480/905: Session 8

*Online handout:* plots of damped oscillations; *online listings:* filename\_test.cpp, diffeq\_pendulum.cpp, GnuplotPipe class

Strings and Things

The filename\_test.cpp code has examples of the use and manipulation of C++ strings, including building filenames the way we do stream output. **Be careful NOT to put << endl when creating filenames.**

1. Using make\_filename\_test, compile and link filename\_test.cpp and run it. Look at the output files and the printout of the code to see how it works.
2. Modify the code so that there is a loop running from 0 to 3 with index variable j. For each j, open a file with a name that includes the current value of j. *Write "This is file j", where "j" here is the current value, into each file and then close it. Did you succeed?*

**Yes**

1. Modify the code to input a double named alpha and open a filename with 3 digits of alpha as part of the name. (E.g., something like pendulum\_alpha5.22\_plot.dat if alpha = 5.21934.) *Output something appropriate to the file. Did it work?*

**Yes**

Upgrades from the diffeq\_oscillation to diffeq\_pendulum code

* There are three new menu items: plot\_start, plot\_end, and Gnuplot\_delay. The equation is still solved from t\_start to t\_end, but results are only printed out from plot\_start to plot\_end. Initially these are the same time intervals, *but you can use plot\_start to exclude a transient region.* So if the system settles down to periodic behavior at t=20, setting plot\_start=20 means that 0 < t < 20 is not plotted, which makes the phase-space plots much easier to interpret.
* We've also incorporated code to do real-time plotting in gnuplot directly from C++ programs. We have made a class to do this but it is rather crude: the interface and documentation needs work, and it probably has bugs! *Look at the GnuplotPipe.h printout and the GnuplotPipe.cpp file to get an idea how it works.* Gnuplot\_delay sets the time in milliseconds between plotted points.

Damped (Undriven) Pendulum

The pendulum modeled here has the analog of the viscous damping: Ff = −b\*v, where v(t) is the velocity, that was used in session 7. The damping parameter is called alpha here.

1. Use make\_diffeq\_pendulum to compile and link diffeq\_pendulum.cpp. Run it while taking a look at the printout of the code. It should look a lot like diffeq\_oscillations.cpp, with different parameter names. Run it with the default parameters, noting the real-time phase-space plot. There is also an output file diffeq\_pendulum.dat.
2. *Modify the code so that the output file includes two digits of the variable alpha in the name. Did you succeed?*

**Yes I did**

1. Generate the analogs of the four phase-space plots on the handout but with pendulum variables and initial conditions theta\_dot0=0 (at rest) and theta0 such that you are in the simple harmonic oscillator regime (note that theta is in radians).

Set f\_ext=0 (no external driving force) and then do four runs with four values of alpha corresponding to undamped, underdamped, critically damped, and overdamped (convert from the conditions on b discussed in the background notes). *What values of theta0 and alpha did you use?*

**Theta = 0.2, 0.4, 0.6, 0.8**

**Alpha = 0.8, 1.6, 2.4, 3.2**

Damped, Driven Pendulum

This is a quick exercise to look at transients.

1. Restart the program so that we use the defaults. There is both damping and an external driving force, with frequency w\_ext = 0.689. The initial plot is from t=0 to t=100. Run it. *The green points are plotted once every period of the external force. What good are they?*

**It shows one oscillation period**

1. Note that it seems to settle down to a periodic orbit after a while. *Plot ("by hand" with gnuplot) theta vs. t from the output file* diffeq\_pendulum.dat *and see how long it takes to become periodic.*  
     
     
   **It took around 60 seconds for it to become periodic**
2. Run the code again with "plot\_start" set to the time you just found. *Have you gotten rid of the transients? What is the frequency of the asymptotic theta(t)?*

**I did and the frequency that I got was 0.689 just by looking at the code to what I set it to. This matches the graph because the period in the no-transient graph varies by about 1.4 seconds**

Looking for Chaos

Now we want to explore more of the parameter space and look at different structures. In Section f of the Session 7 notes there is a list of characteristic structures that can be found in phase space, with sample pictures in Figure 1.

1. In phase space, a fixed point is a (zero-dimensional) point that "attracts" the time-development of a system. By this we mean that many (or all) initial conditions end up at the same point in phase space. The clearest case is a damped, undriven system like a pendulum, which ends up at theta=0 and zero angular velocity no matter how it starts. If the steady-state trajectory in phase space is a closed (one-dimensional) curve, then we call it a limit cycle.
2. Try some prescribed values for the pendulum. You will need to adjust "plot\_start" and extend the plot time (increase "t\_end" and "plot\_end"). *Try the first three combinations in this table:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **description** | **alpha** | **f\_ext** | **w\_ext** | **theta0** | **theta\_dot0** |
| period-1 limit cycle | 0.0 | 0.0 | 0.689 | 0.8 | 0.0 |
|  | 0.2 | 0.52 | 0.689 | −0.8 | 0.1234 |
|  | 0.2 | 0.52 | 0.694 | 0.8 | 0.8 |
|  | 0.2 | 0.52 | 0.689 | 0.8 | 0.8 |
| chaotic pendulum | 0.2 | 0.9 | 0.54 | −0.8 | 0.1234 |

1. *Can you tell how many "periods" the limit cycles have from the graphs? How might you identify whether a function of time f(t) is built from one, two, three, ... frequencies?*  
     
     
   **I was able to tell how many periods there are in the graphs. The frequencies that are in f(t) is determined by the number of distinct amplitudes there are in the graphs. The amplitudes tells the periods and the periods tell the frequency.**
2. One characteristic of chaos is an "exponential sensitivity to initial conditions." *For the last combination, vary the initial conditions very slightly (e.g., change x0 by 0.01 or 0.001); what happens?*

**It didn’t really change much, the initial starting point in the graph just shifts.**