Zach Stewart.

480/905: Session 7

Online listings: eigen_basis_class.cpp and diffeq_oscillations.cpp printouts.

Now that we've got routines to solve differential equations, we're going to explore some interesting ones: nonlinear oscillators. Today we'll play with a program that solves for the time dependence of such an oscillator.

Your goals for today (and ...):

- If you didn't complete it, do the plot from Session 6 of relative error at t=1 vs. mesh size h.
- Think about how to enhance the eigen_basis code with more C++ classes.
- Run a code that solves the differential equation for a (driven) nonlinear oscillator and explore how the time dependence changes as various input parameters change.
- Add friction (damping) to the code.

(Possibly) Leftover Task from Session 6

Spend no more than about 45 minutes on this.

1. **Integrating a First-Order Differential Equation.** Try to finish through part 7. If you find that the errors for Euler and Runge-Kutta lie on top of each other, most likely you have not evaluated the exact answer at precisely the same time as the last points. If you get stuck, ask an instructor.

More on C++ Classes: eigen_basis_class

The code eigen_basis_class.cpp is a simple modification of eigen_basis.cpp to use the Hamiltonian class we introduced for eigen_tridiagonal_class.cpp. Here we'll take a few minutes to think about how to introduce additional classes.

1. Take a look at the eigen_basis_class.cpp printout and note how the Hamiltonian class is re-used without modification. (If you haven't done so yet, read the discussion of this class in the Session 7 notes.) The only tricky change is that matrix indices go from 1 to dimension rather than from 0 to dimension-1. *What parts of the Hamiltonian class implementation do you not yet understand?*
   GSL is a bit like a black box, so I don't understand it's inner workings. Other than that it seems pretty standard. Are workspaces so you can better utilize certain kinds of matrices? (sym or not, etc...)?
2. The potential is another good candidate for a class. We'd like to just evaluate the potential at r without having to use constructions like the switch statement in the Hij_integrand function with all the messy void parameters. (Think about how awkward and prone to error it is to add another potential.) *What would you like the declaration statement for the Potential class to look like? What method(s) would you like the class to have?* I'd rather have a potential that could be constructed with a general function form. We could also have a constructor filled w/ presets, either by giving it specific data types or even better, a string "code". Then you could call the potential value w/ just a function call and a pass of r.
3. *Give at least one example of an additional class that would be useful to define.*
   Having too many classes can make code harder to understand. However, all the harmonic oscillator parameters and wavefunctions could be made into a class to clean up their functions. Though it really only cleans up 2 functions, but could make generalizing the problem easier by storing everything and simply calling w/ n.

Driven Nonlinear Oscillations

The Session 7 notes describe the driven nonlinear oscillator that is coded in diffeq_oscillations.cpp. Note that the force depends on k and an exponent p, the external force has a magnitude f_ext, a frequency w_ext, and a phase phi_ext. The initial conditions in position and velocity are designated x0 and v0. You also have control over the time interval (increase t_end to see longer times), the step size h, and how often points are printed to the file (plot_skip).

1. Use make_diffeq_oscillations to create diffeq_oscillations. This code outputs to the file diffeq_oscillations.dat five columns of data: t, x(t), v(t), kinetic energy, and potential energy. There are four gnuplot plot files provided (diffeq_oscillations1.plt, etc.), each of which generates a different type of plot. Run diffeq_oscillations with the default values (enter "0" when it says "What do you want to change?") to calculate a data set. Start gnuplot and "load diffeq_oscillations1.plt" and then "load diffeq_oscillations2.plt". (Once you've given these commands once, you can use just use the arrows to go back and forth.) *Briefly, what do each of these plots show?*
   1) x vs time, but it says y as a mistake?
   2) x vs velocity. Shows they are cyclical. Not very useful for undamped imo.
2. Wouldn't it be convenient to generate all four plots at once in separate files? Load "diffeq_oscillations_all.plt"!
3. It's always a question whether or not you have coded a problem correctly, so you should always seek ways to check your results. One possibility is if we have a known solution. This works for p=2 (simple harmonic oscillator). What about other p? Another check is to identify a quantity that shouldn't change with time. *Create a plot of such a quantity (you'll want to increase t_end) and observe the effect of changing the step size h to a larger value [e.g., try 10 and 100 times larger]. How do you decide on a reasonable h to use?* Plot total energy. If the variance passes a certain threshold, you need a smaller h, as this should be constant. Lower h will always increase accuracy at the cost of time, and vice versa.

   (The "plot_skip" parameter indicates how often a point is written to the output file. So plot_skip=10 means that every 10 points is output.)
4. Verify that different amplitudes (e.g., different initial conditions determined by x0 and v0) lead to different periods for an anharmonic oscillator (p<2 or p>2). [Hint: You might find the "append" option useful.] *Can you identify a qualitative rule? E.g., does larger amplitude mean shorter or longer period always? Try to explain the rule?* For $p>2$, higher velocity (amplitude) means more oscillations. Very obvious if you consider the $p \to \infty$ limit of a hard box. Throw a ball in a room fast, and it will bounce more, ya' dig?
   For $p=1$, # of oscillation is exactly inversely proportional to ~~lo~~ amplitude. $\boxed{\text{oscillations} \propto \frac{1}{A}}$
5. Go back to the original parameters (quit the program and start it again), which has p=2. *Now add a driving force f_ext=10 with w_ext=1 and look at the time dependence and phase-space plots. Then increase w_ext to 3.14 and then to w_ext=6.28. What are you observing? Now repeat with p=3 (starting with f=0). Can you find resonant behavior?* Observing increase in natuaral resonance. w=6.28 has perfect resonance w/ p=2. There will be no resonce with p>2, at least not in the form of sin(∼), as the natural motion itself isn't even sinusoidal anymore.

Adding Damping

Real-world systems have friction, which means the motion will be damped. The Session 7 notes have a list of three simple models for friction. We'll implement viscous damping: $F_f = -b*v$, where v(t) is the velocity.

1. *Introduce the damping parameter "b" into the code:*
   1. add it to the force_parameters structure (with a comment!);

2. add it to the list of local force parameters in the main program;
3. give it an initial value;
4. add a menu item (e.g., [13]) and a case statement to get a new value.

Try this part out before proceeding. *Did it work?*

Yep.

2. Modify the "rhs" routine to include damping (you're on your own here!). *What did you add?*

Just a $-b \cdot v$ to all of the $\frac{dv}{dt}$ arguments. (After assigning to two to variables.)

$w \quad 2\pi^{w_0 t}$

3. Test your routine starting with p=2 and a small damping and look at both the time dependence and the phase-space plots. Then try some other p values.
4. *Identify the three regimes described in the Session 7 notes: underdamped, critically damped, and overdamped.*

b is either $<,=,>$ $2m\omega_0 = 4\pi$ for unchanged. parameters.

Under	$b < 4\pi$
Critical	$b = 4\pi$
Over	$b > 4\pi$

EXTRA: Looking for Chaos (Part I) [we'll explore in more detail in Session 8]

Now we want to put it all together: a damped, driven, nonlinear oscillator. A different system with the same basic features is the realistic pendulum, which is described in the Session 7 notes.

1. In the notes there is a list of characteristic structures that can be found in phase space, with sample pictures. *Can you find combinations of parameters that produce pictures like these?* (Try to imitate the x(t) vs. t pictures first.)

Not without changing the shape of the potential. and or F_{ext}.
Or rather, none of these allow $x=0$ to be the minimum w/ a simple sine F_{ext}. Our system simply is not a chaotic pendulum.