Problem Set 6: Solving Differential Equations
Computational Physics
Physics 480/680
James Sethna; Due Monday, April 21
Last correction at January 8, 2014, 10:21 pm

Reading
Numerical Recipes chapter 17, skimming the technical bits
Wikipedia, Verlet Integration, http://en.wikipedia.org/wiki/Verlet_integration
Web, Leapfrog method

Exercise 6.2 is optional for those in Physics 4480.

6.1 **Conservative differential equations: Accuracy and fidelity.** (Ordinary differential equations) ③

In this exercise, we will solve for the motion of a particle of mass $m = 1$ in the potential

$$V(y) = (1/8)\, y^2(-4A^2 + \log^2(y^2)). \tag{1}$$

That is,

$$d^2y/dt^2 = -dV/dy = -(1/4)y(-4A^2 + 2\log(y^2) + \log^2(y^2)). \tag{2}$$

We will start the particle at $y_0 = 1$, $v_0 = dy/dt|0 = -A$, and choose $A = 6$.

(a) *Show that the solution to this differential equation is*[1]

$$F(t) = \exp(-6\sin(t)). \tag{3}$$

Note that the potential energy $V(y)$ is zero at the five points $y = 0$, $y = \pm\exp(\pm A)$.

(b) *Plot the potential energy for* $-3/2\exp(\pm A) < y < 3/2\exp(\pm A)$ *(both zoomed in near* $y = 0$ *and zoomed out).* The correct trajectory should oscillate in the potential well with $y > 0$, turning at two points whose energy is equal to the initial total energy. *What is this initial total energy for our the particle? How much of an error in the energy would be needed, for the particle to pass through the origin when it returns? Compare this error to the maximum kinetic energy of the particle (as it passes the bottom of the well).* Small energy errors in our integration routine can thus cause significant changes in the trajectory.

(c) (Black-box) *Using a professionally written black-box differential equation solver of your choice, solve for* $y(t)$ *between zero and* 4π*, at high precision. Plot your answer along with* $F(t)$ *from part (a). (About half of the solvers will get the answer qualitatively*

[1] I worked backward to do this. I set the kinetic energy to $1/2(dF/dt)^2$, set the potential energy to minus the kinetic energy, and then substituted y for t by solving $y = F(t)$.

wrong, as expected from part (b).) Expand your plot to examine the region $0 < y < 1$*; is energy conserved? Finally, read the documentation for your black box routine, and describe the combination of algorithms being used.* You may wish to implement and compare more than one algorithm, if your black-box has that option.

Choosing an error tolerance for your differential equation limits the error in each time step. If small errors in early time steps lead to important changes later, your solution may be quite different from the correct one. Chaotic motion, for example, can never be accurately simulated on a computer. All one can hope for is a *faithful* simulation – one where the motion is qualitatively similar to the real solution. Here we find an important discrepancy – the energy of the numerical solution is drifting upward or downward, where energy should be exactly conserved in the true solution. Here we get a dramatic change in the trajectory for a small quantitative error, but any drift in the energy is qualitatively incorrect.

The Leapfrog algorithm is a primitive looking method for solving for the motion of a particle in a potential. It calculates the next position from the previous two:

$$y(t+h) = 2y(t) - y(t-h) + h^2 f[y(t)] \tag{4}$$

where $f[y] = -dV/dy$. Given an initial position $y(0)$ and an initial velocity $v(0)$, one can initialize and finalize

$$\begin{aligned} y(h) &= y(0) + h(v(0) + (h/2)f[y(0)]) \\ v(T) &= (y(T) - y(T-h))/h + (h/2)f[y(T)] \end{aligned} \tag{5}$$

Leapfrog is one of a variety of *Verlet* algorithms. In a more general context, this is called *Stoermer's rule*, and can be extrapolated to zero step-size h as in the Bulirsch–Stoer algorithms. A completely equivalent algorithm, with more storage but less roundoff error, is given by computing the velocity v at the midpoints of each time step:

$$\begin{aligned} v(h/2) &= v(0) + (h/2)f[y(0)] \\ y(h) &= y(0) + h\, v(h/2) \\ v(t+h/2) &= v(t-h/2) + hf[y(t)] \\ y(t+h) &= y(t) + h\, v(t+h/2) \end{aligned} \tag{6}$$

where we may reconstruct $v(t)$ at integer time steps with $v(t) = v(t-h/2)+(h/2)f[y(t)]$.

(d) (Leapfrog and symplectic methods) *Show that* $y(t+h)$ *and* $y(h)$ *from eqns 4 and 5 converge to the true solutions as as* $h \to 0$ *– that is, the time-step error compared to the solution of* $d^2y/dt^2 = f[y]$ *vanishes faster than* h*. To what order in* h *are they accurate after one time step?*[2] *Implement Leapfrog, and apply it to solving equation 2 in the*

[2]Warning: for subtle reasons, the errors in Leapfrog apparently build up quadratically as one increases the number of time steps, so if your error estimate after one time step is h^n the error after $N = T/h$ time steps can't be assumed to be $Nh^n \sim h^{n-1}$, but is actually $N^2h^n \sim h^{N-2}$.

range $0 < x < 4\pi$*, starting with step size* $h = 0.01$*. How does the accuracy compare to your more sophisticated integrator, for times less than* $t = 2$*? Zoom in to the range* $0 < y < 1$*, and compare with your packaged integrator and the true solution. Which has the more accurate period (say, by measuring the time to the first minimum in* y*)? Which has the smaller energy drift (say, by measuring the change in depth between subsequent minima)?*

Fidelity is often far more important than accuracy in numerical simulations. Having an algorithm that has a small time-step error but gets the behavior qualitatively wrong is less useful than a cruder answer that is faithful to the physics. Leapfrog here is capturing the oscillating behavior far better than vastly more sophisticated algorithms, even though it gets the period wrong.

How does Leapfrog do so well? Systems like particle motion in a potential are Hamiltonian systems. They not only conserve energy, but they also have many other striking properties like conserving phase-space volume (Liouville's theorem, the basis for statistical mechanics). Leapfrog, in disguise, is also a Hamiltonian system. (Eqns 6 can be viewed as a composition of two canonical transformations – one advancing the velocities at fixed positions, and one advancing the positions at fixed velocities.) Hence it exactly conserves an approximation to the energy – and thus doesn't suffer from energy drift, satisfies Liouville's theorem, etc. Leapfrog and the related Verlet algorithm are called *symplectic* because they conserve the *symplectic form* that mathematicians use to characterize Hamiltonian systems.

It is often vastly preferable to do an exact simulation (apart from rounding errors) of an approximate system, rather than an approximate analysis of an exact system. That way, one can know that the results will be physically sensible (and, if they are not, that the bug is in your model or implementation, and not a feature of the approximation).

6.2 **White noise and Fourier symmetries.** (Computation) ③

White light is a mixture of light of all frequencies. White noise is a mixture of all sound frequencies. The hissing noise you hear on radio and TV between stations is approximately white noise; there are a lot more high frequencies than low ones, so it sounds high-pitched. Both white light and white noise have roughly constant average power per unit frequency. What kind of signal in time gives white noise?

(a) White noise. *Generate* $N = 1024$ *independent random numbers* y_ℓ*, either from a normal distribution with mean zero or uniformly in* $(-1, 1)$*. Plot the points; you should see a jagged, random function. Plot the power spectrum*[3] $|\widetilde{y}_m|^2$ *with the components* $m = N/2 + 1, \ldots, N - 1$ *shifted to negative frequencies* $-N/2 + 1, \ldots, -1$*. Does the Fourier series of noise look like noise? On average, is the power in the high-frequency*

[3]For a time signal $f(t)$, the average power at a certain frequency is proportional to $|\tilde{f}(\omega)|^2$; ignoring the proportionality constant, the latter is often termed the power spectrum. This name is sometimes also used for the square of the amplitude of spatial Fourier transforms as well.

Fourier components larger, smaller, or about the same size as the low frequency (long-wavelength) coefficients? (You may wish to average the power over several random seeds, if a single run is too noisy to tell the power dependence on frequency.)

White noise is usually due to random, uncorrelated fluctuations in time.

Periodic functions form an infinite dimensional vector space: the coefficients in the Fourier-space basis $\exp(\mathrm{i}\omega_m t)$ are the Fourier coefficients $\widetilde{y}_m$.[4] The FFT only depends on $y(t)$ at N discrete points $0 \le t_\ell = T\ell/N < T$, so it describes an N-dimensional vector space. The FFT returns N *complex* numbers $\tilde{y}_m^{\mathrm{FFT}}$ with $0 \le m < N$. That doubles the number of (real) constants – how can that be?

(b) *Plot the real part and the imaginary part of the FFT of your random series, with the components $N/2+1, \ldots, N-1$ shifted to negative frequencies $-N/2+1, \ldots, -1$. Examine the region near $m = 0$ on the Fourier plot, and describe how the Fourier transform of the noisy signal is different from a random function. In particular, what symmetry do the real and imaginary parts have? Can you show that this is true for the Fourier coefficients of any real $y(t)$? Do these symmetries roughly make the dimensions (number of independent constants) agree between real space and Fourier space?*

If you are fussy, $\widetilde{y}_0$ has no partner, but is real (only one dimension), and if N is even $\widetilde{y}_{-N/2}$ also is partnerless, but is real. So N k-points are generated by N real points.

6.3 **Signal from noise: Systematic methods.** (Group project, Fourier, Fitting) ③

There are a variety of methods one can use for extracting signals from noise. Most will not be appropriate for all three of our noisy signals; some may not be effective for any of them. Each group should read up on them, and pick a few to implement.

- *Fourier methods*
  - *Power spectrum; low-pass filters*
  - *Adjusting time interval to fit periodicity*
  - *Optimal Wiener filtering*
  - *Windowing*
  - *Overlapping and non-overlapping data segments*
  - *Slepian functions*
  - *Time-domain filters*
  - *All-poles maximum entropy method*
- *Smoothing methods*
  - *Cumulative sum*
  - *Box averages*

[4] More precisely, the space of all 'reasonable' (L^2) functions has a countably infinite basis given by the Fourier coefficients. The space of all real functions of a real variable is of higher *cardinality*, and would need an uncountably infinite basis.

  - *Savitzky-Golay filters*
- *Fits*
  - *Nonlinear least-squares*
    - *Levenberg-Marquardt*
    - *Confidence intervals, sloppiness*
  - *Hidden Markov models*