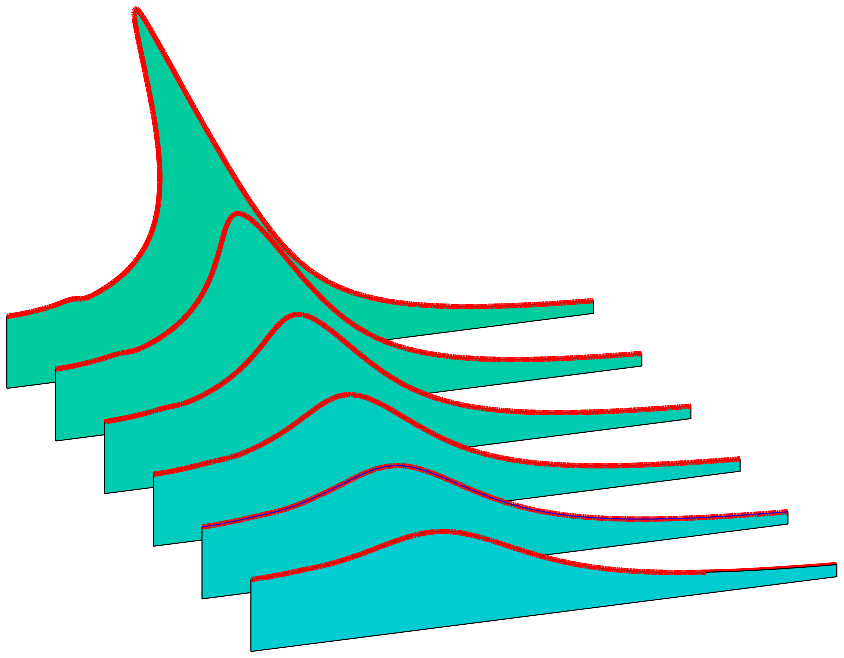
# Computational Physics 330

## Non-Linear Dynamics and Differential Equations



Ross L. Spencer and Michael Ware

Department of Physics and Astronomy
Brigham Young University

# Computational Physics 330

## Non-Linear Dynamics and Differential Equations

Ross L. Spencer and Michael Ware

Department of Physics and Astronomy
Brigham Young University

Last revised: May 6, 2021

Our objective in this course is to learn how to use numerical techniques to analyze physics problems, with a focus on ordinary differential equations. The instructor and teaching assistants will highlight the important ideas and to coach you through the laboratory exercises. This is not an independent study course. Students who try to work through this material on their own usually spend many hours looking for trivial programming mistakes and consequently don't have time to learn the nonlinear dynamics which is at the heart of the course. Attendance at the scheduled lab periods is critical.

We assume that you are already familiar with Mathematica, and we will occasionally use it to help with some of the symbolic manipulation. The first few labs focus on learning some of the necessary programming techniques. Later we use both study nonlinear dynamics, including entrainment, limit cycles, period doubling, intermittency, chaos, ponderomotive forces, and hysteresis using Matlab. This course only provides a very brief introduction to nonlinear dynamics. To master this subject, you should pursue independent reading and take more complete courses in the subject.

Suggestions for improving this manual are welcome. Please direct them to Michael Ware (ware@byu.edu).

# Contents

# Lab 1

# Introduction to Matlab

Differential equations are the language of physics, but most of the interesting problems involve differential equations that can't be solved analytically. In this course we'll learn techniques to numerically solve differential equations with the goal of studying interesting physics problems. Our first step is to learn the basics of the Matlab programming language.

## Basic Syntax

**P1.1** Read and work through *Introduction to Matlab*, Chapter 1. Type and execute all of the material written in `this kind of font`. After you have worked through the chapter, use the Matlab command line to define the matrices

```
A=[1,2,3;4,5,6;7,8,9]
B=[1,4,5;9,6,3;2,3,1]
```

Also define the row and column vectors

```
v1=[1,1,2]
v2=[0.40824829 ; -0.81649658 ; 0.40824829]
```

(a) Use both * and .* to multiply `A` and `B`. Explain the difference.

(b) Perform the operation `A./B` and explain the result.

(c) Perform the operations `A*v1`, `v1*A`, and `A*v2` and explain the results.

(d) Multiply only the center elements of `A` and `B` together

(e) Perform the operation `exp(A+i*B)` and explain what it means.

(f) Extract the center column of `A`, and then divide each element of this column by the corresponding element in `v2`.

## Writing Scripts

**P1.2** Read and work through *Introduction to Matlab*, Chapter 2. Type and execute all of the material written in `this kind of font`. Then complete the following exercises:

(a) In your Freshman physics course, you learned that in the absence of air resistance, a battleship's projectile travels a horizontal distance

$$d = \frac{v^2}{g}\sin(2\theta)\ ,$$

where $v$ is its initial speed and $\theta$ is the initial angle above the horizontal. Write a Matlab script that asks the user to enter a value for $v$ in m/s and $\theta$ in degrees, and then calculates and prints the range formatted with one decimal place, like this: "Range: 45.2 meters". Remember that the standard trig functions are permanently set to use radians, but degree versions also exist. Use your program to find the proper angle to hit a target exactly 10 km away if the initial velocity is 750 m/s. A battleship's guns can't elevate above 45 degrees.

(b) A planet's velocity with respect to its star is $\mathbf{v}_1 = 30000\,\hat{\mathbf{x}}$ m/s when it is hit by an asteroid with velocity $\mathbf{v}_2 = (-5000\,\hat{\mathbf{x}} + 8000\,\hat{\mathbf{y}} + 1000\,\hat{\mathbf{z}})$ m/s. The planet has mass $m_1 = 6 \times 10^{24}$ kg and the asteroid has mass $m_2 = 1 \times 10^{19}$ kg. Write a script that defines the masses and velocities of the planet and asteroid using the variables `m1`, `m2`, `v1`, and `v2`. Then calculate and display the final velocity of the planet after the collision:

$$\mathbf{v}_f = \frac{m_1\mathbf{v}_1 + m_2\mathbf{v}_2}{m_1 + m_2}$$

Your variables and your answer should be vectors. If it bothers you that the planet's velocity didn't change, think about the precision you are using to display it.

## Loops, Logic, and Debugging

**P1.3** Read and work through *Introduction to Matlab*, Chapters 3-4. Type and execute all of the material written in `this kind of font`, and run the code in the listings.

(a) Write a script that uses a for loop to find the factors of 24 by testing every number from 1 to 12 using the `mod` function. Once you are sure your code correctly finds all of the factors of 24, use it to find all of the factors of 18648. To pass this exercise off, you must have at least 4 comment lines in your code.

(b) Use a while loop to automate the process of finding the answer to the battleship range problem in P1.2(a). Start the elevation at zero degrees and increase it in steps of 0.1 degrees until your range exceeds 5 km, and then break out of the while loop.

(c) Your roommate has borrowed $100,000 in student loans. The loan charges a 6% annual interest rate, and interest charges are applied each month. In other words, every month the loan charges a 0.5% interest fee on the remaining balance. This means that the amount by which you reduce your balance each month is not the amount you pay, but your payment minus the monthly interest. Your roommate plans to pay your loan off by paying $1,000 per month. Write a program using loops and logic to calculate how long it will take to finish paying off the loan. Don't get fancy and derive an analytic formula for compound interest payoff time. Just write a loop to track the balance over time, and break out when the balance gets to zero.

# Lab 2

## Visualizations and Qualitative Analysis

Let's start with some review of loops, logic, and matrices.

**P2.1** Write a Matlab script that defines the following array

```
A=[14,42,91,79,95,65,3,84,93,67,75,74,39,65,17]
```

and then performs a "bubble sort" to order the array elements in `A` from smallest to largest. A bubble sort is a simple sorting algorithm that works by repeatedly looping through an array using a `for` loop, comparing each pair of adjacent items and swapping them if they are in the wrong order. The `for` is nested inside a `while` that repeats until no swaps are needed in the `for` loop. The algorithm gets its name from the way smaller elements "bubble" to the top of the list. Step through your code using the debugging commands while watching the values in `A` to make sure it is doing what you think it should. To pass this exercise off, you must have at least 6 comment lines in your code.

☞ The bubble sort is not an efficient way to sort. Matlab's `sort` command is much better, but we are learning how to program here.

## Line Plots

Matlab has a wealth of visualization tools available to help you view your data. Let's look at some of the 1-dimensional plotting tools.

**P2.2** Read and work through *Introduction to Matlab,* Chapters 5. Type and execute all of the material written in `this kind of font` and execute the examples. Then complete the following exercises.

(a) Make a graph of $f(x) = x\sin(x)$ from $x = 0$ to $x = 4\pi$. Label the axes and give the plot a title. Then overlay on the same frame a plot of $\cos(x)$ and add a legend to the plot that identifies each curve. Use Matlab help for the `legend` command to learn how to do this.

(b) Write a script with a for loop that calculates the first 20 terms of the recursion relation

$$a_1 = 1 \quad ; \quad a_{n+1} = \left(\frac{n}{(n-1/2)(n+1/2)}\right) a_n \quad .$$

and stores each value in an array `a`. Use Matlab's debugging commands to step through your code while you watch the values change in the workspace window. (HINT: $a_{20} \approx 1.3 \times 10^{-17}$)

When you are sure your program is working correctly, add some code to plot the values of $a_n$ versus $n$ using `semilogy`. Then overlay plots of $e^{-n}$ and $1/n!$ and label each line with a legend. Which function best matches the way the $a_n$ terms fall off with $n$? You must have at least four comment lines in your code to pass it off.

(c) Define an array `x` that contains the values from $x = 0$ to $x = 5$ with a step size $\Delta x = 0.01$. Make an empty array `f` the same size as `x` using the `zeros` command. Then use a for loop and logic commands to load `f` with the values:

$$f(x) = \begin{cases} e^x & , \quad 0 \le x < 1 \\ e \times \cos(x-1) & , \quad 1 \le x \le 5 \end{cases} , \tag{2.1}$$

Finally, plot $f(x)$ vs. $x$ and label your axes. You must have at least four comment lines in your script to pass it off.

HINT: For this problem *do not* use a command like

```
if x < 1
```

because `x` is an array, not a single number. You will need to address individual elements of the arrays when you do your logic tests and assignment statements.

## How does a differential equation make a curve?

Our purpose in this course is to analyze problems with differential equations. Before becoming reliant on numerical ODE solvers, you need to develop an intuition for how differential equations behave. You will need this intuition to propose and refine mathematical models for physical processes and have a sense of whether the solutions that a computer spits out are reasonable. If you don't develop good intuitive skills, the many differential equations you'll encounter in your physics courses will appear mysterious to you.

Let's look at a simple differential equation and try to translate it into words:

$$\frac{d}{dt} y = y \tag{2.2}$$

Since $\frac{d}{dt} y$ is the slope of the function $y(t)$ this differential equation says that the bigger $y$ gets the bigger its slope gets. Let's consider the two possible cases for initial conditions.

**Case 1:** $y(0) > 0$

The differential equation then says that the slope is positive, so $y$ is increasing. But if $y$ increases its slope increases, making $y$ increase more, making its slope increase more, etc. So the solution of this equation is a function like $e^t$ that gets huge as $t$ increases.

**Case 2:** $y(0) < 0$

Now the differential equation says that the slope is negative, so $y$ will have to decrease, i.e., become more negative than it was at $t = 0$. But if $y$ is more negative then the slope is more negative, making $y$ even more negative, etc. Now the solution is a strongly decreasing function like $-e^t$.

Now consider another example. Suppose that you have discovered some process in which the rate of growth of the quantity $y$ is not proportional to $y$ itself, as in exponential growth, but is instead proportional to some power of $y$,

$$\frac{d}{dt}y = y^p \tag{2.3}$$

This idea is referred to as "explosive growth." Keeping in mind that with $p = 1$ we get the exponential function, this equation says that if $y$ starts out positive, $y$ should increase even more than it did before, i.e., get bigger faster than the exponential function. That would have to be pretty impressive, and it is—$y$ goes to infinity before $t$ gets to infinity. Figure 2.1 shows a plot of the explosive growth function for the cases of $P = 2$ and $P = 3$.

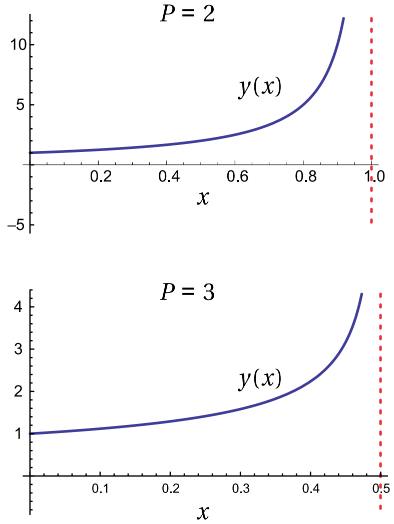


**Figure 2.1** The explosive growth function defined by Eq. 2.3 for two values of $P$.

You can play this qualitative analysis game with second-order differential equations too. Let's translate the simple harmonic oscillator equation

$$\frac{d^2}{dt^2}y = -y \tag{2.4}$$

into words. We need to remember that the second derivative means the curvature of the function: a positive second derivative means that the function curves like the smiley face of someone who is always positive, while negative curvature means that it curves like a frowny face. And if the second derivative is large in magnitude then the smile or frown is very narrow, like a piece of string suspended between its two ends from fingers held close together. If the second derivative is small in magnitude it is like taking the same piece of string and stretching your arms apart to make a wide smile or frown.

So what does Eq. (2.4) say if $y = 1$ and $y' = 0$ to start? The first derivative is zero, so $y(t)$ comes out flat, and the second derivative is negative, so the function curves downward, making $y$ smaller, which makes the frowniness smaller, but still negative, so $y$ keeps curving downward until it crosses $y = 0$. Then with $y$ negative the differential equation says that the curvature is positive, making $y$ start to smile and curve upward. It doesn't curve much at first because $y$ is pretty small in magnitude, but eventually $y$ will have a large enough negative value that $y(t)$ turns into a full-fledged smile, stops going negative, and heads back up toward $y = 0$ again. When it gets there $y$ becomes positive, the function gets frowny and turns back around toward $y = 0$, etc. So the solution of this equation is an oscillation, $\cos(t)$ or $\sin(t)$.

**P2.3** For each of the following cases, use qualitative analysis to sketch the solution of the equation on paper.

(a)

$$\frac{d}{dt}y = y^2 \quad \text{with} \quad y(0) = -1$$

(b)

$$\frac{d^2}{dt^2}y = y \quad \text{with} \quad y(0) = 1 \quad \text{and} \quad \frac{d}{dt}y(0) = 0$$

**P2.4** Don't start this problem until *after* making all of your sketches in P2.3.

(a) Verify, on paper, that the analytic solution to P2.3(a) is $y(t) = -1/(1+t)$. Use Matlab to plot this function and compare it to your sketch.

(b) Verify that the analytic solution to P2.3(b) is $y(t) = (e^{-t} + e^{t})/2$. Use Matlab to plot this function and compare it to your sketch.

# Lab 3

## Phase Space and Matlab Functions

### Surface and Flow Plots

**P3.1** Read and work through *Introduction to Matlab*, Chapter 6. Type and execute all of the material written in `this kind of font` and execute the examples. Then do the following exercises:

(a) Write a script that makes a Matlab surface plot of the "mountain" function Fig. 3.1:

$$f(x,y) = e^{-|x-\sin y|}\left(1+\frac{1}{5}\cos(x/2)\right)\left(1+\frac{4}{3+10y^2}\right). \tag{3.1}$$

Plot it from -5 to 5 in $x$ and from -6 to 6 in $y$ and add labels for the $x$ and $y$ axes. Make sure the labels correspond to the correct axes. If your plot is solid black, don't use such a fine grid in $x$ and $y$.

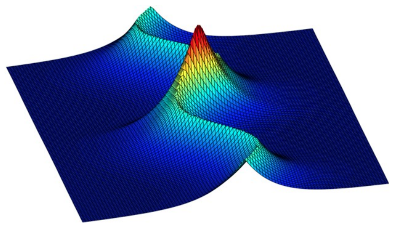


**Figure 3.1** The "mountain" function.

(b) In a certain region of the atmosphere, the wind is blowing with velocity that is constant in time, but varies spatially according to

$$\begin{aligned}\frac{dx}{dt} &= v_x = 0.2x^2+0.5y^2+20\\ \frac{dy}{dt} &= v_y = -0.1y^3+0.5x^2-10\end{aligned} \tag{3.2}$$

Write a script that makes a quiver plot of the wind velocity over the region -10 to 10 for $x$ and $y$. Now add some stream lines beginning on the left edge of your plot using the streamline command as shown in Fig. 3.2.

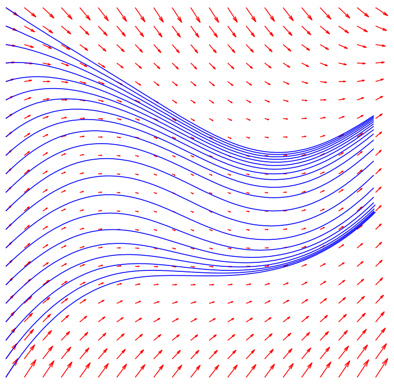


**Figure 3.2** Streamlines and wind velocity for the a wind velocity field.

The plot you created in P3.2(b) is referred to as a *flow plot.* The arrows that you produced with the quiver command show the magnitude and the direction of the velocity at each point, and the streamlines show the path that a particle would follow in this velocity field. You can also use this type of plot to understand the behavior of differential equations.

### Phase Space

You can often visualize the solution of a second-order differential equation without actually solving it using *phase space*[1] techniques. In classical mechanics you

[1] R. Baierlein, *Newtonian Dynamics* (McGraw Hill, New York, 1983), p. 51-54, 140-144, and G. Fowles and G. Cassiday, *Analytical Mechanics* (Saunders, Fort Worth, 1999), p. 93-98.

will learn to call the two-dimensional plane defined by the variables $q$ and $p = \frac{\partial L}{\partial \dot{q}}$ phase space ($L$ is the Lagrangian). But for simplicity, in this lab we will use the position $x$ and velocity $v$ as the phase space variables.

A second order differential equation can always be separated into a set of first-order equations by defining an intermediate variable. For instance, a one-dimensional projectile with the constant acceleration is described by the differential equation

$$\frac{d^2x}{dt^2} = -g \tag{3.3}$$

By defining an intermediate variable $v$, this second-order differential equation can be written as a system of first-order differential equations like this:

$$\frac{dx}{dt} = v \qquad \text{and} \qquad \frac{dv}{dt} = -g \tag{3.4}$$

Notice that the position and velocity coordinates in Eq. (3.4) have the same form as the flow velocities in Eq. (3.2), i.e. the first derivatives on the left equal expressions on the right with no derivatives. If you think of $dx/dt$ and $dv/dt$ in Eq. (3.4) as flow velocities in the $x$-$v$ plane, the right-hand sides of these equations tell you what the "flow" velocity is at each point in space. At any point in time $t$, the coordinate $[x(t), v(t)]$ gives the phase-space point that represents the "state" of the system. Given an initial starting point, you can then trace out a curve called a *phase space trajectory* analogous to the streamlines we plotted in the flow plot.

Part of the power of phase space flow plots is that you don't have to solve the differential equation to make the flow plot. You just evaluate the right-hand sides of Eq. (3.4), for example, and draw arrows at each point in the $(x, v)$ space that indicate which way the solution at that point will move if we take a small step in time. To draw the phase space trajectories, we just connect up the arrows over short time intervals (or let Matlab do it for with with `streamline`). In this way you can explore the behavior of the system for a wide range of initial conditions without ever actually solving the ODE for any of these conditions.

**P3.2** Sketch a phase space diagram for the one-dimensional projectile in Eq. (3.4) by hand on paper. Then draw some phase-space trajectories for a ball being thrown up with various velocities. *After* you have done your work by hand, check it by using Matlab with the `quiver` and `streamline` tools.

**P3.3** Make a phase space diagram for a simple harmonic oscillator given by:

$$\frac{d^2x}{dt^2} = -x \tag{3.5}$$

First sketch on trajectory by hand on a paper, then use Matlab to plot the phase space diagram with many trajectories using the `quiver` and `streamline` tools. Verbally describe the motion represented by each curve.

**P3.4** Use Matlab to plot a phase space diagram for the angle of a rigid pendulum, given by

$$\frac{d^2\theta}{dt^2} = -\sin(\theta) \tag{3.6}$$

Play with the range of the plot until you can clearly see motions that wiggle back and forth, and others that just spin around like a propeller on a plane. Identify curves that are clockwise spinning, curves that are counter-clockwise spinning, and curves that wiggle back and forth.

**P3.5** Use qualitative analysis to sketch the solution of the equation

$$\frac{d^2}{dt^2}y = -y^2 \quad \text{with} \quad y(0) = 1 \quad \text{and} \quad \frac{d}{dt}y(0) = 0$$

on paper like we did in the last lab. This ODE doesn't have an analytic solution. Make a phase-space `quiver` plot and use `streamline` to overlay a phase-space trajectory corresponding to the correct initial conditions and compare this trajectory with your sketched solution and make sure they are consistent.

## Functions in Matlab

To this point, we've mostly relied on Matlab's built-in functions tied together with some code to perform our work. As our numerical techniques advance, we'll need to be able to write our own functions. Pay close attention to this material, because it will be important throughout the remainder of the course.

**P3.6** Read and work through *Introduction to Matlab*, Chapter 7. Type and execute all of the material written in `this kind of font`. Then do the following exercises:

(a) Write an m-file function called EulerSum.m that computes the quantity

$$S_e(N) = \left(\sum_{n=1}^{N} \frac{1}{n}\right) - \ln(N)$$

You can either use a loop or the `sum` command to compute the sum. Write a separate script that loads a variable `Se` with $S_e(N)$ from $N = 1$ to $N = 1,000$. Show that as $N$ becomes large $S_e$ approaches a limit. This limit is called Euler's constant, often represented by the Greek letter $\gamma$. To 15 digits, Euler's constant is

$$\gamma = 0.577215664901532$$

Add a second output to EulerSum.m that returns the error $|S_e(N) - \gamma|$. Use `semilogy` to plot this error as a function of $N$.

(b) A square wave can be approximated by a sum of sine waves according to

$$f(x) = \sum_{n=1,3,5,...}^{N} a_n(x) \tag{3.7}$$

where

$$a_n = \frac{4}{n\pi} \sin\left(\frac{n\pi x}{L}\right) \tag{3.8}$$

Make an anonymous function that evaluates $a_n(x)$ and then write a loop that evaluates $f(x)$ for a given value of $N$. Use $L = 1$ and plot $f(x)$ from -5 to 5. Notice that your function will need to accept two arguments: $n$ and $x$. Use your code to explore how big $N$ needs to be to get a good approximation to a square wave. If you get a nice clean picture of a square wave, make your $x$ grid finer and finer until you see the Gibbs phenomenon spikes at the points of discontinuity that you learned about in Physics 318.

# Lab 4

# Calculus and a Bouncing Ball

For the past couple of labs we've focused on ways to visualize the solutions to differential equations using phase-space plots and qualitative analysis. In this lab we'll begin to learn how to solve differential equations numerically. To do this, we represent functions of space and time using discrete grids rather than continuous variables. Then we approximate derivatives as finite differences on this discrete grid rather than the infinitesimally small differences in analytic calculus. We begin by exploring how to do calculus on grids. Then we'll use these ideas to develop a crude technique for numerically solving the differential equations for a bouncing ball.

## Calculus on a Discrete Grid

**P4.1** Read and work through *Introduction to Matlab*, Chapter 8. Type and execute all of the material written in `this kind of font`. Then complete the following exercises.

(a) Use the simple mid-point rule to numerically do the integral

$$\int_0^2 x^2 e^{-x} \cos x \, dx \quad . \tag{4.1}$$

Experiment with different values of $N$ until you are confident that you have the answer correct to 6 decimal places. Then verify that you did it right by doing the same integral using Matlab's `integral` command with an anonymous function.

(b) Consider the function $f(x) = e^x$. The derivative of this function is obviously $f'(x) = e^x$, but imagine you didn't know that, and numerically evaluate $f'(x)$ at $x = 1$ using both the forward and centered difference approximation to the first derivative with a step size $h = 0.5$. Then compare the approximations with the analytic answer (i.e. $f'(1) = e$).

When you are sure you have the numerical derivatives coded correctly, switch `h` to be an array with values

$$\left[\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \cdots, \frac{1}{2^{65}}\right]$$

Then write a for loop that calculates the error of the two derivative formulas as `err=abs(fp/exp(1)-1)` (where `fp` is the numerical derivative) for each value of $h$, and stores the error in another array that is the same size as `h`. Finally, make overlaid `loglog` plots of the two errors vs. $h$. Show that the centered difference formula works better, but that both formulas are bad for very small values of $h$.

Explain why very small values of $h$ make the approximate derivative be wrong, giving zero instead of a good approximation to $f'$. The section below on roundoff will be helpful.

## Roundoff

The effect illustrated in exercise 4.1(b) is called *roundoff* and it rears its ugly head every time you subtract two numbers on a computer. To understand roundoff, consider the following two 15-digit numbers: $a = 1.2345678912345$ and $b = 1.2345678918977$. These are impressively accurate numbers, but their difference is not so impressive: $b - a = .0000000006632$. Where did all of the significant digits go; we started with 15 and now we only have 4? The problem is that the numbers were so close together that subtraction made most of the significant figures go away. When you work with numerical data on a computer you only have a finite number of significant digits (15 in Matlab), so you have to be careful when you subtract. And because subtraction is the key idea in differentiating, we have to be careful about how we choose our step size $h$. As you can see in this exercise, making it very small makes things worse, not better.

## Numerical Solutions to Differential Equations

Now that we understand the basics of taking derivatives on a grid, let's look at how to numerically solve differential equations. Consider the motion of a projectile near the surface of the earth with no air resistance. The differential equations that describe the projectile are

$$\begin{aligned} \frac{dx}{dt} &= v_x \qquad & \frac{dy}{dt} &= v_y \\ \frac{dv_x}{dt} &= 0 \qquad & \frac{dv_y}{dt} &= -g \end{aligned} \tag{4.2}$$

along with some initial conditions, $x(0)$, $y(0)$, $v_x(0)$, and $v_y(0)$. This set of equations is easily solved analytically, but imagine that we didn't have an analytic solution. How could we numerically model the motion of the projectile?

The basic idea behind a numerical solution is to think of your independent variable (time in this case) as being a discrete grid rather than a continuous quantity. It is easiest to represent time with an evenly spaced grid $[t_0, t_1, t_2, ...]$ with $t_0 = 0$, $t_1 = \tau$, $t_2 = 2\tau$, etc. Then we label the dependent variables (space in this case) using the same indexing as the time grid, like this: $x_0 \equiv x(0)$, $x_1 \equiv x(\tau)$, $x_2 \equiv x(2\tau)$, etc. With this notation, we can write the equations in (4.2) using the (inaccurate) forward difference approximation of the derivative that you learned about in the reading:

$$\begin{aligned} \frac{x_{n+1}-x_n}{\tau} &= v_{x,n} & \frac{y_{n+1}-y_n}{\tau} &= v_{y,n} \\ \frac{v_{x,n+1}-v_{x,n}}{\tau} &= 0 & \frac{v_{y,n+1}-v_{y,n}}{\tau} &= -g \end{aligned} \tag{4.3}$$

Notice that the left sides of these equations are centered on the time $t_{n+1/2}$, but the right sides are centered at time $t_n$. This makes this approach inaccurate, but if we make $\tau$ small enough it can work well enough to see the principles involved.

By solving the equations in (4.3) we can obtain a simple algorithm for stepping our solution forward in time:

$$\begin{aligned} x_{n+1} &= x_n + v_{x,n}\tau & y_{n+1} &= y_n + v_{y,n}\tau \\ v_{x,n+1} &= v_{x,n} & v_{y,n+1} &= v_{y,n} - g\tau \end{aligned} \tag{4.4}$$

This method of approximating solutions is called Euler's method. In general, it's not very good, especially over many time steps. However, it provides a foundation for learning other better methods.

☞ The name Euler does not rhyme with "cooler"; it rhymes with "boiler". You will impress your fellow students and your professors if you give this important name from the history of mathematics its proper pronunciation.

**P4.2** Make a program in Matlab to model the motion of a ball bouncing on the floor using Euler's method. In your script, define the initial position of the ball with `x=0` and `y=1`, and the initial velocity with `vx=1` and `vy=0`. Then write a while loop to step the position and velocity forward in time using Eq. (4.4). Have your while loop exit when $x > 10$. Use new variable names for the quantities at time level $n+1$, like this:

```
xnew = x + vx*tau;
vynew= vy - g*tau;
etc.
```

Then when you have advanced all four quantities, update the current values to get ready for the next step, like this:

```
x=xnew;
y=ynew;
etc.
```

(a) To simulate bouncing, put an `if` statement in your loop that checks if `y` is less than zero. When it is, make `vy` positive like this

```
vy=abs(vy)
```

Make a movie by plotting the position of the ball as a dot each time the loop iterates, like this:

```
plot(x,y,'.')
axis([0 10 0 1.5])
pause(0.001)
```

(b) Our bouncing condition in part (a) is lousy. Make it better by adding some more logic that does the following:

(i) Test to see if `y` will go less than zero on this time step, but don't actually change `y` yet.

(ii) If `y` won't go less than zero this step, just do a regular Euler step.

(iii) If it will go negative this time step, determine a smaller time step $\tau_1$ such that an Euler step will take the ball to $y = 0$. Then take an Euler step with $\tau_1$. After taking this small step, make the $y$-velocity positive as before

```
vy=abs(vy)
```

and then take an Euler step of $\tau_2 = \tau - \tau_1$ to finish off the time interval. Play with different values of $\tau$ and notice that even with this improved bouncing condition, Euler's method is always unstable (i.e. the amplitude of the bounce continues to grow). This is a limitation of Euler's method, and we'll develop better methods to overcome this shortcoming next time.

(c) Make your model look more realistic by adding some energy loss during the bounce process by changing your bounce code to look like this

```
vy=0.95*abs(vy)
```

This damping will mask the growth of Euler's method for a suitably small $\tau$.

# Lab 5

## Playing Baseball with ODEs

In the previous lab, we learned a crude method for numerically solving differential equations called Euler's method. In this lab we learn how to take the next step in refining that rudimentary technique into a more accurate ODE solver. After we have a good idea how ODE solvers are built and refined, we will introduce you to some powerful differential equation solvers built into Matlab.

## Numerically Solving Differential Equations

**P5.1** Read and work through *Introduction to Matlab*, Sections 9.1-9.2. Type and execute all of the material written in `this kind of font`. Then work the following problems.

(a) Let's start simple by modeling an object dropped from rest 10 m above the ground. Neglect air resistance, so that the gravitational force is simply $F = mg$. On a piece of paper, write down Newton's second law for this system and then convert it to a first-order set of coupled equations. Write the derivatives as finite differences on a grid in time like we did in the last lab, and solve the resulting algebra equations to derive the equations for Euler's method. (Don't peek at the answer, derive Euler's method again from scratch.)

(b) Implement your equations from part (a) in a Matlab script to solve the differential equation. Keep track of all the values, and plot $y(t)$ until your object hits the ground. Overlay a plot of the analytic solution and compare these plots for various values of $\tau$.

(c) Modify your code from part (b) to use second-order Runge-Kutta. Evaluate how your accuracy changes as you vary $\tau$, and overlay plots of the Euler's method solution, the Runge-Kutta solution, and the analytic solution. Compare the answer for various values of $\tau$.

**P5.2** Read and work through *Introduction to Matlab*, Sections 9.3.

**Important:** As you work through this material in *Introduction to Matlab*, you will learn how to use an M-file named `rhs.m` to solve differential equations. As you do this problem and in later labs, please don't keep using the name `rhs.m` over and over. Invent a unique name, like `rhs5_2.m`, and change the call to `ode45` to correspond: `ode45(@rhs5_2,...)`. This will make it possible for you to come back later and see how you did each of the problems.

(a) Use Matlab's numerical differential equation solver `ode45` to solve the motion of the particle dropped from rest at a height of 10 m (no air friction) by writing a rhs function and using Matlab's `ode45`.

(b) Make another version of your code from part (a) that uses an anonymous function instead of an external m-file rhs function. In subsequent problems and labs, you are free to use either syntax, but when the functions get complicated its usually easier to use the external rhs function.

**P5.3** Use Matlab's ode45 to numerically solve the following equation.

$$\frac{dy}{dt} = y \sin t \qquad ; \qquad y(0) = 1 \tag{5.1}$$

Plot the numerical solutions from $t = 0$ to $t = 100$ and overlay a plot of the analytic solution

$$y(t) = e^{1-\cos(t)}$$

Fiddle `RelTol` and get a feel for how the accuracy changes with this parameter. Note: this differential equation is only first order, so you won't have `u(1)` and `u(2)` this time. Think carefully about how to change your code from P5.2 to do this first-order problem.

## Baseball

In Physics 121 you did the problem of a hard-hit baseball, but because you did it without air friction you were playing baseball on the moon. Let's play ball in a real atmosphere now. The air-friction drag[1] on a baseball is approximately given by the following formula

$$\mathbf{F}_{\text{drag}} = -\frac{1}{2} C_d \rho_{\text{air}} \pi a^2 |\mathbf{v}|\mathbf{v} \tag{5.2}$$

where $C_d$ is the drag coefficient, $\rho_{\text{air}}$ is the density of air, $a$ is the radius of the ball, and $\mathbf{v}$ is the vector velocity of the ball. The absolute value in Eq. (5.2) pretty much guarantees that we won't find a formula for the solution of this problem, but that's fine since we know how to numerically solve differential equations now.

There are two forces acting on a baseball: air drag and gravity. Using Newton's second law $m\ddot{\mathbf{r}} = \sum \mathbf{F}$, we see that equation of motion for the ball is

$$m\ddot{\mathbf{r}} = \mathbf{F}_{\text{drag}} - mg\hat{\mathbf{y}} \tag{5.3}$$

where $\mathbf{r}$ is the vector position of the ball, $m$ is the mass of the baseball, $g$ is the acceleration of gravity, and we have chosen the $\hat{\mathbf{y}}$ direction to be up. Since this is

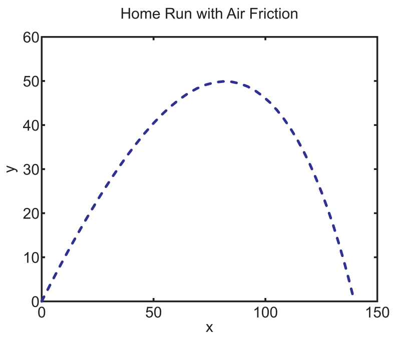


**Figure 5.1** The trajectory for a home run hit, including the effect of air friction. Note that the path is not a parabola.

[1] For more information about the subject of air drag see R. Baierlein, *Newtonian Dynamics* (McGraw Hill, New York, 1983), p. 1-7, and G. Fowles and G. Cassiday, *Analytical Mechanics* (Saunders, Fort Worth, 1999), p. 55-65.

a vector equation, it represents a system of equations—one for each dimension. To simplify our life, let's consider the motion to be just in the $x$-$y$ plane with $\hat{\mathbf{x}}$ as the horizontal direction. Using the definition of velocity, we can convert Eq. (5.3) into the following set of four coupled first-order equations

$$\begin{aligned} \frac{dx}{dt} &= v_x & \frac{dv_x}{dt} &= -\frac{C_d \rho_{\text{air}} \pi a^2 v_x \sqrt{v_x^2 + v_y^2}}{2m} \\ \frac{dy}{dt} &= v_y & \frac{dv_y}{dt} &= -\frac{C_d \rho_{\text{air}} \pi a^2 v_y \sqrt{v_x^2 + v_y^2}}{2m} - g \end{aligned} \tag{5.4}$$

**P5.4** (a) Use Matlab's ODE solver to solve the set of equations (5.4) for a baseball with the following parameters:

$$\begin{aligned} C_d &= 0.35 & \rho_{\text{air}} &= 1.2\ \text{kg/m}^3 \\ a &= 0.037\ \text{m} & m &= 0.145\ \text{kg} \\ g &= 9.8\ \text{m/s}^2 \end{aligned}$$

Put the point of contact between bat and ball at the origin ($x(0) = 0$, $y(0) = 0$). Write your initial conditions in terms of the initial angle $\theta$ and velocity $v_0$ of the baseball (i.e. $v_{0x} = v_0 \cos\theta$, $v_{0y} = v_0 \sin\theta$) so we can play with the angle and initial speed.

Plot $y(t)$ and $x(t)$ for the initial conditions of $\theta = 45°$ and $v_0 = 60$ m/s. Then plot the trajectory $y(t)$ vs. $x(t)$.

(b) Once you have your plot for the trajectory in air, overlay the trajectory that the ball would have experienced without air drag on the same plot. Estimate the difference in range caused by air friction.

(c) Power hitters say they would rather play in Coors Field in Denver than in sea-level stadiums because it is so much easier to hit home runs. Do they know what they are talking about? To find out, repeat part (a), but instead of overlaying the no air friction plot, overlay the trajectory of a ball hit in Denver and see if the ball goes significantly farther. The density of air in Denver is about 15% lower than it is at sea level.

# Lab 6

## The Harmonic Oscillator and Resonance

The harmonic oscillator is probably the most studied system in dynamics. In this lab we use the numerical tools that we have developed to explore some of the behavior of this system.[1] Before we dive into the computational details, let's remind ourselves of the basic physics of a harmonic oscillator.

## The Basic Oscillator

The basic oscillator equation is given by

$$\frac{d^2}{dt^2}x(t) = -\omega_0^2 x(t). \tag{6.1}$$

The solutions to this equation are just sines and cosines that wiggle forever in time with angular frequency $\omega_0$:

$$x(t) = A\sin(\omega_0 t) + B\cos(\omega_0 t) \tag{6.2}$$

or equivalently,

$$x(t) = A\sin(\omega_0 t + \phi) \tag{6.3}$$

**P6.1** (a) Sketch a phase space diagram for the harmonic oscillator by hand on paper. Draw at least two phase-space curves for initial conditions $x(0) = 1$, $v(0) = 0$ and another for $x(0) = 0$ and $v(0) = 1$.

(b) Use Matlab to make a phase-space plot of $x(t)$ and $v(t)$ for a simple harmonic oscillator with $\omega_0 = 2$.

(c) Solve Eq. (6.1) numerically using Matlab's ODE solver with initial values $x_0 = 1$ and $v_0 = 0$, $v_0 = 1.4$, and $v_0 = -1$ and run them from $t = 0$ to $t = 1$. Overlay the three trajectory plots on your flow plot. Identify each of the three initial conditions on your plot, and explain what the harmonic oscillator does along each trajectory.

Of course, no real oscillator wiggles forever. To model a real system we need to add damping.

[1] You can read more about the simple harmonic oscillator in the following references: R. Baierlein, *Newtonian Dynamics* (McGraw Hill, New York, 1983), Chap. 2, and G. Fowles and G. Cassiday, *Analytical Mechanics* (Saunders, Fort Worth, 1999), Chap. 3.

## The Damped Oscillator

If we add some linear damping to the system, the harmonic oscillator equation becomes

$$\frac{d^2}{dt^2}x(t) = -\omega_0^2 x(t) - 2\gamma\frac{d}{dt}x(t), \tag{6.4}$$

where the damping factor $\gamma$ describes the amount of damping—a large $\gamma$ means that there is a lot of damping. If you ask Mathematica to solve Eq. (6.4), it will tell you that the solution is

$$x(t) = Ae^{-t\left(\gamma+\sqrt{\gamma^2-\omega_0^2}\right)} + Be^{-t\left(\gamma-\sqrt{\gamma^2-\omega_0^2}\right)} \tag{6.5}$$

Equation (6.5) looks impressive, but if it's supposed to be an oscillator that damps, where are the sines and cosines? The problem is that we haven't specified how big $\omega_0$ and $\gamma$ are yet. Let's think physically for a minute.

Suppose that you put a pendulum in motor oil at 50 degrees below zero. This is an oscillator with a big $\gamma$. If you pull the pendulum back and release it, you are not going to see any swinging; the pendulum will just slowly ooze back to the vertical position and stay there. We refer to this system as being *overdamped*. Look at Eq. (6.5) and convince your lab partner that this solution is made up of decaying exponentials when $\gamma$ is big (specifically $\gamma > \omega_0$).

Now imagine what would happen if we decrease the damping, say by warming the oil up, or using WD-40 instead, or maybe even just air. In this case, the pendulum will swing back and forth, but the amplitude will decrease over time. But by what miracle did the exponential functions in the original solution become sines and cosines? Recall Euler's formula

$$e^{i\theta} = \cos(\theta) + i\sin(\theta)$$

which relates exponentials to wiggles through an imaginary argument. Note that when $\gamma < \omega_0$, the square-root in Eq. (6.5) has a negative argument, and the square root is imaginary. In this situation, we can rewrite Eq. (6.5) as

$$x(t) = e^{-t\gamma}\left[Ae^{-i\omega_d t} + Be^{i\omega_d t}\right] \qquad \text{(when } \gamma < \omega_0\text{)} \tag{6.6}$$

where the frequency at which the damped oscillator "wants" to wiggle is given by

$$\omega_d = \omega_0\sqrt{1-\gamma^2/\omega_0^2} \tag{6.7}$$

If the argument of this square root is positive ($\gamma > \omega_0$), then both of the fundamental solutions in Eq. (6.5) are decaying exponentials and we only have damping (no wiggles). The transition between the two is when the argument of the square roots is zero, i.e., when $\gamma = \omega_0$. This special case is called critical damping.

**P6.2** (a) Use Matlab to make a phase-space plot for a damped harmonic oscillator with $\omega_0 = 2$ and $\gamma = 0.5$.

(b) Solve Eq. (6.4) numerically using Matlab's ODE solver with initial values $x_0 = 1$ and $v_0 = 0$, $v_0 = 1.4$, and $v_0 = -1$, and run from $t = 0$ to $t = 20$. Overlay the three trajectory plots on your flow plot. Identify each of the three initial conditions on your plot, and explain what the harmonic oscillator does along each trajectory.

(c) Change the damping coefficient to $\gamma = 4$ and repeat (a) and (b). Explain how the flow plot describes the overdamped system.

(d) An air-damped oscillator has damping more closely proportional to the square of velocity rather than proportional to velocity. In equation form, we write this as

$$\frac{dx}{dt} = v \quad ; \quad \frac{dv}{dt} = -\omega_0^2 x - 2\gamma v|v| \,. \tag{6.8}$$

Repeat (a) and (b), but change your model to use the quadratic air-damping in Eq. (6.8) with $\gamma = 0.5$ instead of linear damping. Explain how this picture looks different from the ones in (a) and (b), and why.

## The Driven, Damped Oscillator and Resonance

If we add a sinusoidal driving[2] force at a frequency $\omega$ to the harmonic oscillator, the equation of motion becomes

$$\frac{d^2}{dt^2}x(t) = -\omega_0^2 x(t) - 2\gamma \frac{d}{dt}x(t) + \frac{F_0}{m}\cos(\omega t) \tag{6.9}$$

Now we have two frequencies in play—the driving frequency $\omega$ and the damped-oscillator frequency $\omega_d$ given by Eq. (6.7). The typical behavior of the driven-damped harmonic oscillator starting from rest is as follows: an initial period of start-up with some beating between the two frequencies ($\omega$ and $\omega_d$), then the oscillations at $\omega_d$ damp out and the system transitions to a state of oscillation at the driving frequency $\omega$.

It is possible to solve Eq. (6.9) symbolically, but let's study its behavior numerically for practice.

**P6.3** Use Matlab to numerically solve Eq. (6.9) and plot $x(t)$ from $t = 0$ to $t = 300$ with $\omega_0 = 1$, $F_0 = 1$, $m = 1$, $\omega = 1.1$, and $\gamma = 0.01$. Start from rest, with $x(0) = 0$ and $\dot{x}(0) = 0$. Note the initial beating between frequencies and verify graphically that the final oscillation frequency of $x(t)$ is $\omega$.

[2]For more information about the driven, damped harmonic oscillator, see: R. Baierlein, *Newtonian Dynamics* (McGraw Hill, New York, 1983), p. 55-62, and G. Fowles and G. Cassiday, *Analytical Mechanics* (Saunders, Fort Worth, 1999), p. 99-106.

## Resonance Curves

When you push someone in a swing, you find that if you drive the system at the right frequency, you can get large amplitude oscillations. This phenomenon is an example of resonance, and the frequency at which the system has the maximum response is called the *resonance frequency* $\omega_r$. If you drive a system at a frequency far from $\omega_r$ you only get small oscillations.

**P6.4** (a) Make a new script by modifying your script from P6.3 so it makes a plot of $x(t)$ starting from rest and running for a long time so all the beating has stopped. Use $F_0 = 1$, $m = 1$, $\gamma = 0.1$ $\omega_0 = 1$. Drive the system at $\omega = 1.1$. Then write some code that measures the amplitude of the steady-state oscillations using the colon command to select a few cycles of oscillation at the end of the time period and the `max` command to find the maximum value within these oscillations.

(b) Add a `for` loop to your code in (a) that varies the driving frequency from $\omega = 0.5$ to $\omega = 1.5$ in steps of $\Delta\omega = 0.2$. For each driving frequency, use your code to measure the steady-state oscillation amplitude $A$ (i.e. the amplitude of oscillation after all the beating has died out) and make a plot of the steady-state amplitude $A$ versus the driving frequency $\omega$. Note the region where this curve has a maximum value somewhere in the vicinity of $\omega_d$, but our resolution is too coarse to see exactly where the maximum is.

(c) To better locate the maximum, modify your loop to look at the region $\omega = 0.98$ to $\omega = 1.02$ with steps of $\Delta\omega = .001$. Find the frequency where this curve has a maximum, and compare its location to $\omega_d$ for this system. Are they the same?

In this problem you should note that the resonance frequency $\omega_r$ (i.e. the peak of the resonance curve) is not the same as the same as the damped frequency $\omega_d$. When damping is small, $\omega_r$ and $\omega_d$ are close, but they are not the same.

The plot you made in P6.3 is called a *resonance curve.* A resonance curve plots the steady state oscillation amplitude (after the beating has died away) vs. the driving frequency. You did this by brute force, but for the simple driven-damped equation we can find an analytic solution for the resonance curve. The steady-state oscillation has the form

$$x(t) = A\cos(\omega t - \phi) \tag{6.10}$$

where $A$ is the steady state amplitude, $\omega$ is the driving frequency, and $\phi$ is the phase difference between the driving force and the oscillator's response. By substituting Eq. (6.10) into Eq. (6.9) and analyzing the result, we find that the steady-state amplitude $A$ is given by

$$A(\omega) = \frac{F_0/m}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\gamma^2\omega^2}} \tag{6.11}$$

while the phase shift $\phi$ is given by

$$\tan\phi = \frac{2\gamma\omega}{\omega_0^2 - \omega^2} \,. \tag{6.12}$$

**P6.5** (a) Write a matlab script that plots Eq. (6.11) for the parameters in P6.4 and compare the plot with your numerical results. Also plot $\phi(\omega)$ and describe to your lab partner what $\phi$ represents.

(b) Now make plots of $A(\omega)$ for several values of $\gamma$ and verify that a smaller damping coefficient $\gamma$ leads to larger and sharper resonance.

(c) Show analytically that the peak of the resonance curve $A(\omega)$ is not at the damped frequency $\omega_d$, but occurs at

$$\omega_r = \sqrt{\omega_d^2 - \gamma^2} = \sqrt{\omega_0^2 - 2\gamma^2} \tag{6.13}$$

HINT: Remember that to find the peak of a curve, you take its derivative and set it equal to zero.

# Lab 7

# The Pendulum

The harmonic oscillator is an incredibly useful system to understand because it is a reasonably good approximation to essentially every system that exhibits oscillations, as long as the amplitude remains small. The classic example of a oscillating system is a simple pendulum. In this lab we'll study how the pendulum resembles a harmonic oscillator and also how it differs.

## The Simple Pendulum

The equation of motion of a simple pendulum is

$$\ddot{\theta} = -\omega_0^2 \sin\theta \,, \tag{7.1}$$

where $\theta$ is the angle (in radians) between the pendulum and the vertical direction and $\omega_0$ is the small-amplitude oscillation frequency. This is a nonlinear equation, so we often use the small angle approximation $\sin\theta \approx \theta$ to simplify Eq. (7.1) into a simple harmonic oscillator. But it doesn't take a very large amplitude before the small angle approximation falls apart. In this lab, we study the large amplitude behavior of the pendulum, which can be quite different from the simple harmonic oscillator.

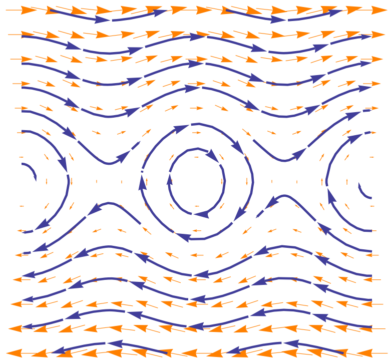


**Figure 7.1** A phase space flow plot for a pendulum.

**P7.1** Show that Eq (7.1) is, in fact, nonlinear by showing that if you have two of its solutions $\theta_1(t)$ and $\theta_2(t)$, then their sum $\theta_1(t) + \theta_2(t)$ *is not* a solution of the differential equation. When this happens, we say that the differential equation is nonlinear. Use pencil and paper; Mathematica will just slow you down.

**P7.2** Use Eq. (7.1) with Matlab to make a phase space diagram with Matlab's `quiver` and `streamline` commands. Use this diagram to describe the pendulum behavior for small oscillations, large oscillations, and motion where the pendulum is rotating completely around rather than oscillating back and forth. Identify trajectories for both clockwise and counter-clockwise rotations.

## Period and Frequency of the Pendulum

A pendulum is an extended object that is free to rotate with moment of inertia $I$ about a pivot point. The distance from the pivot point to the center of mass of the object is $\ell$, and the small-amplitude oscillation frequency is $\omega_0 = \sqrt{mg\ell/I}$. If the

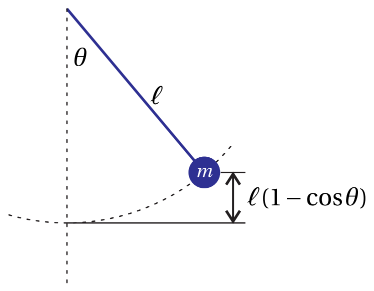


**Figure 7.2** A simple pendulum comprised of a massless stick of length $\ell$ with a mass $m$ at the end.

pendulum is a simple massless stick of length $\ell$ with all of the mass at the end of the stick, the small-amplitude oscillation frequency simplifies to $\omega_0 = \sqrt{g/\ell}$.

We can find the large-amplitude oscillation frequency of the pendulum by using an energy method.[1] The kinetic energy of the pendulum is $I\dot{\theta}^2/2$ and the potential energy is $mg\ell(1-\cos\theta)$ (see Fig. 7.2). The total energy of a pendulum can be found when the pendulum is at the maximum displacement, which we will denote by $\theta_0$. At this point, the center of mass is at a height of $\ell(1-\cos\theta_0)$ above the equilibrium position and the kinetic energy is zero, so the total energy is $mg\ell(1-\cos\theta_0)$. As the pendulum oscillates, energy shuttles back and forth between kinetic and potential according to

$$\frac{1}{2}I\dot{\theta}^2 + mg\ell(1-\cos\theta) = mg\ell(1-\cos\theta_0) \tag{7.2}$$

The first term on the left is the kinetic energy, the second term is the potential energy, and the right side is the total energy of the system.

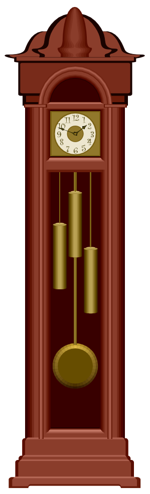


**Figure 7.3** The frequency of a pendulum depends on the amplitude of oscillation. The variation of frequency with amplitude is smallest for low-amplitude oscillations, so its easier to get good accuracy with long pendulum and small angle oscillations as in a grandfather clock.

**P7.3** Using paper and pencil, separate the variables $\theta$ and $t$ in Eq. (7.2) and show that it can be written as

$$\omega_0 dt = \frac{d\theta}{\sqrt{2\cos\theta - 2\cos\theta_0}} \tag{7.3}$$

To find the period of oscillation, we integrate both sides of Eq. (7.3) over a quarter period of the motion (from $\theta = 0$ to $\theta = \theta_0$ on the angle side and from $t = 0$ to $t = T/4$ on the time side), like this

$$\omega_0 \int_0^{T/4} dt = \frac{1}{\sqrt{2}} \int_0^{\theta_0} \frac{d\theta}{\sqrt{\cos\theta - \cos\theta_0}} \tag{7.4}$$

The time integral on the left is simply $\omega_0 T/4$, but the $\theta$ integral on the right is difficult. After carrying out the time integral and performing some judicious variable substitutions and a little algebraic massaging, we can rewrite Eq. (7.4) as

$$T = \frac{4}{\omega_0} \int_0^{\pi/2} \frac{d\phi}{\sqrt{1-\sin^2(\theta_0/2)\sin^2\phi}} \tag{7.5}$$

The $\phi$ integral in Eq. (7.5) is not any easier than the $\theta$ integral in Eq. (7.4), but it has come up in enough problems that it has been given a name: the complete elliptic integral of the first kind, called $K(m)$:

$$K(m) \equiv \int_0^{\pi/2} \frac{d\phi}{\sqrt{1-m\sin^2\phi}} \tag{7.6}$$

[1] G. Fowles and G. Cassiday, *Analytical Mechanics* (Saunders, Fort Worth, 1999), p. 318-320

Matlab and Mathematica know how to evaluate $K(m)$ functions for $0 \le m \le 1$ just like they can evaluate sines, cosines, and Bessel functions. Thus, we can write the period $T$ of the pendulum as

$$T = \frac{4}{\omega_0} K\left(\sin^2(\theta_0/2)\right) \tag{7.7}$$

Now we can use the relation $\omega = 2\pi/T$ to obtain an expression for the angular frequency of the pendulum as a function of amplitude $\theta_0$.

$$\omega(\theta_0) = \frac{\pi\omega_0}{2K\left(\sin^2(\theta_0/2)\right)} \tag{7.8}$$

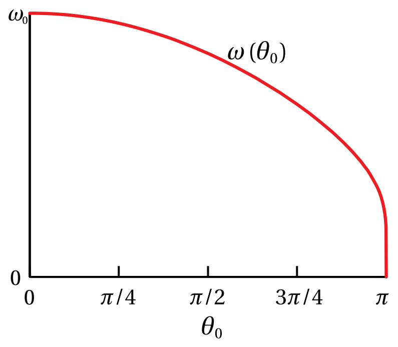


**Figure 7.4** Oscillation frequency as a function of the maximum amplitude $\theta_0$.

Note that the natural oscillation frequency $\omega(\theta_0)$ of the pendulum depends on amplitude $\theta_0$, as shown in Fig. 7.4. This gives the pendulum some interesting characteristics.

**P7.4** Use Matlab to plot $\omega(\theta_0)$ from $\theta_0 = 0$ to $\theta_0 = \pi$ with $\omega_0 = 1$ and explain physically why it looks like it does. In particular, explain why the frequency goes to zero at $\theta_0 = \pi$. You'll need to use the online help to see the syntax for evaluating the elliptic integral function.

Now let's solve the pendulum equation numerically using Matlab.

**P7.5** Use Matlab's numerical differential equation solver `ode45` to solve the pendulum, again with $\omega_0 = 1$ and initial conditions $\theta(0) = \theta_0$ and $\omega(0) = 0$. Plot the solution $\theta(t)$ for the following values of $\theta_0$: 0.1, 0.5, 1.0, $\pi/2$, $0.9\pi$, and $0.98\pi$. For each case overlay a plot of a cosine function of matching amplitude and with a frequency $\omega(\theta_0)$ from Eq. (7.8). Verify that Eq. (7.8) gives the correct frequency, but that for large amplitudes the pendulum motion is not sinusoidal.

**P7.6** Now let's study what happens when we add driving and damping.

(a) First we'll review what happens when we drive an undamped harmonic oscillator. Write a Matlab script that solves the driven oscillator equation

$$\ddot{y} + \omega_0^2 y(t) = F_0 \sin(\omega t) \tag{7.9}$$

and plot the solution $y(t)$ with $\omega_0 = \omega = 1$. Start from rest and run for a long enough time that you can see the amplitude heading off to infinity, even with small values of $F_0$.

(b) Now drive an undamped pendulum with an external torque, like this

$$\ddot{\theta} + \omega_0^2 \sin\theta = \alpha \sin\omega_\tau t\,. \tag{7.10}$$

Drive the pendulum at resonance for small amplitudes, with $\omega_0 = 1$, $\omega_\tau = 1$, and $\alpha = 0.1$. Start at rest and run for a long enough time that you can see that the pendulum amplitude doesn't simply go to infinity like the harmonic oscillator. Explain why not.

(c) Finally, add some linear damping, to the pendulum equation like this:

$$\ddot{\theta} + \omega_0^2 \sin\theta = \alpha \sin\omega_\tau t - \gamma\dot{\theta} \; . \tag{7.11}$$

Use $\gamma = 0.1$ and the same conditions as in (b) and watch how the motion changes. Explain the damped behavior and explore how it depends on $\alpha$. Also vary the driving frequency $\omega$ in the range $0.90\omega_0 \rightarrow 1.05\omega_0$ and explain why $\omega = \omega_0$ doesn't give the largest amplitude.

## Differential Equations in Mathematica

While the focus of this course is on learning numerical techniques in Matlab, Mathematica also has some excellent differential equation solving abilities that you should be aware of. Let's take a break from Matlab and learn some of the basics in Mathematica.

**P7.7** Read the section titled "Symbolic solutions to ordinary differential equations" in the Mathematica tutorial *Differential equations with Mathematica* (available on the Physics 330 course web page).

**P7.8** Use Mathematica to solve the following differential equations in general form (no initial conditions).

(a) Bessel's Equation

$$x^2\left(\frac{d^2}{dx^2}f(x)\right) + x\left(\frac{d}{dx}f(x)\right) + (x^2 - n^2)f(x) = 0$$

(b) Legendre's Equation

$$(1-x^2)\left(\frac{d^2}{dx^2}f(x)\right) - 2x\left(\frac{d}{dx}f(x)\right) + n(n+1)f(x) = 0$$

**P7.9** Read the section titled "Numerical solutions to ordinary differential equations" in the Mathematica tutorial *Differential equations with Mathematica.*

(a) Ask Mathematica to solve the following differential equation symbolically and see what happens.

$$\frac{d^2}{dx^2}y(x) = 10\sin\big(y(x)\big)\cos(x) \tag{7.12}$$

Now write the equation as a first order set, and solve it numerically with $y(0) = 0$ and $v(0) \equiv y'(0) = 0.1$. Plot $y(x)$ from $x = 0$ to $x = 100$.

# Lab 8

## Two Gravitating Bodies

Let's continue our study of differential equations by considering two masses interacting through Newton's law of gravity.[1] The Newton's second-law equations describing this situation are

$$m_1\ddot{\mathbf{r}}_1 = -\frac{Gm_1m_2}{|\mathbf{r_1}-\mathbf{r_2}|^3}(\mathbf{r_1}-\mathbf{r_2}) \tag{8.1}$$

$$m_2\ddot{\mathbf{r}}_2 = -\frac{Gm_1m_2}{|\mathbf{r_1}-\mathbf{r_2}|^3}(\mathbf{r_2}-\mathbf{r_1}) \tag{8.2}$$

where

$$\mathbf{r}_1 = x_1\hat{\mathbf{x}}+y_1\hat{\mathbf{y}}+z_1\hat{\mathbf{z}}$$

$$\mathbf{r}_2 = x_2\hat{\mathbf{x}}+y_2\hat{\mathbf{y}}+z_2\hat{\mathbf{z}}$$

There are twelve components of the motion described by these equations:

$$\begin{array}{ll} x_1(t),\ y_1(t),\ z_1(t) & \dot{x}_1(t),\ \dot{y}_1(t),\ \dot{z}_1(t) \\ x_2(t),\ y_2(t),\ z_2(t) & \dot{x}_2(t),\ \dot{y}_2(t),\ \dot{z}_2(t) \end{array} \tag{8.3}$$

so you'll need to be careful when writing out the solution.

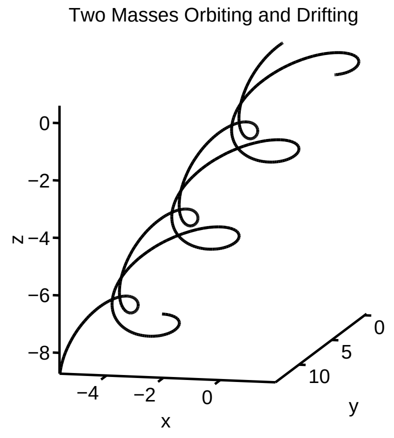


**Figure 8.1** Two masses interacting via the inverse-square law.

**P8.1** (a) Use Eqs. (8.1) and (8.2), plus $\dot{x}_1 = v_{x1}$, etc., to obtain the 12 first order differential equations for this system. Write them down on paper in terms of the individual components of the motion listed in Eq. (8.3).

(b) Now use your information from (a) to code up a right-hand-side function for this system. Have Matlab solve this system of equations using $G = 1$, $m_1 = 1$, $m_2 = 2$ and initial conditions

$x_1(0) = 1,\quad x_2(0) = -1$
$y_1(0) = 0.5,\quad y_2(0) = -0.3$
$z_1(0) = -0.3,\quad z_2(0) = 0.6$
$v_{x1}(0) = 0.65,\quad v_{x2}(0) = -0.45$
$v_{y1}(0) = 0.2,\quad v_{y2}(0) = 0.3$
$v_{z1}(0) = 0.1,\quad v_{z2}(0) = -0.3.$

Run the solution from $t = 0$ to $t = 50$, and then plot the two trajectories overlaid on the same plot using `plot3`.

[1] R. Baierlein, *Newtonian Dynamics* (McGraw Hill, New York, 1983), Chap. 5, and G. Fowles and G. Cassiday, *Analytical Mechanics* (Saunders, Fort Worth, 1999), Chap. 6.

A plot like Fig. 8.1 gives us some sense of how these two objects interact, but it doesn't show what the dynamics of this system are like. For that, it would be better to make a movie of the interaction by plotting the positions as dots, and then making a movie by showing successive plots at equal time intervals.

The problem with making movies directly from the data returned by Matlab's ODE solvers is that the data that Matlab's ODE solver returns is not equally spaced in time. You can force the Matlab functions to return equally-spaced data by giving it a list of specific times for which you want the solution evaluated, but this is computationally expensive if you need a lot of closely spaced time intervals. A better approach is to get the uneven data back from the solver and then use *interpolation* to resample it out over a much finer grid.

## Interpolation and Extrapolation

**P8.2** Read and work through *Introduction to Matlab*, Chapter 10. Type and execute all of the material written in `this kind of font`. Then write a Matlab script that creates coarse and fine grids for sin $x$ like this

```
x=0:2*pi;
y=sin(x);
xfine=-2*pi:0.1:2*pi;
yfine=sin(xfine);
```

Use linear interpolation to plot a line using the fine grid that passes through `y(1)` and `y(2)`. Then use the `pchip` method (cubic interpolation) and the `spline` method to plot a curve on the fine grid that passes through `y(1)`, `y(2)`, and `y(3)`. Overlay all of the curves: the coarse plotted as stars, the fine and the interpolated curves as lines. Use these curves to explain the benefits and hazards of using linear and cubic interpolation and extrapolation.

**P8.3** Now let's go back to your code from P8.1. After obtaining the solution arrays interpolate them onto new arrays equally spaced in time (`x1e, y1e, z1e, x2e, y2e,...` with `N=5*length(t)`). The point here is to make an evenly-spaced array of time points with 5 times as many time values as `ode45` returned, but covering the same amount of time. This can be done by defining the evenly-spaced time interval `dt=t(end)/N` and then building the evenly spaced time array like this:

```
te=0:dt:t(end)
```

Then use `interp1` to build evenly-spaced position data like this:

```
x1e=interp1(t,x1,te,'spline')
```

Now animate the motion of the two masses by using the `plot3` command. A nice way to do this animation is to use the arrays that are equally spaced in time, so that you can see the masses speed up as they approach each other, and to plot the orbits in segments of 5, or so, data points. Using just one point makes the orbits appear as sequences of dots, and using more points makes the plots be "jerky." A loop that will do this kind of animation is shown below:

```
for n=5:4:N
  plot3(x1e(n-4:n),y1e(n-4:n),z1e(n-4:n),'b-');
  hold on
  plot3(x2e(n-4:n),y2e(n-4:n),z2e(n-4:n),'r-');
  axis equal;
  pause(.1)
end
hold off
```

As your script runs you should see your masses doing an intricate gravitational dance, and the final picture should look just like the one in Fig. 8.1 (after the appropriate rotation of your figure).

## Linear Algebra

**P8.4** Read and execute the examples in *Introduction to Matlab*, Chapter 11. Then complete the following exercises.

(a) Use Matlab's `dot` command to find the angle between the vectors $\mathbf{A} = [1,2,3]$ and $\mathbf{B} = [-3,2,1]$.

HINT: You will need to calculate the magnitude of a vector to do this problem.

(b) Use Matlab's `cross` command to find the angular momentum $\mathbf{L} = m\mathbf{r}\times\mathbf{v}$ of a particle at $\mathbf{r} = [1,2,3]$ with velocity $\mathbf{v} = [6,3,1]$ and mass $m = 2.3$.

## Center of Mass Coordinates

In physics we always seek the simplest description of the motion, which is why in classical mechanics we trade in $\mathbf{r}_1$ and $\mathbf{r}_2$ for the center of mass position and the relative position of $m_1$ with respect to $m_2$:

$$\mathbf{R} = \frac{m_1\mathbf{r}_1 + m_2\mathbf{r}_2}{m_1+m_2} \quad ; \quad \mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2 \tag{8.4}$$

**P8.5** (a) Use plot3 to graph $\mathbf{R}$ and $\mathbf{V} = \dot{\mathbf{R}}$ for the initial conditions in P8.1 and show that their motion is very simple. To do the calculations in Eq. (8.4) it will be easier to transform the separate $x$, $y$, and $z$ arrays into vectors. For instance, the $\mathbf{r}_1$ vector would be a matrix with 3 columns and as many rows as there were time steps. For example to make the matrix representing $\mathbf{r}_1(t)$, you would use code like this:

```
r1=[x1,y1,z1];
```

Also define versions of these vectors with the data equally spaced in time so we can animate some plots. Once you have the **R** matrices, you can access the various components using the colon syntax. For example `R(:,1)` gives the $x$-component of **R** etc.

(b) Make a 3d plot of the difference vector **r** and use the frame rotation tool on the figure frame to see that this vector seems to sweep out a curve that lies in one plane and looks like an ellipse. Then animate your plot to show the orbit as a function of time.

(c) To see why the difference motion $\mathbf{r}(t)$ lies in a plane, compute the angular momentum in the center of mass frame

$$\mathbf{L} = m_1(\mathbf{r}_1 - \mathbf{R}) \times (\mathbf{v}_1 - \mathbf{V}) + m_2(\mathbf{r}_2 - \mathbf{R}) \times (\mathbf{v}_2 - \mathbf{V}) \tag{8.5}$$

and show numerically that this vector is constant in time. Since you have the vectors that appear on the right-hand side of this expression for **L** you can evaluate the angular momentum as a matrix (rows are time, columns are $x, y, z$ components):

```
L=m1*cross(r1-R,v1-V)+m2*cross(r2-R,v2-V);
```

And then plot each component of the angular momentum vs. time.

(d) Show graphically that **L** is perpendicular to both $\mathbf{r}_1 - \mathbf{R}$ and $\mathbf{r}_2 - \mathbf{R}$ (and hence to $\mathbf{r}_1 - \mathbf{r}_2$). To evaluate these two dot products using Matlab's `dot` command you will need to make a slight change to the syntax we used above with the `cross` command. The `dot` command when used with matrices needs to know whether we want to do the dot product along the row direction or the column direction. In this lab the rows label time, and the columns label $x, y, z$ components. Since we want to do the dot product with the $x, y, z$ components, we tell the dot product command to use the second, or column, index like this:

```
dot1=dot(r1-R,L,2)
dot2=dot(r2-R,L,2)
```

Do not panic when your plots of these two dot products look surprising; check the scale on the left side of the plot. Note that this means that the planar motion you observed in the plot of **r** is simply a consequence of conservation of angular momentum (think about this and discuss it with your lab partner until you are convinced that it is true).

**P8.6** Play around with initial condition and plot $\mathbf{r}(t)$ for a bunch of cases to see what orbital shapes you can observe. Then choose some initial conditions that make a nice ellipse. Once you have an ellipse, change the power in the denominator of the force law from 3 to 3.1 to see what kinds of orbits power laws other than inverse square make. You should find that the orbit is still sort of elliptical, but that the semi-major and semi-minor axes rotate; we call this kind of motion "precession" and it looks like Fig. 8.2.

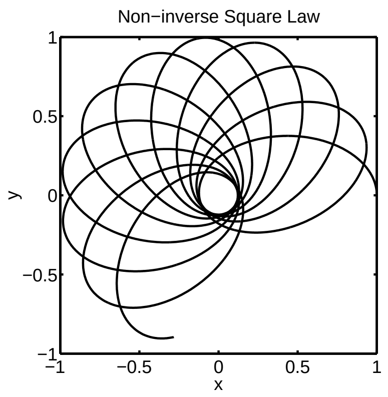


**Figure 8.2** Precession of the orbit, non-inverse-square.

In general relativity the gravitational force law is not precisely inverse-square, so this kind of precession is expected to occur. Mercury's orbit has a small precession of this kind (the famous "precession of the equinox of Mercury") which has been measured for centuries. When Einstein's equations correctly predicted this precession it was a major triumph for his theory of general relativity.

# Lab 9

# Fourier Transforms

## Fourier Transforms

Suppose that you went to a Junior High band concert with a digital recorder and made a recording of Mary Had a Little Lamb. Your ear told you that there were a whole lot of different frequencies all piled on top of each other, but perhaps you would like to know exactly what they were. You could display the signal on an oscilloscope, but all you would see is a bunch of wiggles. What you really want is the spectrum: a plot of sound amplitude vs. frequency.

The mathematical method for finding the spectrum of a signal $f(t)$ is the Fourier transform

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt \tag{9.1}$$

If you remember Euler's relation $e^{i\omega t} = \cos(\omega t) + i\sin(\omega t)$, you can see that the real part of $g(\omega)$ is the overlap of your signal with $\cos(\omega t)$ and the imaginary part of $g(\omega)$ is the overlap with $\sin(\omega t)$.[1] Often, we aren't interested in the phase information provided by the complex nature of $g(\omega)$, so we just look at the *power spectrum* $P(\omega)$

$$P(\omega) = |g(\omega)|^2 \tag{9.2}$$

$P(\omega)$ gives the signal intensity as a function of frequency without any phase information. In this lab, we will learn how to make these types of plots.

## FFTs and Fourier Transforms

**P9.1** Read and work through *Introduction to Matlab*, Chapter 13. Type and execute all of the material written in `this kind of font`.

**P9.2** On the class web site is a file called "Beethoven.wav" that has the first four notes of Beethoven's 5th symphony. Save it to your computer and listen to it. Load the sound waveform into the matrix `f` using

```
f = audioread('beethoven.wav');
```

Then construct the corresponding `t` time series by noting that the recording was sampled at 44100 points/second. Plot the signal versus time and plot its power spectrum versus $\nu$ (not $\omega$) over the range 0-1000 Hz with both a linear scale and with `semilogy`. (You should get in the habit of looking at

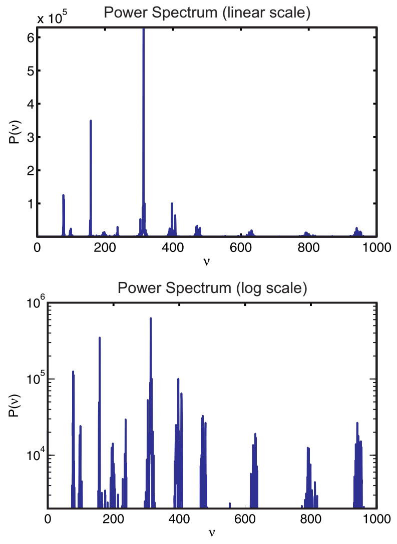


**Figure 9.1** The power spectrum of the first four notes of Beethoven's 5th symphony.

[1]People often work with complex signals, in which case this separation is less clear.

spectra with a log scale to see structure that may not be evident on a linear scale.)

Now we need to make sense of the spectrum. The short notes at the beginning of the music are the note "G" (repeated three times) played in octaves by the violins/violas (394 Hz), cellos (197 Hz), and basses (99 Hz). The last note is an "E-flat," again played in octaves by the various stringed instruments (312 Hz, 156 Hz, and 78 Hz). Identify each of these peaks on the spectrum, and explain what their relative amplitudes mean.

Note that there are also smaller peaks at 234 Hz, 468 Hz, 624 Hz, 788 Hz, and 936 Hz. Explain where these extra peaks come from, and how each of the smaller peaks are connected to the notes in the four-note theme (see Fig. 9.2).

To convince yourself that you know what you are doing, split the time series into two pieces: one that contains the first three short notes, and a second that only contains the one last long note. Then repeat the analysis above and compare your two new spectra to the original one. Show the TA all three spectra and explain the origin of all of the peaks.

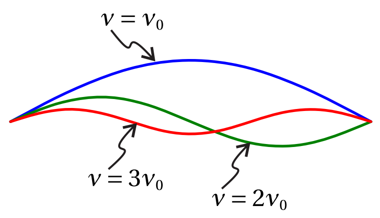


**Figure 9.2** A string's fundamental mode of vibration has nodes at the ends and an antinode in the middle. However, the string can also vibrate in harmonic modes with nodes between the ends. When a musician drags her bow across a string, she excites mostly the fundamental, but the harmonics are also present. The frequencies of these modes are: $\nu_0$ = fundamental, $2\nu_0$ = second harmonic, $3\nu_0$ = third harmonic, etc.

## The Uncertainty Principle

The uncertainty principle connects the duration of a signal in time with the spread of its spectrum. It was made famous in quantum mechanics by Werner Heisenberg, but it is really an idea from classical wave physics[2] which we can understand by using the `fft`.

Suppose that we have a time signal which has a frequency $\omega_0$, but which only lasts for a finite time $\Delta t$. For example, consider the Gaussian function

$$f(t) = \cos(\omega_0 t) e^{-(t-t_0)^2/W^2} \tag{9.3}$$

which has a "bump" centered at $t_0$ with a width controlled by $W$. Because the signal oscillates at $\omega_0$ we would expect to see a peak in the spectrum at $\omega = \omega_0$. This frequency peak also has a well-defined width, and this width is related to the width of the signal in time through the uncertainty principle.

**P9.3** Write a Matlab script to build $f(t)$ from Eq. (9.3), with $t_0$ chosen so that the bump is in the center of your time window. Plot $f(t)$ and its power spectrum for $\omega_0 = 200\ \mathrm{s}^{-1}$ and $W = 10$, 1, and 0.1.

Choose appropriate values for your number of points $N$ and your time step $\tau$ so that

(i) `fft` will run fast

[2]The weirdness of quantum comes not from the fact that waves obey the uncertainty principle, but from the idea that things like electrons behave like waves.

(ii) you can see frequencies up to $\omega = 400$ s$^{-1}$ without aliasing trouble

(iii) your spectral resolution will be at least $d\omega = 0.2$ s$^{-1}$.

To see where the uncertainty principle is lurking in these plots, visually measure and write down the full width at half maximum (FWHM) of the time signal ($\Delta t$) and FWHM of the frequency peak ($\Delta\omega_{\text{plot}}$). Write these measurements in Table 9.1 for each value of $W$. Then deduce a rough product relation $\Delta\omega\Delta t \approx$ const between the width of the time signal and the width of the frequency peak from this data.[3]

| $W$ | $\Delta t$ | $\Delta\omega_{\text{plot}}$ | $\Delta t\Delta\omega_{\text{plot}}$ |
|---|---|---|---|
| 10 | | | |
| 1 | | | |
| 0.1 | | | |

**Table 9.1** Enter your data here

You have probably experienced the uncertainty principle when listening to music. For a musical instrument to play a nice-sounding note the width of its spectrum must be narrow relative to the location of the peak. So for a flute playing a high note at $\omega = 6000$ s$^{-1}$ to produce a spectrum with, say, a 1% width requires $\Delta\omega = (0.01)(6000 \text{ s}^{-1}) = 60 \text{ s}^{-1}$. Then the uncertainty principle tells us that this note can be produced by only holding it for the relatively short time of

$$\Delta t \approx \frac{1}{60} = 0.017 \text{ s}$$

where we have arbitrarily chosen $\Delta\omega\Delta t = 1$ to make the calculation. But when a tuba plays a low note around $\omega = 200$ s$^{-1}$, the same calculation using $\Delta\omega = (200)(.01) = 2$ gives a note-duration of only

$$\Delta t \approx \frac{1\pi}{\Delta\omega} \approx \frac{1}{2} = 0.5 \text{ s}$$

Now tubas can play faster than this, but if you listen carefully, when they do their sound becomes "muddy", which simply means that the note isn't a very pure frequency, corresponding to a wide frequency peak.[4] Your ear/brain system also helps you out here. It is pretty talented at turning lousy signals into music, so you can still enjoy "Flight of the Bumblebee" even when played by a tuba.[5]

You can also hear this effect simply by clapping your hands. If you cup your hands when you clap, you trap a lot of air, which responds rather slowly to your clap. This makes a larger value of $\Delta t$, which in turn means that $\Delta\omega$ is smaller, corresponding to the low frequencies that make up the low, hollow boom of a cupped clap. But if you slap your third and fourth fingers quickly on your palm you trap almost no air, resulting in a very small $\Delta t$, and hence, via the uncertainty principle, a larger $\Delta\omega$. And a larger $\Delta\omega$ means a higher set of frequencies in the sound of your clap, which you can clearly hear as a higher-pitched burst of sound.

[3]This is not a mathematically rigorous uncertainty relation, but it illustrates the idea.

[4]The length of the tuba also contributes to the "muddyness" of the sound, since it takes a while for sound to propagate back and forth between the mouthpiece and the bell and set up the standing wave. This causes a messy "attack" transient at the beginning of each note, which means you have less of the sustained pitch to listen to.

[5]At tuba frequencies, your ear/brain system can perceive pitch for pulses containing only a few cycles.

## Windowing

Review the material on windowing in *Introduction to Matlab*, then work through the following problem.

**P9.4** Modify Listing 13.1 in *Introduction to Matlab* so that it uses the following time signal

```
f=sin(t)+.5*sin(3*t)+.4*sin(3.01*t)+.7*sin(4*t)+.2*sin(6*t);
```

Plot the power spectrum versus $\omega$ and verify the relative amplitude problem discussed in the windowing section in *Introduction to Matlab*. To make the ratio issue clear, normalize the spectrum so the biggest peak has height 1 (i.e. plot `P/max(P)` instead of `P`).

Multiply the time signal by a Gaussian window function like this

```
win = window(@gausswin,length(f),alpha)';
f = f .* win;
```

The transpose operator (`'`) at the end of the first line switches the window from a column vector to a row vector so that the multiplication works. The parameter `alpha` is specific to a Gaussian window, and is related to Eq. (9.3) via $\alpha \propto 1/W$—i.e. a bigger $\alpha$ creates a narrower signal in time. Try several values of `alpha` and look at plots of `win` and `f.*win` to see what the window function does.

Make the window really narrow with `alpha=25` and plot the power spectrum of `f.*win`. Look at the peaks at $\omega = 1, 4, 6$, and verify that the relative amplitudes are now right on. (Remember that power is proportional to amplitude squared.) But what happened to the peaks at $\omega = 3$ and $\omega = 3.01$? We've made the peaks so broad that they've smooshed into each other due to leakage. Find an `alpha` that is a good compromise between getting the right peak amplitude and maintaining good resolution. Explain the concepts of windowing and leakage, and tell how they relate to resolving the height and width of closely spaced peaks.

**P9.5** Use Matlab to numerically verify the trig identity $\cos^4(t) = 3/8+(1/2)\cos(2t)+(1/8)\cos(4t)$ by plotting the Fourier transform of the function. You will need to choose an appropriate time series and window function to see the relationships accurately.

# Lab 10

# Pumping a Swing

A playground swing is basically a driven and damped pendulum. But there are two ways to pump a swing: angular momentum pumping and parametric oscillation. In this lab we'll study and numerically model both methods.

## Pumping With Angular Momentum

You are probably most familiar with angular momentum pumping. In this technique, you sit on the seat and lean back, then lean forward, and lean back, etc. You enhance angular momentum pumping when you stretch your legs out in front as you swing forward and lean back, then tuck your legs back under as you swing backward and lean forward. You can see why this works by imagining yourself suspended in outer space with your arms extended to the side. If you were to move your right arm up and your left arm down, your body would twist sideways in the opposite direction to conserve angular momentum. Now imagine doing the same thing with your arms while sitting on a swing. When your body twists opposite to your arms to try to conserve angular momentum, friction between your jeans and the swing seat will drag the swing with your body and you will start the swing moving to the side.

Usually you want to pump a swing forward and backward rather, rather than side to side. Since your torso and legs have more mass than your arms, you can do a better job of pumping the swing by leaning your body and moving your legs than you can by waving your arms. When you twist your body backward, you exert a torque on the swing in the forward direction, and when you sit up again you create a torque in the other direction. When you repeat these motions at the resonant frequency $\omega_0$ of the swing, you will be resonantly driving the pendulum, as we discussed in lab 7. This seems to be something that kids on a playground just do without knowing any physics at all.

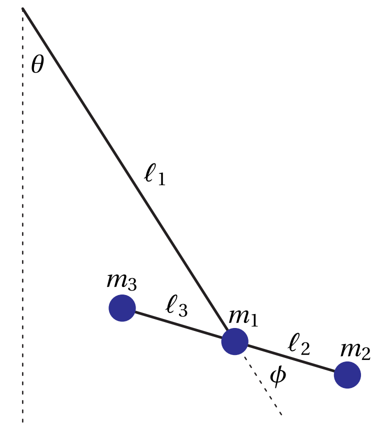


**Figure 10.1** A simple model of a swing being pumped from the seated position.

To see how this works analytically, consider a the model of a swinger shown in Fig. 10.1.[1] The overall position of the swing is described by $\theta$. The swinger is represented by the masses $m_1$, $m_2$, and $m_3$ and the possibility for the swinger to twist is represented by $\phi$. If we make the assumption $m_2\ell_2 = m_3\ell_3$, the Lagrangian for this system simplifies to

$$L = \frac{1}{2}I_1\dot{\theta}^2 + \frac{1}{2}I_2(\dot{\theta}+\dot{\phi})^2 + Mg\ell_1\cos\theta \tag{10.1}$$

where $M = m_1 + m_2 + m_3$, $I_1 = M\ell_1^2$, and $I_2 = m_2\ell_2^2 + m_3\ell_3^2$.

[1]This model is taken from W. B. Case and M. A. Swanson, "The pumping of a swing from the seated position", American Journal of Physics **58**, 463-467 (1990).

**P10.1** On paper, use the Lagrangian equation of motion for $\theta$, i.e.

$$\frac{\partial L}{\partial \theta} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = 0$$

to generate the equation of motion for $\theta$. Then assume that the swinger twists harmonically, with

$$\phi(t) = A + A\cos(\omega_\phi t) \tag{10.2}$$

and show that the equation of motion becomes

$$\ddot{\theta} + \omega_0^2 \sin\theta = \alpha\cos(\omega_\phi t) \tag{10.3}$$

where

$$\omega_0^2 = \frac{Mg\ell_1}{I_1 + I_2} \qquad \alpha = \frac{I_2 A\omega_\phi^2}{I_1 + I_2} \tag{10.4}$$

Compare with Eq. (7.10) and confirm that this is the equation for a driven pendulum.

**P10.2** Add some friction to Eq. (10.3) by appending a linear damping term $-\gamma\dot{\theta}$ on the right-hand side, and then solve the modified equation numerically using Matlab. Use the following ballpark numbers for a child on a playground swing: $\ell_1 = 2$ m, $m_1 = m_2 = m_3 = 10$ kg, $\ell_2 = \ell_3 = 0.5$ m, and $A = 0.5$ rad (about 30°). By experiment with a backyard swing, we find that $\gamma = 0.1\ \text{s}^{-1}$ is reasonable. Start from at-rest conditions and pump with $\omega_p = \omega_\phi = \omega_0$. Plot the solution from $t = 0$ to $t = 800$ s and note that it comes to a steady state amplitude as driving and damping balance. Then use the following code to animate the pumping process.

**Listing 10.1** (pumpanimate.m)

```
% Put the code solving the equation above. The code below assumes
% that you have evenly spaced time steps with the following variables
% te    -> time array
% xe    -> angle theta
% wp    -> the pumping frequency
% A     -> the pumping amplitude
% l1    -> the main swing length
% l2    -> the head-to-middle and middle-to-foot length

tau=te(2)-te(1); L = l1+l2;

for istep=1:length(te)
    % Position of swing relative to the pivot
    theta = xe(istep);
    xswing=l1*sin(theta);
    yswing=-l1*cos(theta);

    % position of head/legs with respect to swing
    phi=A + A*cos(wp*te(istep));
    xpers=l2*sin(phi + theta);
    ypers=l2*cos(phi + theta);

    % Plot the swing and the swinger
    plot([0, xswing],[L,L+yswing],...
         [xswing+xpers,xswing-xpers],[L+yswing-ypers,L+yswing+ypers])

    % Make the x and y dimensions scale equally
    axis([-L/2 L/2 0 L])
    axis square

    % We'd like the plots frames to show at intervals of tau so the movie
    % matches the physical time scale.  However, the calculations
    % and plotting take some time, so we decrease the pause a bit.
    % Depending on the speed of your computer, you may need to adjust
    % this offset some.
    pause(tau-0.01)
end
```

## Pumping With Parametric Oscillations

You can also pump a swing by standing on the seat and doing deep knee bends. As you start to swing forward you bend your knees and then stand up hard as you go through the bottom of the motion. When you start back you repeat this motion by bending your knees and then standing up hard as you go backward through the bottom of your motion.[2] This was easy to do on the solid wooden seats that swings used to have. However, after a couple of generations of kids getting their teeth knocked out by these wicked flying planks, playgrounds put in soft flexible seats. These seats are safer, but they are hard to stand on while moving your body up and down, so you may not have pumped a swing this way. Nevertheless, it is a very good way to pump a swing, although it seems a bit mysterious since no rotation is taking place. All you do is move your center of mass up and down vertically and rotation of the swing magically appears. The name of this mysterious technique is *parametric oscillation*.

The so-called *parametric oscillator* equation is [3]

$$\ddot{x} + \gamma\dot{x} + \omega_0^2(1 + \epsilon\cos(\omega_p t))x = 0\,. \tag{10.5}$$

Notice that this is different from a driven oscillator because the oscillating term $\cos(\omega_p t)$ is multiplied by $x(t)$. What is happening here is that the natural frequency (one of the system parameters) is wiggling in time, which is why this is

[2]There are some nice videos of pumping a swing this way at http://retro.grinnell.edu/academic/physics/faculty/case/swing.

[3]L. D. Landau and E. M. Lifshitz *Mechanics* (Pergamon Press, New York, 1976), p. 80-83, and M. Abramowitz and I. A. Stegun *Handbook of Mathematical Functions* (Dover, New York, 1971), Chap. 20.

called a parametric oscillator. In the case of a swing (which is really just a pendulum) you are changing the distance $\ell$ from the top of the chain to your center of mass. Since the natural frequency of a pendulum is given by $\omega_0^2 = g/\ell$, as you wiggle your center of mass you are wiggling the natural frequency. Equation (10.5) is simpler than the real equation for a pendulum. We will do the pendulum correctly later in this lab, but for small oscillation angles of the swing Eq. (10.5) gives a reasonable approximation to the the motion of a swing.

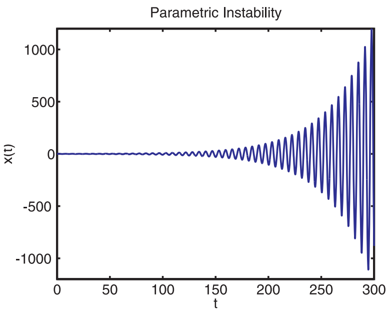


**Figure 10.2** Pumped swing instability.

**P10.3** (a) Use Matlab to solve Eq. (10.5) with initial conditions $x(0) = 0$ and $v(0) = 1$, and parameters $\omega_0 = 1$, $\gamma = 0$, $\epsilon = 0.1$, and $\omega_p = 1.1$. Plot the solution $x(t)$ for a long enough time that you can see that nothing much happens except wiggles with some beating between the natural motion at $\omega_0$ and the parametric drive at $\omega_p$.

Then run your code again with $\omega_0 = 1$, $\omega_p = 1$, $\gamma = 0$, and $\epsilon = 0.1$. This matches the pumping frequency with the natural frequency of the swing, something you might expect would resonantly drive the oscillator. Verify that the system is only weakly unstable (meaning that the motion slowly grows exponentially with time.) You might have to run for a long time to see this instability.

(b) Now run your Matlab code again with $\omega_p = 2$ and watch what happens. You should reproduce Fig. 10.2. Verify that $\omega_p = 2\omega_0$ is more unstable (i.e. the amplitude grows faster) than $\omega_p = \omega_0$. Once you see the $2\omega_0$ instability on your screen, come observe it with the physical pendulum at the front of the class.

(c) Show by numerical experimentation that the oscillator is unstable at $\omega_p = 2\omega_0$ for all choices of $\epsilon$, but that the instability growth rate is small for small $\epsilon$.

(d) Show that when $\omega_p$ is not quite $2\omega_0$ the oscillator is stable for small $\epsilon$, but that when $\epsilon$ exceeds some threshold, it becomes unstable again. Find this threshold value for $\omega_p = 2.05\omega_0$ and for $\omega_p = 1.95\omega_0$.

(e) Now add damping by setting $\gamma = 0.03$ and show that there is a threshold value of $\epsilon$ even at $\omega_p = 2\omega_0$. Find it by numerical experimentation.

You should have discovered by now that the best way to parametrically drive an oscillator is to use $\omega_p = 2\omega_0$. Is this what you do when you pump a swing by standing on the seat? Think about how often you move your center of mass up and down in one period of the swing and explain to your TA how $\omega_p$ and $\omega_0$ are related as you do this.

## Interpreting the Spectrum of the Parametric Oscillator

To gain some insight into why $\omega_p = 2\omega_0$ is more unstable than $\omega_p = \omega_0$ it is helpful to look at the power spectrum of $x(t)$ for the parametric oscillator. In doing this analysis we will use a form of perturbation theory which all physicists love, but which you may not have seen. So before we look at the spectrum of x(t), let's do a perturbation theory problem as a warm-up.

Suppose that you wanted to solve the equation

$$x^3 = 1 + 0.1e^x \tag{10.6}$$

for a real solution near $x = 1$. This equation is horrible, but if it weren't for the $e^x$ term, it wouldn't be so bad: $x^3 = 1$ so $x = 1$. But look; the exponential term is not so important since it is multiplied by 0.1, which is small. Shouldn't we be able to exploit this smallness somehow? The answer is yes, and here is how to do it, step by step.

**Step 0:** Ignore the small $e^x$ term altogether and just solve the easy equation:

$$x^3 = 1 \quad \Rightarrow \quad x_0 = 1 \tag{10.7}$$

We call this beginning, and easiest, solution $x_0$ to keep track of which step we are in.

**Step 1:** We will now get a better approximation to the solution by writing the equation down again, but with a twist in the small exponential term. We will guess that since it is small, it might be OK to replace the horrible $e^x$ by an approximate version of it, namely $e^{x_0}$:

$$x^3 = 1 + 0.1e^{x_0} \quad \Rightarrow \quad x^3 = 1 + 0.1e^1 \quad \Rightarrow \quad x_1 = (1 + 0.1e^1)^{1/3} \tag{10.8}$$

The replacing of $e^x$ by $e^{x_0}$ again made the equation easy to solve, which is good, but we still haven't found the correct solution.

**Step 2:** To further improve our solution we repeat step 1, writing

$$x^3 = 1 + 0.1e^{x_1} \quad \Rightarrow \quad x^3 = 1 + 0.1\exp\left[(1 + 0.1e^1)^{1/3}\right] \tag{10.9}$$

$$\Rightarrow \quad x_2 = \left(1 + 0.1\exp\left[(1 + 0.1e^1)^{1/3}\right]\quad\right)^{1/3}$$

**P10.4** This procedure starts to look ugly analytically, but if we just want a numerical answer there is no point in writing all of this out. Solve Eq. 10.6 by continuing this step by step approach all the way to 15 significant figures in the Matlab command window by typing

```
format long e
x=1
```

and then

```
x=(1+0.1*exp(x))^(1/3)
```

and then using the ↑ key to repeat the last step over and over again. Just watch the result for $x$ and quit when the digits in the answer quit changing. You should find that the procedure converges to $x = 1.090733645657879$; verify that this is the solution to the equation

$$x^3 = 1 + 0.1e^x$$

This is a very powerful trick and we will now use it to understand the parametric instability at $\omega_p = 2\omega_0$.

**P10.5** Using $x(0) = 0$, $v(0) = 1$, $\omega_0 = 1$, $\omega_p = 1.3$, $\gamma = 0$, and $\epsilon = 0.3$, run your model from $t = 0$ to $t = 500$ with $2^{14}$ equally spaced time steps. Take the Fourier transform and display its power spectrum using a `semilogy` plot. Our upcoming analysis will be easier if we consider negative frequencies, so use `ft.m` from Chapter 14 of *Introduction to Matlab* and construct your frequency array appropriately.

You should immediately notice the big peaks at $\pm\omega_0$. This is not a surprise because what we have is an oscillator at frequency $\omega_0 = 1$ plus a small perturbation of size $\epsilon$ at frequency $\omega_p = 1.3$. But if you look for a peak at $\omega = 1.3$, you won't find it, even though there are plenty of other peaks. Our job now is to explain why these other peaks are where they are.

Since $\epsilon$ is small and the damping is weak, let's begin by ignoring them both ($\epsilon = 0$ and $\gamma = 0$). (This is step 0 in our perturbation analysis.) Then note that with these simplifications Eq. (10.5) is solved by

$$x_0(t) = A\cos(\omega_0 t) \tag{10.10}$$

Now we will proceed by perturbation theory as we did in the previous problem, like this. Make a more precise guess at the solution by writing Eq. (10.5) down again, but with $x_0$ in place of $x$ in the small term $\omega_0^2\epsilon\cos(\omega_p t)x$:

$$\ddot{x} + \gamma\dot{x} + \omega_0^2(1 + \epsilon\cos(\omega_p t))x_0 = 0 \quad \Rightarrow \tag{10.11}$$

$$\ddot{x} + \gamma\dot{x} + \omega_0^2 x \approx -\epsilon\omega_0^2\cos(\omega_p t))x_0$$

With $x_0 = A\cos(\omega_0 t)$ this is just a complicated version of the driven harmonic oscillator.

**P10.6** Use Mathematica to solve this equation with $\gamma = 0$. You may need to recall that inhomogeneous linear differential equations like this have solutions of the form $x = x_h + x_p$, where $x_h$ is the homogeneous solution (without the parametric driving term) and where $x_p$ is the particular solution with the driving term included. Mathematica will give you a pretty complicated answer, but if you look at it closely you will see that the homogeneous solution is just our friend $x_0 = A\cos(w_0 t)$ and that the particular solution (the messy part) can be written as a sum of terms that involve sines and cosines at new frequencies. It's as if the driving force had a split personality involving more than one driving frequency. This is exactly right, as you can see by using the identity

$$\cos\alpha\cos\beta = \frac{1}{2}\left(\cos(\alpha+\beta)+\cos(\alpha-\beta)\right)$$

to rewrite the driving term on the right side of the differential equation above. Do the math and see what frequencies turn up. Can you see these"sideband" frequencies in your spectrum from P10.5 and in your Mathematica solution?

In this first-order perturbation theory we see that in addition to the peak at $\omega_0$, there are two other, smaller, sideband contributions at $(\omega_0+\omega_p)$ and at $(\omega_0-\omega_p)$. If we now take the second step in perturbation theory the driving term will be

$$-\epsilon\omega_0\cos(\omega_p t))x_1\ . \tag{10.12}$$

If you use the trig identity above for this second step in the perturbation theory, you will find that each of the frequency components from the first-order step is multiplied by $\cos\omega_p t$ and that they then produce new sidebands shifted again from the first-order frequencies $\pm\omega_p$. These second-order sidebands are smaller in magnitude than the first-order sidebands because in each step the new driving term is multipled by another factor of $\epsilon$. But they are clearly there in the spectrum. This procedure, of course, never ends, so it is easy to see that this equation can produce a very rich spectrum.

Now, what does this have to do with the observed instability at $\omega_p = 2\omega_0$? Well, as we saw in first-order perturbation theory, when we parametrically oscillate at $\omega_p$ the system looks like a driven oscillator with driving frequencies at $\omega_0 \pm \omega_p$. When $\omega_p = 2\omega_0$, the sum and difference frequencies fall at $3\omega_0$ and $-\omega_0$. Since one of the apparent driving frequencies is at $\omega_0$ (note that $-\omega_0$ is just as resonant as $\omega_0$), the system feeds back on itself and is unstable.

This effect also allows us to see why $\omega_p = \omega_0$ is not the most unstable choice. The reason is that with this choice the sideband frequencies are 0 and $2\omega_0$, neither one of which is resonant. But the second-order sidebands are $-\omega_0$, $\omega_0$, $\omega_0$, and $3\omega_0$. Three out of four come back to $\omega_0$ in second order, so there is a possibility of instability due to this nonlinear resonance. But because it takes two perturbation steps to get to this resonance, and since each step involves another power of $\epsilon$, this choice for $\omega_p$ is less unstable.

**P10.7** Explain all of the frequency peaks in your spectrum from P10.5 using the concepts explained above. Explain what the amplitudes of the various peaks mean and how the $\omega_p = 2\omega_0$ instability arises.

## Parametrically unstable pendulum

When you pump a real swing, your oscillation amplitude doesn't become infinite because the swing is a pendulum, not a harmonic oscillator. Using the Lagrangian formulation of mechanics to obtain the equation of motion of a pendulum whose length $\ell(t)$ is changing with time, and adding some damping because of air friction, gives us the equation

$$\ddot{\theta} + 2\frac{\dot{\ell}}{\ell}\dot{\theta} + \gamma\dot{\theta} + \frac{g}{\ell}\sin\theta = 0 \ . \tag{10.13}$$

If we let the length change sinusoidally at frequency $\omega_p$ by only the small amount $\Delta L$ about the constant length $L_0$, then

$$\ell(t) = L_0 + \Delta L \cos\omega_p t \ . \tag{10.14}$$

**P10.8** Use Matlab to solve Eqs. (10.13) with (10.14) using the following realistic parameter values. A typical backyard swing has a length of about $L_0 = 2$ m. As you do deep knee bends you move most of your mass up and down, so the $\Delta L$ of your parametric oscillation is about half the distance you drop your body during the bend. Use $\gamma = 0.1$ s$^{-1}$ again for the decay. In your numerical solution gradually increase $\Delta L$ from zero up to around 0.3 m with $\omega_p = 2\omega_0$ and find the threshhold value of $\Delta L$ at which the swing becomes unstable. (Explain to your TA why the pendulum amplitude doesn't just keep getting bigger forever.)

# Lab 11

# The Pendulum with a High Frequency Driving Force

Consider[1] an un-driven equation of motion of the form

$$\ddot{x} = -\frac{\partial V}{\partial x} \quad . \tag{11.1}$$

For instance, a harmonic oscillator has $V(x) = kx^2/2m$ and pendulum has $V(x) = -(g/L)\cos x$. Let the characteristic time over which this system changes appreciably be the period $T$, e.g. $T = 2\pi/\sqrt{g/L}$ for the pendulum. We now drive this system with a very high frequency force that depends on both the particle position $x$ and time $t$ so that the equation of motion becomes

$$\ddot{x} = -\frac{\partial V}{\partial x} + A(x)\sin\omega t \, , \tag{11.2}$$

with

$$\omega \gg 2\pi/T \tag{11.3}$$

defining what we mean by high frequency. If we use our intuition (perhaps thinking about what it feels like to drive at high speed over a back-country dirt road that has developed wash boards) we might guess that the motion described by this differential equation would consist of some sort of slowly varying motion on the time scale $T$ plus a high frequency low amplitude vibration at frequency $\omega$. We make this guess precise by writing

$$x(t) = X(t) + \xi(t) \, , \tag{11.4}$$

where $X(t)$ describes the slow motion (think about the car winding its way around curves and over hills) and $\xi(t)$ describes the small amplitude high frequency oscillations (think about stuff in the glove compartment rattling, your teeth chattering, etc.). An example of this kind of motion is shown in Fig. 11.1. The smooth curve is $X(t)$ while the bumpy curve is $x(t) = X(t)+\xi(t)$. The function $\xi(t)$ is the difference between the two curves.

## Perturbation Theory

Because $\xi(t)$ is caused by the sinusoidal driving force, we will see that its time average is zero, and the wide separation of time scales allows us to assume that

[1]This analysis is borrowed from *Mechanics* by Landau and Lifshitz: L. D. Landau and E. M. Lifshitz *Mechanics* (Pergamon Press, New York, 1976), p. 93-95.

$X(t)$ changes only slightly during one period of the high frequency motion. Substituting Eq. (11.4) into Eq. (11.2) and expanding in small $\xi$ through first order gives

$$\ddot{X} + \ddot{\xi} = -\left.\frac{\partial V}{\partial x}\right|_{x=X} - \xi \left.\frac{\partial^2 V}{\partial x^2}\right|_{x=X} + A(X)\sin\omega t + \xi \left.\frac{\partial A}{\partial x}\right|_{x=X} \sin\omega t \,. \tag{11.5}$$

We first attack this equation by looking at the high frequency terms. The term $A\sin\omega t$ is a big term, as is $\ddot{\xi}$ because of its rapid variation in time ($\ddot{\xi} \approx -\omega^2\xi$ with $\omega$ large). All of the other high frequency terms are small compared to these two because $\xi$ is small, so we have (approximately)

$$\ddot{\xi} = A(X)\sin\omega t \,, \tag{11.6}$$

with $X$ approximately constant because it varies so slowly. A simple integration yields the rapidly varying position and velocity

$$\xi(t) = -\frac{A(X)}{\omega^2}\sin\omega t \quad ; \quad \dot{\xi}(t) = -\frac{A(X)}{\omega}\cos\omega t \,. \tag{11.7}$$

We now substitute this result into Eq. (11.5) and time average every term in the equation over one period of the high frequency motion. Terms that contain single powers of $\xi$, $\cos\omega t$, or $\sin\omega t$ average to zero while in the last term, which contains $\sin^2\omega t$, we may replace $\sin^2\omega t$ by its time average of $1/2$ to obtain

$$\ddot{X} = -V'(X) - \frac{A(X)}{2\omega^2}\frac{dA}{dX} \tag{11.8}$$

As you can see, the low frequency motion of the oscillator is altered by the presence of this rapidly oscillating force, provided that the force depends on $X$. This means that a simple high-frequency external force of the form $A\sin\omega t$ with $A$ constant has no effect on the slow motion.

We are not quite finished because we haven't discussed the initial conditions. Suppose that we have initial conditions

$$x(0) = x_0 \quad ; \quad \dot{x}(0) = v_0 \,,$$

Using Eq. (11.4) we have

$$X(0) + \xi(0) = x_0 \quad ; \quad \dot{X}(0) + \dot{\xi}(0) = v_0$$

which can be combined with Eq. (11.7) at $t = 0$ to obtain the proper initial conditions for the slow-motion variable $X$:

$$X(0) = x_0 \quad ; \quad \dot{X}(0) = v_0 + A(x_0)/\omega \,. \tag{11.9}$$

With this choice of initial conditions a combined plot of $x(t)$ and $X(t)$ shows that $x(t)$ wiggles and slowly varies, while $X(t)$ tracks right with it, but with all of the wiggles smoothed out.

## Driven Pendulum

An interesting example of this kind of system is a pendulum whose support point vibrates rapidly up and down like this:

$$y_{\text{support}} = b \sin \omega t\,. \tag{11.10}$$

A simple way to find the new equation of motion of the pendulum is to use Einstein's principle of equivalence between acceleration and gravity: If the support point is accelerating upward with acceleration $a_{\text{support}}$, then the pendulum will experience a downward gravitational force $-ma_{\text{support}}$. Hence we may write for the effective acceleration of gravity acting on the pendulum $a_{\text{support}} = \ddot{y}_{\text{support}} = -\omega^2 b \sin \omega t$ so that the total acceleration, including ordinary gravity is

$$g_{\text{eff}} = g - a_{\text{support}} = g - \ddot{y}_{\text{support}} = g + b\omega^2 \sin \omega t\,, \tag{11.11}$$

which then leads to the equation of motion

$$\ddot{\theta} = -\omega_0^2 \sin\theta - \frac{b\omega^2}{L} \sin\theta \sin\omega t\,, \tag{11.12}$$

where $\omega_0^2 = g/L$. This equation of motion matches Eq. (11.2) if we write

$$A(\theta) = -\frac{b\omega^2}{L} \sin\theta\,, \tag{11.13}$$

which then leads to the following slow time-averaged equation of motion [see Eq. (11.8)]:

$$\ddot{\Theta} = -\omega_0^2 \sin\Theta - \frac{b^2\omega^2}{2L^2} \sin\Theta \cos\Theta\,. \tag{11.14}$$

**P11.1** (a) Use Matlab's `ode45` to solve for the motion of a rapidly driven pendulum by solving both Eq. (11.12) and Eq. (11.14) with $\omega_0 = 1$, $L = 1$, $b = .02$, and $\omega = 30$ with initial conditions $\theta(0) = 1$, $\dot{\theta}(0) = 0$ and run for a total time of 30 seconds. Overlay the plots of $\theta(t)$ from both equations to see that the averaged solution approximates the un-averaged solution.

Now run it again with initial conditions $\theta(0) = 3.1$, $\dot{\theta}(0) = 0$ and check the agreement again.

**Note:** The averaged solution won't go through the middle of the wiggles of the full solution unless you adjust the averaged initial conditions as shown in Eq. (11.9). When you do it right your plot should look like Fig. 11.1.

(b) Now redo part (a) with everything the same except use $b = .05$ this time. You should be surprised, astounded, and amazed at what happens with $\theta(0) = 3.1$. This case is a nearly straight up pendulum, which should fall over, but as you can clearly see, the pendulum is now stable in the straight-up position. This is not a mistake, as you can discover by examining the electric saber-saw demonstration at the front of the room.

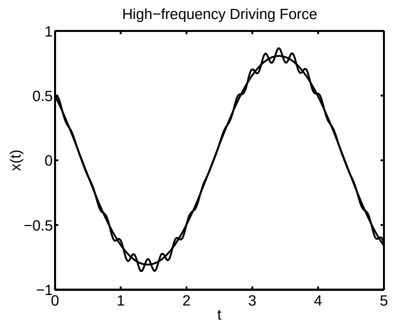


**Figure 11.1**

(c) Analyze this situation more carefully by finding the effective potential that produces the right-hand side of the slow equation of motion, Eq. (11.14), i.e., find $V(\Theta)$ such that

$$-\partial V/\partial\Theta = -\omega_0^2 \sin\Theta - \frac{b^2\omega^2}{2L^2}\sin\Theta\cos\Theta \tag{11.15}$$

(integrate both sides of this equation to obtain $V(\theta)$). Then plot this potential from $\Theta = 0$ to $\Theta = 2\pi$ for various values of $b$ in the range $b = 0$ to $b = .1$ and notice what happens at $\Theta = \pi$ as $b$ increases. Then use calculus to find the critical value of $b$ at which the straight-up pendulum first becomes stable and use your code from part (b) to verify that this threshold value is correct.

These low frequency effective forces that arise from high-frequency non-linear effects are called *ponderomotive forces* and they show up all the time in physical problems. This problem is just a small taste of a very large field.

# Lab 12

# Chaos

## The van der Pol Oscillator

Consider the following non-linear oscillator equation, called the van der Pol oscillator:[1]

$$\ddot{x} - \epsilon(\ell^2 - x^2)\dot{x} + \omega_0^2 x = 0 . \tag{12.1}$$

This is a simple model differential equation for systems that have an external source of energy which causes the resting state ($x = 0,\ v = 0$) to be unstable, but which also have sufficient damping that the instability cannot grow to an arbitrarily large amplitude.

Begin by studying Eq. (12.1) and convincing yourself that the resting state is indeed unstable, but that large amplitude motion is damped (on average). You can't see that $x = 0$, $v = 0$ is unstable by starting the system there and waiting for something to happen, because nothing will happen. This is an equilibrium point and if you start it there it will remain there forever. To test for stability, start the system in a point very close to equilibrium and watch to see if it stays near the equilibrium point, or runs away from it. Appropriate initial conditions to test for stability might be $x = 0.0001$, $v = 0$. The phase-space flow plot, made with `quiver`, in Fig. 12.1 illustrates these two features. Notice the arrows leading away from the origin and the general inward flow at the outer edges of the picture. The flow is not uniformly inward, however, and later in this lab you will see the effect of the squeezed inward flow patterns visible in the figure.

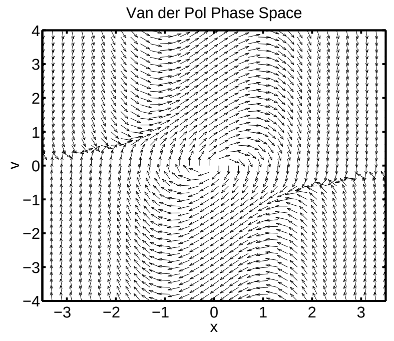


**Figure 12.1** Flow in phase space for the Van der Pol oscillator with $\omega_0 = 1$, $\ell = 1$, and $\epsilon = 1$.

**P12.1** (a) Use Matlab's `ode45` to solve Eq. (12.1) numerically for $\epsilon = 0.3$, $\omega_0 = 1.3$, and $\ell = 1$. Use `options=odeset('RelTol',1e-5)` to set the accuracy of `ode45` at a level that will make it possible to do long runs in a reasonable time. (We would normally use a smaller tolerance than this, but we only have 3 hours together). Make both a plot of $x$ vs. $t$ as well as a phase space plot of $v$ vs. $x$ for a bunch of different initial conditions. Notice that the phase space plot eventually settles on the same curve for any initial conditions you pick. The phase space curve on which the solutions settle is called a *limit cycle*

(b) Repeat part (a) for $\epsilon = 1$ and $\epsilon = 20$ and note how the limit cycle changes shape. Also plot the power spectrum of $x(t)$ with `semilogy` using an axis command to display the spectrum from $\omega = 0$ to $\omega = 20$ and note where the major peaks are. Remember to interpolate $x(t)$ onto an even time grid before computing the Fourier transform.

[1] R. Baierlein, *Newtonian Dynamics* (McGraw Hill, New York, 1983), p. 88-93.

## Limit Cycles and Attractors

The limit cycle you observed in this problem is a simple example of an *attractor* in phase space. An attractor is a curve in the phase-space of the differential equation to which many different solutions (having different initial conditions) tend. For instance, for the damped un-driven harmonic oscillator the attractor is just the state of no motion: $x = 0$, $v = 0$, because all solutions end up here. For the driven damped harmonic oscillator the attractor is more interesting: it is the final driven steady state of the oscillator, which looks like an ellipse in phase space. Since this attractor is not a single point, we also call it a limit-cycle. For the van Der Pol equation the attractor is the oddly-shaped curve (or limit-cycle) in the $(x, v)$ phase space to which all solutions tend.

Sometimes an attractor is not a single curve, but rather a very complex structure, like the famous Lorenz attractor (which you can explore a little bit by typing `lorenz` at the command prompt in Matlab). These kind of attractors are called *strange attractors*, and are examples of chaotic systems. We'll study chaos later in this lab and you will see other examples of attractors, but none of the attractors encountered in this lab are strange attractors (except the Lorenz attractor).

**P12.2** Now let's add a driving force to the van der Pol oscillator, like this:

$$\ddot{x} - \epsilon(\ell^2 - x^2)\dot{x} + \omega_0^2 x = A\cos\omega t\ . \tag{12.2}$$

Using $\ell = 1$, $\epsilon = 2$, $\omega_0 = 1.3$, and $\omega = 1.4$, gradually increase $A$ from 0 to 1.5 and watch what happens to the power spectrum of $x(t)$. Change $A$ by steps large enough to see qualitative changes, i.e., don't do $A = 0.01$, $A = 0.02$, $A = 0.03$, etc.

You should find that as $A$ is increased the limit cycle becomes fuzzy and that the power spectrum becomes increasingly filled with spikes. Finally, around $A = 1.25 \rightarrow 1.27$ the power spectrum becomes so complicated that it is fuzzy too (use the zoom feature on the spectrum to see that the spectrum is made up of many tiny peaks). And then, quite abruptly, at about $A = 1.28$ the oscillator becomes slaved to the drive, meaning that the oscillator vibrates at the driving frequency $\omega = 1.4$ and its harmonics, making the spectrum simple again. (Look carefully at the power spectrum to see that this is true).

## Entrainment

The kind of behavior illustrated in P12.2 is called *entrainment*, in which an oscillator becomes synchronized to another periodic signal. An important example of a system like this is the human heart. The heart has an external source of power, has an unstable resting state (it wants to beat rather than sit still), and, normally, a stable limit cycle (thump-Thump, thump-Thump,...). Sometimes this stable limit cycle becomes irregular, in which case it is desirable to supply a periodic driving signal via a pacemaker which, if strong enough, can force the heart to become entrained with it, restoring a stable limit cycle, albeit at a frequency determined by the pacemaker rather than by the physical needs of the patient.

## Dynamical Chaos

Now let's switch gears a bit and take a brief tour through one of the most exciting areas in the study of differential equations: *dynamical chaos*. A chaotic system is one where dynamical variables (e.g. position and velocity) behave in seemingly erratic ways and exhibit extreme sensitivity to initial conditions. Chaotic systems are deterministic, since a given set of parameters and initial conditions reproduce the same motion, but it is usually difficult to predict how tiny variations in parameters or initial conditions will affect the motion.

Chaotic systems have been known and studied for a long time. For instance, it comes as no surprise that when you have $10^{23}$ atoms bouncing around inside a container, hitting the walls and hitting each other, that the motion of any given atom is pretty chaotic. But in the middle of the twentieth century it was discovered that even simple systems can be chaotic. For instance, here is the apparently nice, smooth, and well-behaved differential equation for the driven damped pendulum:

$$\frac{d^2\theta}{dt^2} + \gamma\dot{\theta} + \omega_0^2 \sin\theta = A\cos\omega t\,. \tag{12.3}$$

This system has only two degrees of freedom (way less than $10^{23}$) and all of the functions that appear in it are nice and smooth. But for certain choices of $A$, $\omega$, $\omega_0$, and $\gamma$ the solutions of this differential equation are almost as unpredictable as the motion of an atom in a gas.

Chaos is hard to study because that old standby of physical theory, the formula, is not of much help. If we had a formula for the solution of this differential equation its behavior would be perfectly predictable and un-chaotic. Since the dynamics in chaotic systems are not represented by analytic formulas, their solution had to wait for computers to be invented and to become powerful. The computers we will use in this laboratory are more powerful than the computers we used to send men to the moon and to design nuclear weapons in the 1960s and 1970s, so we have all the computing power we need to at least be introduced to this fascinating field.

**P12.3** A simple system in which chaos can be observed is a particle moving in a potential well with two low spots:

$$U(x) = -\frac{x^2}{2} + \frac{x^4}{4}\,. \tag{12.4}$$

(a) Plot this potential vs. $x$ and locate the two stable equilibrium points (the one in the middle is unstable).

(b) Let a particle have mass $m = 1$ and use the force relation

$$F_x = -\frac{\partial U}{\partial x} \tag{12.5}$$

to derive the equation of motion of the particle. Then write a Matlab script and a function that employs `ode45` to solve for the motion of the particle. Use `options=odeset('RelTol',1e-6)` to set the accuracy of `ode45`. Try several different initial conditions and watch how the particle behaves in this double well. Look at the motion in phase space for enough different initial conditions that you can see the transition from motion in one well or the other to motion that travels back and forth between the wells.

(c) Now add a driving force of the form $F = A\cos 2t$ and also include a linear damping force $F_{damp} = -m\gamma\dot{x}$ with $\gamma = 0.4$. Use initial conditions $x(0) = 1$, $v(0) = 0$, and make a series of runs with $A$ gradually increasing until you observe chaotic behavior. (The transition from regular motion to chaos occurs between $A = 0.7$ and $A = 0.8$). Run from $t = 0$ to $t = 1000$. A plot of $x(t)$ should show random jumping between the left and right sides of the double well, as illustrated in Fig. 12.2. For each run make a plot of the power spectrum of $x(t)$. Show the TA how your plots illustrate intermittency and $1/f$ noise (described below).

(d) With $A = 0.9$ do two runs, one with initial conditions $x(0) = 1$, $v(0) = 0$, and the other with $x(0) = 1.000001$ and $v(0) = 0$. Plot $x(t)$ for each of these cases, and explain to the TA how these plots illustrate the butterfly effect (described below).

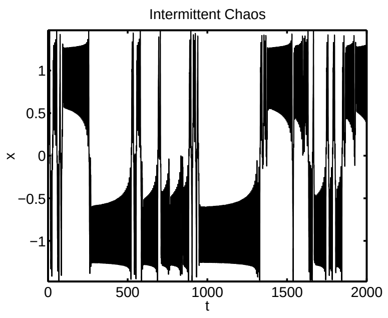


**Figure 12.2** Intermittent random bouncing between the two wells.

## Intermittency, $1/f$ Noise, and the Butterfly Effect

The random switching back and forth between equilibrium positions observed in P12.3(c) is called *intermittency* and is one of standard ways that regular systems become chaotic. As the motion becomes chaotic you should also see an increase in the spectrum near $\omega = 0$. This low frequency peak in the spectrum is one of the symptoms of chaos (called "$1/f$ noise") and is a direct consequence of the slow random switching of intermittency.

Another hallmark of chaotic systems is the so-called "butterfly effect" (illustrated in P12.3(d)), where very small changes in the initial conditions cause large differences in the motion. This effect was discovered by Edward Lorenz (for whom the Lorenz attractor is named), who was a meteorologist that studied numerical models for weather prediction in the early 1960s. He noticed that very tiny differences in initial conditions (too small to even be measured) led to vastly different outcomes in his model. The effect gets its name from a talk that he gave in 1972 titled "Predictability: Does the Flap of a Butterfly's Wings in Brazil set off a Tornado in Texas?".

**P12.4** (a) Make a phase space plot for the system in P12.3(c) with $A = 0.96$. You should find that the chaotic behavior quiets down and is replaced by a limit cycle in phase space. It will be difficult to see the limit cycle on the phase space plot because of the messy transients at the beginning. To eliminate the transients make the phase space plot like this (We chose to skip the first 60%–you can try your own value):

```
N=length(x);
n1=ceil(.6*N); % n1 starts 60% into the array
plot(x(n1:N),v(n1:N));
```

(b) Make another phase space plot at $A = 1.30$. The single limit cycle should be replaced by a 2-cycle (two loops in phase space before repeating);

**Note:** to really see the multiple character of these cycles, use the zoom feature in the plot window to look carefully at them, especially near the tight loops. This is shown in Fig. 12.3 for the 4-cycle state. In the upper window the full time history is shown from the beginning while in the lower window the late-time final state in the window from the upper frame is shown. There are clearly 4 repeated loops in phase space, so this is called a 4-cycle. This is an example of the famous "period-doubling route" to chaos, as well as an example of regular behavior in a region of parameter space where you might have expected chaos.

(c) Make phase space plots of the limit cycle at $A = 1.36$ (a 4-cycle state), which is then replaced by an 8-cycle at $A = 1.371$, and then chaos takes over again.

If you run with $A = 1.97$, 1.99, and 2.0, you will see chaos disappear to be replaced by a 2-cycle, a 4-cycle, and an 8-cycle. Beyond 2 there is chaos again.

At $A = 3$ the amplitude is large enough that the oscillator becomes slaved to the drive and we have entrainment. You might think that large $A$ would always cause entrainment, but $A = 50$ is chaotic, and there are probably lots of 2,4,8,... cycles and chaotic regions as $A$ is varied. We ran out of patience; let us know what you find.

**Note:** when you run with $A = 1.36$ your phase-space picture may look like an upside-down left-right flipped version of Fig. 12.3. This is OK–the differential equation is almost unchanged if $(x, v)$ is replaced with $(-x, -v)$. The only difference is that the driving term is replaced by its negative, which is equivalent to a phase shift of $\pi$. Such a phase shift could occur by having the oscillator start up in a different way, which might easily happen if your initial conditions were not exactly the same as ours. This flipped-over state is to be expected on physical grounds. Our picture has tight loops on the left and big loops on the right, but the potential is left-right symmetric; there should be another state with tight loops on the right and big ones on the right as well.

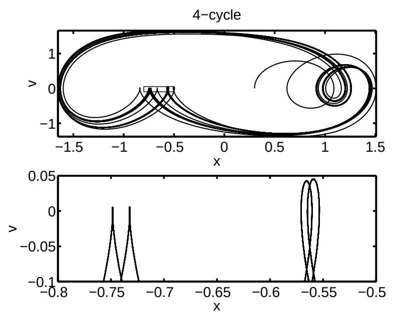


**Figure 12.3** Full time history, then final 4-cycle state from the small window. There are fewer loops in the lower trace because it is the final state; the extra loops in the box in the upper trace are from early times.

## Fractals

**P12.5** (a) The Fibonacci sequence $F_n$ is defined by

$$F_1 = 1 \quad ; \quad F_2 = 1 \quad ; \quad F_n = F_{n-1} + F_{n-2} \text{ for } n \geq 3 \tag{12.6}$$

(The first few numbers in the sequence are $1, 1, 2, 3, 5, 8, 13, \ldots$). Write a loop that fills the array `Fn` with the first 100 values of the Fibonacci sequence.

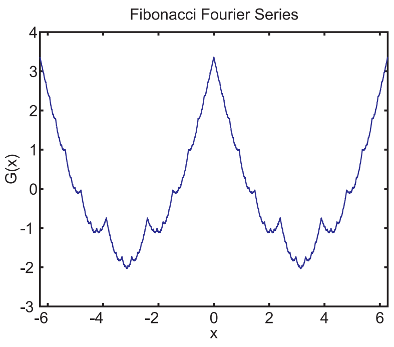


**Figure 12.4** The function plotted in P12.5

(b) Now define an array `x` that goes from $-2\pi$ to $2\pi$ with 50,001 equally spaced values, like this:

```
h=4*pi/50000;
x=-2*pi:h:2*pi;
```

Then write a loop that evaluates the Fourier-like series

$$G(x) = \sum_{n=1}^{100} \frac{\cos(F_n x)}{F_n} \, . \tag{12.7}$$

Plot this function vs. $x$ and carefully observe its shape (this function is shown in Fig. 12.4). Then use the zoom feature to more closely examine some of the smaller mountain peaks to discover that each mountain peak contains smaller versions of itself.

If you zoom in too much you will run out of points, so now plot the function again using 50,001 points between $x = 3.1$ and $x = 3.2$, and zoom in again. This kind of curve is called a *fractal*, or *fractal curve*[2] and such curves are important in chaos theory.

[2]S. N. Rasband, *Chaotic Dynamics of Nonlinear Systems* (John Wiley and Sons, New York, 1990), Chap. 4, and http://sprott.physics.wisc.edu/fractals.htm

# Lab 13

# Coupled Nonlinear Oscillators

When two or more oscillators are hooked together, we say that they are coupled. [1] Our final model for driving a swing was an example of coupled pendula. In that case the rotation of the swinger was coupled to the rotation of the overall swing, which allows you to drive the swing.

In this lab we consider a different method for coupling pendula. Consider two pendula hanging from the same piece of horizontally-stretched rubber tubing. If one pendulum is held fixed and the second is displaced from equilibrium, the second one experiences a restoring torque from two separate sources: (a) gravity and (b) the rubber tubing. If both pendulums are displaced together each one experiences gravity and restoring torque from the tubing, but the tubing between the two plays no role because they both twist it in the same direction. But if the pendulums are displaced in opposite directions then the tubing between them is flexed, causing an extra restoring torque. This difference in restoring force between the "together" and "opposite" motions is the cause of the two slightly different frequencies that produce the beating you will see throughout this lab.

## Coupled Equations of Motion via Lagrangian Dynamics

For small displacements, the effect of gravity (plus a small contribution from the tubing) can be modeled as restoring torsional springs with spring constants $\kappa_1$ and $\kappa_2$ for pendulum 1 and pendulum 2, respectively. This leads to a potential energy

$$U = \frac{1}{2}\kappa_1\theta_1^2 + \frac{1}{2}\kappa_2\theta_2^2 \,. \tag{13.1}$$

The tubing also adds a coupling term to the potential energy of the form

$$U_c = \frac{1}{2}\kappa_c(\theta_1 - \theta_2)^2 \tag{13.2}$$

Note that this extra restoring potential energy is zero if the angular displacements are equal. The total potential energy is

$$U = \frac{1}{2}\kappa_1\theta_1^2 + \frac{1}{2}\kappa_2\theta_2^2 + \frac{1}{2}\kappa_c(\theta_1 - \theta_2)^2 \,. \tag{13.3}$$

**P13.1** Combine this potential energy function with the kinetic energy

$$T = \frac{1}{2}I_1\dot{\theta}_1^{\,2} + \frac{1}{2}I_2\dot{\theta}_2^{\,2} \tag{13.4}$$

[1] G. Fowles and G. Cassiday, *Analytical Mechanics* (Saunders, Fort Worth, 1999), p. 443-460.

to build the Lagrangian ($L = T - U$). Use the Lagrangian equation of motion

$$\frac{\partial L}{\partial q_i} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) = 0 \tag{13.5}$$

to derive equations of motion for $\theta_1(t)$ and $\theta_2(t)$. Then simplify these equations by assuming that the two pendula are identical so that $\kappa_1 = \kappa_2 = \kappa$ and $I_1 = I_2 = I$. Also eliminate the spring constants and moments of inertia in favor of frequencies according to the definitions

$$\omega_0^2 = \frac{\kappa}{I} \quad ; \quad \omega_c^2 = \frac{\kappa_c}{I} \quad . \tag{13.6}$$

Finally, put these two second-order differential equations in coupled first order form.

If you did P13.1 correctly, you should have arrived at the following four first-order differential equations for describing the motion of the coupled-pendulum system:

$$\dot{\theta}_1 = \omega_1 \tag{13.7}$$
$$\dot{\theta}_2 = \omega_2 \tag{13.8}$$
$$\dot{\omega}_1 = -\omega_0^2\theta_1 - \omega_c^2(\theta_1 - \theta_2) \tag{13.9}$$
$$\dot{\omega}_2 = -\omega_0^2\theta_2 - \omega_c^2(\theta_2 - \theta_1). \tag{13.10}$$

The four variables are: the angular positions of the two pendulums $\theta_1(t)$ and $\theta_2(t)$ (remember that $\theta = 0$ corresponds to the pendulum hanging straight down) and, the angular velocities of the two pendulums $\omega_1(t)$ and $\omega_2(t)$. The parameter $\omega_0$ is associated with the natural frequency of a single pendulum without any coupling, and the parameter $\omega_c$ is associated with the natural frequency of the middle section of tubing when attached to the two pendula.

**P13.2** (a) Use Mathematica to solve Eqs. (13.7)-(13.10) symbolically without initial conditions to see if you can find the two separate frequencies that cause the beating you will see when you solve them in Matlab.

(b) Now numerically solve this system in Matlab with $\omega_0 = 1.3$ and $\omega_c = 0.3$; for initial conditions let everything be zero except $\theta_1(0) = 0.3$. Make plots of $\theta_1(t)$ and $\theta_2(t)$, one above the other using `subplot`[2] like this:

```
subplot(2,1,1)
plot(te,th1e)
subplot(2,1,2)
plot(te,th2e)
```

Run long enough that something interesting happens, i.e., run at least long enough that $\theta_2(t)$ becomes large, then small again. You should

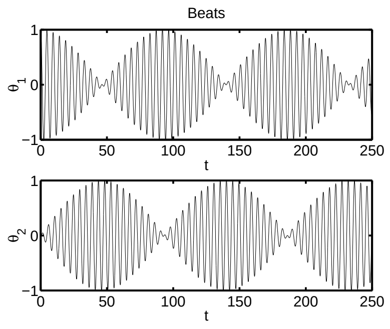


**Figure 13.1** Energy passes back and forth between $\theta_1$ and $\theta_2$ due to beating.

[2]For more information about subplots, look it up via `help subplot` using online help in Matlab.

be looking at the beat plot in Fig. 13.1. This pattern of increasing and decreasing amplitude in the plots of $\theta_1(t)$ and $\theta_2(t)$ is an example of interference beats caused by the presence of two different frequencies in the dynamics.

(c) Run the solution out for long enough that when you take the FFT of $\theta_1(t)$ you can see the two peaks in the power spectrum corresponding to the two frequencies whose mixing causes the beats. Verify that the beat frequency $\omega_b = 2\pi/T_b$, (where $T_b$ is the time for one of the oscillators to be at maximum amplitude, go to zero amplitude, then come back to maximum amplitude again) is related to the two peaks in the spectrum $\omega_+$ and $\omega_-$ by

$$\omega_b = \omega_+ - \omega_- \,. \tag{13.11}$$

Also verify that the two frequencies you observe in the FFT are the two frequencies predicted by your Mathematica calculation.

**Note:** the figure at the beginning of this lab does not have enough oscillations in it for the FFT to work well. As a general rule, your time plots should look solid if you want to use the FFT. A maximum time around 2000 works fine.

(d) Now add a linear damping term to the equation of motion for pendulum number 2, start the system with these initial conditions: $\theta_1(0) = 0.3$, $\dot\theta_1(0) = 0$, $\theta_2(0) = 0$, $\dot\theta_2(0) = 0$. and study the motion of the two oscillators. Use

$$\ddot\theta_2 = -\gamma\dot\theta_2 + \cdots \tag{13.12}$$

with $\gamma = 0.07$. Look at the plots for $\theta_1(t)$ and $\theta_2(t)$ and discuss what happens to the energy that was initially put into pendulum number 1.

(e) Now drive pendulum number 1 by applying a small negative torque $N_1 = -0.3$ whenever $\theta_1$ is positive and $\dot\theta_1$ is negative. You will need to use an `if` statement in the M-file that defines the right-hand side of your set of differential equations to make this work. This driving force is similar to the escapement in a pendulum clock in which a mechanical linkage allows the weights to push on the pendulum when it is at the proper place in its motion. Think about this drive and verify that it always puts energy into pendulum number 1.

☞ This driving force is essentially the same as the intermittent torque you apply to someone when you push them in a swing.

As in part (b), don't damp pendulum number 1; just keep the damping in pendulum number 2. Run the code long enough that the system comes to a steady state in which both pendulums have constant amplitude. Discuss the flow of energy in this system.

## Coupled Wall Clocks

Now we are ready to study a very famous problem in dynamics. In the 1600s Christian Huygens observed that when two clocks are hung next to each other on a wall, they tend to synchronize with each other. Let's see if we can make our equations of motion do this.

**P13.3** (a) Begin by making *both* pendulums be damped and driven as described in P13.2(d) and (e), but remove the coupling by setting $\omega_c = 0$. Run the code and make sure that each clock comes to its own independent steady state.

Now add weak coupling between the two by setting $\omega_c = 0.3$ again (the slight pushes and pulls that each clock exerts on the wall is the source of this coupling) and see if the clocks ever synchronize with each other. (Synchronization means that the two pendulums have the same period with some definite phase shift between them. This effect is called *entrainment* in the nonlinear dynamics literature).

When you do these runs, start pendulum 1 with $\theta_1 = 1$ and $\dot{\theta_1} = 0$ and try various choices for the initial conditions of pendulum number 2. When they become entrained, check the phase difference between the two clocks. (A visual inspection is probably sufficient). In your numerical experiments, how many different phase relationships do you observe? (Try overlaid plots of $\theta_1$ and $\theta_2$ to see the phase relationships). Do your in-phase and out-of-phase entrained states have the same frequencies?

(b) According to the nonlinear dynamics literature, entrainment is an effect that depends on the oscillators being damped, driven, and nonlinear. Where is the nonlinearity in our equations of motion?

(c) Finally, let's make these clocks a little more realistic by (i) replacing $-\omega^2\theta$ by $-\omega^2 \sin\theta$ in each equation of motion and by (ii) having their natural frequencies be slightly different. Do this by changing $\omega^2$ in the equation of motion for pendulum 2 to $1.03\omega^2$, $1.1\omega^2$, and $1.25\omega^2$ (do all three cases). You should find that entrainment is relatively robust, meaning that the clocks don't have to have exactly the same period to synchronize, but that if they are too different the effect is lost. Does this robustness depend on the strength of the coupling parameter $\omega_c$? Comment on what your answer to this last question has to do with real clocks on a wall.

## Solving Nonlinear Equations

**P13.4** Read and execute the examples in *Introduction to Matlab*, Chapter 14. Then complete the following exercises.

(a) Write a loop that makes an array containing the first 40 zeros of the Bessel Function $J_0(x)$. Find these zeros by writing a loop to load them using Matlab's `fzero` command. You will have to give `fzero` a search range instead of just an initial guess, and this will be easier if you remember that the zeroes of $J_0(x)$ are separated by about $\pi$.

Look through your list of zeros and make sure that there are no repeated values. Then plot $J_0(x)$ and put a red x at every zero.

(b) Solve the following set of equations using Matlab's `fsolve` command

$$\begin{aligned} x^2+y^2+z^2 &= 139 \\ \frac{x}{x+y-z} &= 3 \\ x\sqrt{z} &= (10-y)^2 \end{aligned}$$

# Index