1}{4}u^4. \tag{1}$$

To solve this equation with the techniques discussed in class rewrite it in terms of a system of two equations for $u(t)$ and $v(t) = \frac{du}{dt}$. Build on the template code available on the class web site to write a MATLAB program to solve this equation numerically. To monitor the quality of your simulation have the program also calculate the total energy

$$E = \frac{1}{2}(\frac{du}{dt})^2 + V(u)$$

of the system. A particularly interesting initial condition is $u(t=0) = 1.414214$, $v(t=0) = 0$. It explores a sensitive aspect of the system, because the energy is just large enough for the particle to cross the maximum of the potential at $u = 0$.

1. **Good Coding Practice**. In MATLAB you do not have to decide before hand how large the arrays should be; MATLAB can adjust the array size as needed. However, this readjustment is slow. Measure the difference in computation time by running the template code for some initial condition up ot $t = 50$ with time step $\Delta t = 0.005$ without and with the lines in the template that preallocate the memory to the arrays `sol, energy, deltat` and `t`. To clear any allocations of memory and variable assignments between runs use the command `clear`.

2. **Forward Euler**

   (a) Validate the provided template:

      i. Measure the frequency of oscillation for the initial condition $u(t=0) = 1.0001$, $v(t=0) = 0$. Does it converge to the analytical value for small-amplitude oscillations $\omega = \sqrt{2}$? To measure the frequency reasonably precisely measure the time for 10 full oscillations.

      ii. For the initial condition $u(t=0) = 1.414214$, $v(t=0) = 0$ plot the difference of successive approximations $u_{\Delta t}(t=20) - u_{\Delta t/2}(t=20)$ as a function of the time step Δt on a log-log plot (decrease the time step by factors of 2). Use the command `loglog` instead of calculating the log and using `plot`. Does the solution $u(t=20)$ converge with the correct rate?

   (b) How small do you have to choose the time step Δt to keep the (numerically induced) energy change $\Delta E \equiv E(t=20) - E(t=0)$ below $|\Delta E| = 0.1$? Measure the computation time for this run. How does the change in energy manifest itself in the solution?

(c) Improve the program by making the time step adaptive (modify `stepeuler.m`). In each time step choose the optimal time step for a given tolerance Δ. Measure how small you have to choose Δ to reach $|\Delta E(t = 20)| = 0.1$. Compare the computation time with that of the code without adaptive time step.
Plot the maximal and minimal time step the code chooses as a function of Δ. Do you obtain a scaling similar to the scaling of the time-step as a function of the error for the code without adaptive time step (see 2(a)ii)?
Choose a tolerance Δ for which you obtain the correct *qualitative* behavior for $0 < t < 100$. Measure the computation time for that run.

3. **Runge-Kutta**

(a) Replace the fixed-time-step Euler code by a fixed-time-step fourth-order Runge-Kutta code. As you have done for the Euler code, validate your code by measuring the frequency for $u(t = 0) = 1.0001$, $v(t = 0) = 0$ and by determining the scaling of $u_{\Delta t}(t = 20) - u_{\Delta t/2}(t = 20)$ with Δt.

(b) Repeat 2b for the fixed-time-step fourth-order Runge-Kutta code.

(c) Repeat 2c for the fixed-time-step fourth-order Runge-Kutta code.

Additional comments that apply to all assignments:

- When changing time steps in convergence studies always decrease/increase it by a factor of 2 (cf. 2(a)ii).
- If you do log-log plots in matlab use the command `loglog` instead of `plot`. **Do not** take the logarithm of the data and plot that result using `plot`. (On unix machines a very nice plotting package is `xmgrace`).
- Please type your results for legibility and include the graphs in the text. Please also include your code in your write-up. As pointed out in class, part of the grade on the homework will be based on the clarity and presentation of the write-up (text structured logically, written comprehensibly, figures labelled appropriately, code written in a structured way and sufficiently well documented so that others (and you yourself in a couple of weeks) can understand it without spending hours and hours deciphering it.
- for plots: show the data as symbols rather than (or in addition to) lines connecting the values. To demonstrate that certain data fall on a line with a slope of a certain value, plot also a straight line with that slope next to the data to convince the reader that the results are indeed close to the prediction.
- in general: support your statements with appropriate plots or tables.