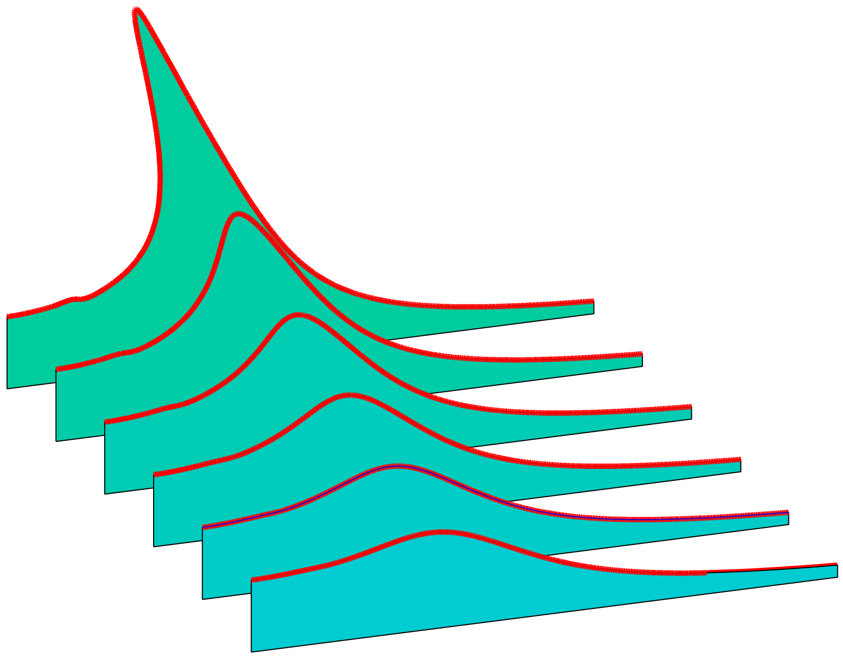
Computational Physics 330

Non-Linear Dynamics and Differential Equations

using Mathematica and Matlab



Ross L. Spencer and Michael Ware

Department of Physics and Astronomy
Brigham Young University

Computational Physics 330

Non-Linear Dynamics and Differential Equations

using Mathematica and Matlab

Ross L. Spencer and Michael Ware

Department of Physics and Astronomy
Brigham Young University

Last revised: June 22, 2012

Our objective in this course is to learn how to use a symbolic mathematics program and programming in Matlab to analyze physics problems in terms of ordinary differential equations and solve them numerically. The instructor and a 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 familiar with Mathematica from the start so that our study of differential equations can begin in this language. Initially the labs consist of Mathematica exercises involving differential equations and assignments to work through sections of the text *Introduction to Matlab*. Later we use both Matlab and Mathematica to 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

Mathematica, Baseball, and Matlab

Differential equations are the language of physics. To this point, you have probably focused on systems where the differential equations can be solved analytically to obtain an explicit formula describing the dynamics. However, there is literally a world of interesting systems for which it is not possible to obtain simple formulas for the dynamics. In this course we use numerical methods to explore such systems.

Differential Equations in Mathematica

Mathematica has some excellent differential equation solvers built into it, both analytic and numerical.

P1.1 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).

P1.2 Use Mathematica to solve the differential equation governing the current $i(t)$ in a circuit containing a battery of emf $\mathscr{E}$, resistance R, and inductance L is[1]

$$L\left(\frac{d}{dt}i(t)\right)+Ri(t)=\mathscr{E} \tag{1.1}$$

Use Mathematica to solve this differential equation for the current and plot the result if the initial current is zero, $L = 0.001$ H, $R = 100\ \Omega$, and $\mathscr{E} = 6$ V.

HINT: When choosing the plot range, look at the solution and notice that the time constant for an RL circuit is $\tau = L/R$.

P1.3 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

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

[1] Perhaps you are wondering why we aren't using $I(t)$ for the current. Recall that I is the imaginary number ($\sqrt{-1}$) in Mathematica, so we don't want to use this symbol for anything else.

P1.4 Read the section titled "Numerical solutions to ordinary differential equations" in the Mathematica tutorial *Differential equations with Mathematica.*

(a) Determine what a rocket's initial velocity would need to be if launched vertically away from the earth's surface ($z_0 = 6.4 \times 10^6$ m) for it to just reach the moon at $z = 3.8 \times 10^8$ m before falling back to earth. How long would the rocket take to get to the moon?

HINT: The escape velocity from the earth's surface is about 11,200 m/s, so your velocity will be less than this. Also, if you let time run too long, `NDSolve` will break because the differential equation has a zero in the denominator. Just run time out long enough to get the projectile to the apex of its flight.

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

$$\frac{d^2}{dx^2}y(x) = \sin[\pi y(x)/x] \tag{1.2}$$

Now write the equation as a first order set, and solve it numerically with $y(1) = 0$ and $v(1) \equiv y'(1) = 0.01$. Plot $y(x)$ from $x = \pi$ to $x = 100$.

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[2] on a baseball is approximately given by the following formula

$$\mathbf{F}_{\text{drag}} = -\frac{1}{2}C_d\rho_{\text{air}}\pi a^2|v|\mathbf{v} \tag{1.3}$$

where C_d is the drag coefficient, ρ_{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. (1.3) 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.

Newton's second law now provides us with the equation of motion for the ball

$$m\ddot{\mathbf{r}} = \mathbf{F}_{\text{drag}} - mg\hat{\mathbf{y}} \tag{1.4}$$

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 a vector equation, it represents a whole system of equations—one for each

[2]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.

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. (1.4) into the following set of four coupled first-order equations

$$\begin{aligned} \frac{\partial x}{\partial t} &= v_x & \frac{\partial v_x}{\partial t} &= -\frac{C_d \rho_{\text{air}} \pi a^2 v_x \sqrt{v_x^2 + v_y^2}}{2m} \\ \frac{\partial y}{\partial t} &= v_y & \frac{\partial v_y}{\partial t} &= -\frac{C_d \rho_{\text{air}} \pi a^2 v_y \sqrt{v_x^2 + v_y^2}}{2m} - g \end{aligned} \tag{1.5}$$

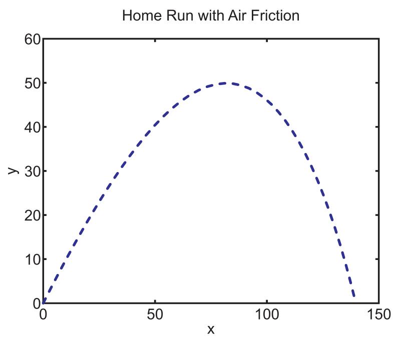


Figure 1.1 The trajectory for a home run hit, including the effect of air friction. Note that the path is not a parabola.

P1.5 (a) Use Mathematica to solve the set of equations (1.5) for a baseball with the following parameters:

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

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 θ 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. $x(t)$ using `ParametricPlot`.

HINT: If you get your numeric solutions for $x(t)$ and $y(t)$ out of the solution list like this

```
xpos = x[t] /. sols
ypos = y[t] /. sols
```

then you'll need to `Flatten` them to get a properly nested list for `ParametricPlot`, like this:

```
ParametricPlot[Flatten[{xpos, ypos}],{t,0,7}]
```

You can use `PlotRange` to set the plot limits of the `ParametricPlot` if you like.

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

(b) 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 10% lower than it is at sea level.

(c) Set your initial speed to 47 m/s (105 mph) and vary the angle to find the maximum range you can get from your sea-level model.

Physicists studying baseball say that backspin (which makes the ball float by deflecting air downward through the Bernoulli effect) is essential for record hits. One expert says that an initial speed of 47 m/s with optimal backspin gives a range of about 122 m (400 feet). Compare this value to the range you found with Mathematica (which doesn't include backspin). The amount you fell short shows the importance of backspin.

(d) Finally, to really see that the trajectory is not a parabola, use a very large initial velocity and observe that the ball finally comes down almost vertically. Explain why this is so by contrasting the x and y forces felt by the ball during flight.

Learning Matlab: Basic Functionality

As we've just seen, Mathematica is a great tool for solving physics problems. It provides about every symbolic and numerical appliance you can imagine, and they are always available at your fingertips. But this flexibility comes with a price: speed. The default settings in Mathematica can bog the computer down and make numerical models run unacceptably slow. If you work hard and program Mathematica correctly, you can work around these issues, but it can be painful. Its good for you to be familiar with a variety of numerical tools, so we'll now start to expand your horizons a bit.

Throughout the remainder of this course we'll learn how to use Matlab. Matlab has generally good speed,[3] has been streamlined for ease of use, and comes with a large array of built-in visualization tools. You will need the skills acquired in the "Learning Matlab" reading sections to do problems in later labs, so don't just skim the reading to get just enough information to do the exercise.

P1.6 Read and work through *Introduction to Matlab*, Chapters 1-2. Type and execute all of the material in `typewriter font`.

(a) Execute the following command at the Matlab command prompt

```
tic;sum(1:1e7);toc
```

to measure the time it takes Matlab to allocate an array of integers from 1 to 10,000,000 and then sum them. Then execute the analogous command in Mathematica

```
AbsoluteTiming[Total[Table[n, {n, 1, 10^7, 1}]]]
```

[3]Fortran or variants of c are more commonly used by physicists when raw number-crunching speed is essential. If you have some significant modeling to do, you may want to check them out, but it takes longer to learn these languages than we'll have together in this class. For the most common tasks, its usually better to have the conveniences of Matlab or Mathematica.

to see how the two platforms compare in number-crunching speed. On our computer, Matlab was about 30 times faster (0.1 s in Matlab versus 3.5 s in Mathematica). This speed difference can become more pronounced in more complicated numerical analysis.

NOTE: If you are careful about what data types and operations you let Mathematica use, you can often make Mathematica about the same speed as Matlab. But by default, Mathematica is often slower because it works hard to keep everything as general as possible.

(b) Write a Matlab program (i.e. an M-file script) that defines the Matlab matrices

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

and

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

Also define the row vector

```
v1=[1,1,2]
```

and the column vector

```
v2=[0.40824829 ; -0.81649658 ; 0.40824829]
```

Then add lines of code that perform the following operations:

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

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

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

(c) Write a script that accepts the argument x and prints the values of $\sinh x$, $\cosh x$, and $\tanh x$, all three properly labeled and on the same output line. Display 7 decimal places in scientific notation, i.e., `3.1415927e+00`.

Lab 2

Qualitative Analysis and Matlab

It is important for a physicist to have an intuitive feel for what differential equations mean rather than just memorizing a bunch of techniques and formulas. Without this feel you won't be able to propose and refine mathematical models for physical processes. In this lab, we work on training your intuition so that you can understand a differential equation without having to plot its solution with a computer.

How does a differential equation make a curve?

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

$$\frac{\partial}{\partial t}y = y \tag{2.1}$$

Since $\frac{\partial}{\partial t}y$ is the slope of the function $y(t)$ this 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 $-e^t$.

Let's 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,

$$\frac{\partial}{\partial t}y = y^p \tag{2.2}$$

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.

P2.1 Use Mathematica to solve Eq. (2.2) for $p = 2$ and $p = 3$ with the initial condition $y(0) = 1$. Then plot the solution over a big enough time interval that you can see the explosion.

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

$$\frac{\partial^2}{\partial t^2} y = -y \tag{2.3}$$

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.

Ok, what does Eq. (2.3) say if $y = 1$ to start? Well, the second derivative is negative, so the function y 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 $\sin(t)$.

P2.2 For each of the following cases, use qualitative analysis to sketch the solution of the equation on paper. Don't peek at the answer by having Mathematica solve them first.

(a)
$$\frac{\partial}{\partial t} y = y^2 \quad \text{with} \quad y(0) < 0$$

(b)
$$\frac{\partial^2}{\partial t^2} y = y \quad \text{with} \quad y(0) = 1 \quad \text{and} \quad \frac{\partial}{\partial t} y(0) = 0$$

(c)
$$\frac{\partial^2}{\partial t^2} y = -y^2 \quad \text{with} \quad y(0) = 1 \quad \text{and} \quad \frac{\partial}{\partial t} y(0) = 0$$

(d)
$$\frac{\partial^2}{\partial t^2} y = -y^3 \quad \text{with} \quad y(0) = 1 \quad \text{and} \quad \frac{\partial}{\partial t} y(0) = 0$$

P2.3 After making all of your sketches in P2.2, check your answer by plotting the solutions for each equation using Mathematica. The solutions for (c) and (d) will need to be found numerically.

Learning Matlab: Loops, Logic, and Plotting

P2.4 Read and work through *Introduction to Matlab*, Chapters 3-4. Type and execute all of the material in `typewriter font`.

P2.5 Write a 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 the 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.

☞ 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.

P2.6 Read and work through *Introduction to Matlab*, Chapter 5. Type and execute all of the material in `typewriter font`. Then complete the following exercises.

(a) Make a graph of the Bessel function $J_0(x)$ from $x = 0$ to $x = 50$. Label the axes and give the plot a title. Then overlay on the same frame a plot of $J_1(x)$ and add a legend to the plot that identifies each curve. You will need to use `help legend` to learn how to do this.

(b) Write a 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 every line of your code while you watch the values change in the workspace window.

Plot the `a` array with `semilogy` and overlay plots of e^{-n} and $1/n!$ and label each line with a legend. Which function matches the way the terms fall off with n?

(c) Define an array of x-values like this: `x=0:.01:5`. Then make an array `f` the same size as array `x` using the `zeros` command, and use loop and logic commands to load `f` with the values:

$$f(x) = \begin{cases} e^x & , \quad 0 \le x < 1 \\ e \times \cos(x-1) & , \quad 1 \le x \le 5 \end{cases} \quad , \tag{2.4}$$

Plot $f(x)$ vs. x and label your axes.

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

```
    if x < 1
```

because `x` is an array. You will need to address individual elements of the arrays inside your loop.

Lab 3

The Harmonic Oscillator

The harmonic oscillator is probably the most studied system in dynamics. In this lab we explore some of the behavior of this system.[1]

The Basic Oscillator

The basic oscillator equation is given by

$$\frac{d^2}{dt^2}y(t) = -\omega_0^2 y(t). \tag{3.1}$$

The solutions to this equation are just sines and cosines that wiggle forever in time at the natural frequency ω_0:

$$y(t) = A\sin(\omega_0 t) + B\cos(\omega_0 t) \tag{3.2}$$

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

The Damped Oscillator

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

$$\frac{d^2}{dt^2}y(t) = -\omega_0^2 y(t) - 2\gamma\frac{d}{dt}y(t), \tag{3.3}$$

where the damping factor γ describes the amount of damping—a large γ means that there is a lot of damping. If you ask Mathematica to find the general solution to Eq. (3.3), it will tell you that the solution is

$$y(t) = Ae^{-t\left(\gamma+\sqrt{\gamma^2-\omega_0^2}\right)} + Be^{-t\left(\gamma-\sqrt{\gamma^2-\omega_0^2}\right)} \tag{3.4}$$

Equation (3.4) 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 ω_0 and γ 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 γ. 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

[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.

vertical position and stay there. We refer to this system as being *overdamped.* Look at Eq. (3.4) and convince your lab partner that it is just a decaying exponentials when γ 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? To see, let's work through the algebra explicitly:

P3.1 Use paper and pencil for this problem. Mathematica will just slow you down.

(a) Assume that the argument of the square root in Eq. (3.4) is negative. Then factor out the imaginary number ($i = \sqrt{-1}$) from the square root and show that Eq. (3.4) can be written as

$$y(t) = e^{-t\gamma}\left[Ae^{-i\omega_d t} + Be^{i\omega_d t}\right] \qquad \text{(when } \gamma < \omega_0\text{)} \tag{3.5}$$

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

$$\omega_d = \omega_0\sqrt{1-\gamma^2/\omega_0^2} \tag{3.6}$$

☞ 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.

Then use Euler's formula $e^{i\theta} = \cos(\theta) + i\sin(\theta)$ to write Eq. (3.5) as an exponential decay multiplying some sinusoids.

(b) You will have some imaginary numbers in your solution, which is bothersome since the displacement $y(t)$ is a real number. Fix this issue by choosing complex numbers for A and B and requiring that A is the complex conjugate of B. Enforce this requirement by substituting $A = a + ib$ and $B = a - ib$ (where a and b are real numbers) into your solution and see how it simplifies.

Hint: Remember $\cos(-x) = \cos(x)$ and $\sin(-x) = -\sin(x)$.

(c) Describe how the decay rate and the oscillation frequency change as damping is increased.

To recap, a negative argument in the square-root $\sqrt{\gamma^2-\omega_0^2}$ gives rise to the oscillatory terms. If the square root is imaginary ($\gamma < \omega_0$), the exponentials are complex (sines and cosines multiplied by a decaying exponential) and we have oscillation. If the argument of this square root is positive ($\gamma > \omega_0$), then both of the fundamental solutions in Eq. (3.4) 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.

☞ All of the numbers in this manual are assumed to be in MKS units unless we tell you otherwise. Thus, time is in seconds, distance in meters, etc.

P3.2 (a) Use Mathematica to solve Eq. (3.3) with $y(0) = 1$ and $y'(0) = 0$. Pick a natural frequency ω_0 of 2π, so the period $T = 2\pi/\omega_0$ is 1, and use $\gamma = 10$. Plot the solution from $t = 0$ to $t = 2$ and note the overdamped behavior.

(b) Use the same parameters as in (a), except change the damping $\gamma = 0.025$. Have Mathematica plot the solution again from $t = 0$ to $t = 50$ to see what happens.

(c) Use the same parameters as in (a), except change γ so the system is critically damped. The plot will look just like overdamping with no hint that you are right on the border between damping and oscillation. To verify that you are right on the border decrease γ by 2% and make the plot again. Window the graph down enough that you can see that $y(t)$ goes a little bit negative, indicating that oscillation has begun. You will have to do some serious windowing to see this effect.

The Driven, Damped Oscillator

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

$$\frac{d^2}{dt^2}y(t) = -\omega_0^2 y(t) - 2\gamma\frac{d}{dt}y(t) + F_0\cos(\omega t) \tag{3.7}$$

Now we have two frequencies in play—the driving frequency ω and the damped-oscillator frequency ω_d given by Eq. (3.6). 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 (ω and ω_d), then the oscillations at ω_d damp out and the system transitions to a state of oscillation at the driving frequency ω.

P3.3 Use Mathematica solve Eq. (3.7) symbolically and plot it from $t = 0$ to $t = 300$ with $\omega_0 = 1$, $F_0 = 1$, $\omega = 1.1$, and $\gamma = 0.01$. Start from rest, with $y(0) = 0$ and $\dot{y}(0) = 0$. Note the initial beating between frequencies and verify graphically that the final oscillation frequency of $y(t)$ is ω.

Resonance

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* ω_r. If you drive the system at a frequency far from ω_r you only get small oscillations.

[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.

P3.4 Study resonance in the damped-driven oscillator by making plots of $y(t)$ vs. t with $y(0) = 0$ and $\dot{y}(0) = 0$. Use $F_0 = 1$, $\gamma = 0.05$, and $\omega_0 = 1$ and drive it with three frequencies: $\omega = 0.7$, $\omega = 1$, and $\omega = 1.3$. For each driving frequency, find the steady-state oscillation amplitude A (i.e. the amplitude of oscillation after all the beating has died out) and calculate ω_d. Write down ω_d and A in Table 3.1. Your amplitude numbers don't need to be really precise; we are just looking for general trends here.

ω	Amplitude
0.7	
1	
1.3	

$\omega_d =$

Table 3.1 Fill in your data from Mathematica

In this problem the steady-state oscillation amplitude gets bigger as the driving frequency gets in the neighborhood of ω_d. This may lead you to believe that the resonance frequency ω_r is the same as the damped frequency ω_d; but they aren't. When damping is small, ω_r and ω_d are close, but they are not the same. To calculate ω_r we need to do some more work.

Resonance Curves

We can study resonance in more detail by making a *resonance curve*. A resonance curve plots the steady state oscillation amplitude (after the beating has died away) vs. the driving frequency. The steady-state oscillation has the form

$$y(t) = A\cos(\omega t - \phi) \tag{3.8}$$

where A is the steady state amplitude, ω is the driving frequency, and ϕ is the phase difference between the driving force and the oscillator's response. Let's derive an expression for $A(\omega)$.

P3.5 Our algebra will be easier if we write Eq. (3.8) using complex notation like this:

$$y(t) = Ae^{i(\omega t - \phi)} \tag{3.9}$$

The real part of Eq. (3.9) is equivalent to Eq. (3.8) because of Euler's relation $e^{ix} = \cos x + i\sin x$. If we also write the driving force using this complex notation trick, we can find the steady state solution with just a little algebra.

(a) On paper, substitute the complex steady-state solution in Eq. (3.9) into the driven damped oscillator equation with a complex driving force:

$$\frac{d^2}{dt^2}y(t) = -\omega_0^2 y(t) - 2\gamma\frac{d}{dt}y(t) + F_0 e^{i\omega t} \tag{3.10}$$

Carry out the derivatives and cancel the common factor of $e^{i\omega t}$. Then manipulate the equation so that $e^{i\phi}$ appears in only one term on the right side. Replace the complex exponential with trig functions using $e^{i\phi} = \cos\phi + i\sin\phi$ to find the following equation:

$$-\omega^2 A + i2\gamma\omega A + \omega_0^2 A = F_0(\cos\phi + i\sin\phi)$$

Finally, separate this complex equation into two real equations: one for the real part and one for the imaginary part.

(b) Divide the two equations found in (a) to find the phase difference ϕ between the driving force and the oscillatory response:

$$\tan\phi = \frac{2\gamma\omega}{\omega_0^2 - \omega^2} \; .$$

Now square both equations found in (a), add them together, and solve for the the steady-state amplitude A to find

$$A(\omega) = \frac{F_0}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\gamma^2\omega^2}} \tag{3.11}$$

HINT: In solving for A it will be useful to recall that $\cos^2\phi + \sin^2\phi = 1$.

(c) The *resonance curve* $A(\omega)$ defined by Eq. (3.11) gives steady state amplitude of the oscillation as a function of the driving frequency. Make plots of Eq. (3.11) for the parameters in P3.4 and compare the plot with your data in Table 3.1. Also plot $\phi(\omega)$. $\phi(\omega)$ tells you the phase difference between the driving force and the oscillator response.

Now make plots for several values of γ and verify that a smaller damping coefficient γ leads to larger and sharper resonance. Show analytically that the peak of the resonance curve is not at the damped frequency ω_d, but occurs at

$$\omega_r = \sqrt{\omega_d^2 - \gamma^2} = \sqrt{\omega_0^2 - 2\gamma^2} \tag{3.12}$$

Learning Matlab: M-File Functions

P3.6 Read and work through *Introduction to Matlab*, Chapters 6-7. Type and execute all of the material in `typewriter font`. Then do the following exercises:

(a) Make a Matlab surface plot of the “mountain function" that appears on the cover of *Introduction to Matlab for Physics Students*:

$$f(x,y) = e^{-|x-\sin y|}\left(1+\frac{1}{5}\cos(x/2)\right)\left(1+\frac{4}{3+10y^2}\right) \; . \tag{3.13}$$

Make an inline function that evaluates $f(x,y)$ and plot it from -5 to 5 in x and from -6 to 6 in y and label the x and y axes. Make sure the labels correspond to the correct axes.

(b) Write an M-file function called InvSum.m that contains a loop to perform the sum

$$S(N) = \sum_{n=1}^{N} \frac{1}{n}$$

Your function should take `N` as an input and return the sum as its output. Write a separate script that contains a loop that loads a variable `S` with $S(N)$ from $N = 1$ to $N = 1,000$. Plot the `S` and note that the sum diverges as N gets bigger, but only weakly.

(c) Write another function called EulerSum.m that computes the quantity

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

Write a separate script that contains a loop 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 γ. To 15 digits, Euler's constant is

$$\gamma = 0.577215664901532$$

Add a second output to EulerSum.m that returns the error $|S_e(N) - \gamma|$. Make a 2-panel subplot that plots S_e in the upper pane and a `semilogy` plot of the error in the lower pane.

Lab 4

Phase Space and Fitting Data

One of the best ways of visualizing the solution of a second-order differential equation without actually solving it is to use *phase space*.[1] In classical mechanics you 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 x and v as the phase space variables. An example of a phase-space diagram is Fig. 4.1, which shows what phase-space looks like for the pendulum. As time progresses a particle follows these curves in (x, v) space, moving to the right above the x-axis and moving to the left below it. In this lab, we'll continue to explore the harmonic oscillator using flow plots in phase space.

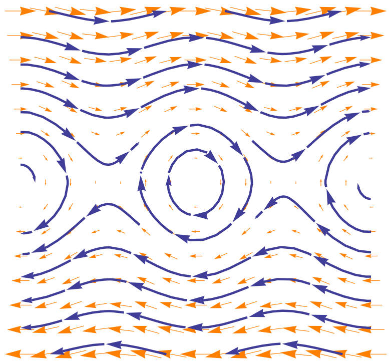


Figure 4.1 A phase space flow plot for a pendulum.

Flow Plots

Perhaps you learned in your differential equations course that a first order set of differential equations defines a flow in the space of dependent variables. For instance, here is the undamped harmonic oscillator:

$$\frac{dx}{dt} = v \quad ; \quad \frac{dv}{dt} = -\omega_0^2 x \quad . \tag{4.1}$$

If you think of dx/dt and dv/dt in Eq. (4.1) as flow velocities in the (x, v) phase space, the right-hand sides of these equations tell us what the flow lines look like in this space., just as $E_x(x, y)$ and $E_y(x, y)$ would tell us what electric field lines look like in a two-dimensional electric field problem. When we solve the differential equations to obtain $x(t)$ and $v(t)$, and let time t run, we find trajectories in the (x, v) plane parameterized by time, and these trajectories are the flow lines. But we can have a feel for what these trajectories are going to look like without solving the differential equations by drawing 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. The x and v components of this arrow are simply dx/dt and dv/dt, which in this case are $dx/dt = v$ and $dv/dt = -\omega_0^2 x$, simple functions of x and v.

Learning Mathematica: Flow Plots

P4.1 Read the section titled "Phase Space" in the Mathematica tutorial *Differential Equations with Mathematica* (available on the Physics 330 course web page).

[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.

Let's make some flow plots using Mathematica. They can also be made by Matlab using the `quiver` command, but in the interest of time we will only do it in Mathematica.

(a) Use Mathematica to make a flow plot of $x(t)$ and $v(t)$ for a simple harmonic oscillator with $\omega_0 = 2$. To make it look a little nicer, use the plot options

```
StreamPoints->Fine, VectorScale->Small,VectorStyle->Orange
```

Now plot three phase space trajectories using `ParametricPlot` with $x_0 = 1$ and $v_0 = 0$, $v_0 = 1.4$, and $v_0 = -1$ and run them from $t = 0$ to $t = 1$. Finally, 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.

(b) Now add linear damping to your model with damping coefficient $\gamma = 0.5$:

$$\frac{dx}{dt} = v \quad ; \quad \frac{dv}{dt} = -\omega_0^2 x - 2\gamma v \quad . \tag{4.2}$$

Make a flow plot for this model and overlay trajectories with the same initial conditions as (a), but run from $t = 0$ to $t = 20$. Explain how the arrows show what the solution will do.

Now change the damping coefficient to $\gamma = 4$ and explain how the flow plot describes the overdamped system.

(c) Repeat (b), but change your model to include air-damping with $\gamma = 0.5$ and $\bar{v} = 1$ in the following differential equation:

$$\frac{dx}{dt} = v \quad ; \quad \frac{dv}{dt} = -\omega_0^2 x - 2\gamma v|v|/\bar{v}\,. \tag{4.3}$$

Explain how this picture looks different from the ones in part (b), and why.

Learning Matlab: Linear Algebra and Curve Fitting

P4.2 Read and execute the examples in *Introduction to Matlab*, Chapter 8. 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 $m = 2.3$.

P4.3 Read and execute the examples in *Introduction to Matlab*, Chapter 9. Then complete the following exercises.

(a) Use `polyfit` to find a fourth-order polynomial approximation to the function $f(x) = e^{-x}J_1(x)$ on the interval $x \in [0,3]$. (This function and a mediocre third-order fit are shown in Fig. 4.2). Plot both $f(x)$ and the polynomial fit together on the same graph using different colors. Then plot the polynomial as a dashed line instead of a solid one. Finally, change the polynomial order to 6 and see if the fit improves.

(b) Make some fake data using the following code

```
x=-10:0.1:10;
y=cos(pi*(x)/10)+0.1*rand(1,length(x));
```

Save it to a text file called data.fil in a two-column format. Then find the best fit of this data to the following functional form

$$y(x) = a_1 \text{sech}(a_2 x) + a_3$$

You may use the code in *Introduction to Matlab* as a reference, but type all your code from scratch so you learn how each part of the code works.

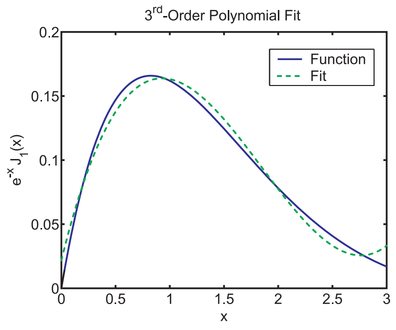


Figure 4.2 This third-order fit is not very good; you will do better.

P4.4 Read and execute the examples in *Introduction to Matlab*, Chapter 10. 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 π.

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

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

Lab 5

A Bouncing Ball

To this point we've solved differential equations numerically by using the black box of Mathematica's `NDSolve` command. In this lab we begin to peak inside the box and learn some crude techniques for numerically solving equations. We'll refine these techniques in the next lab.

Our first task is to learn how to work with numerical grids and take derivatives numerically. Then we'll model a simple dynamical system with these tools.

Learning Matlab: Interpolation and Calculus

P5.1 Read and work through *Introduction to Matlab*, Chapter 11. Type and execute all of the material in `typewriter 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 quadratic interpolation to plot a curve on the fine grid that passes through `y(1)`, `y(2)`, and `y(3)`. Overlay four curves: the coarse plotted as stars, the fine and the two fitted curves as lines. Use these curves to explain the benefits and hazards of using linear and quadratic interpolation and extrapolation.

P5.2 Read and work through *Introduction to Matlab*, Chapter 12. Type and execute all of the material in `typewriter font`. Then complete the following exercises.

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

$$\int_0^2 x^2 e^{-x} \cos x dx \quad . \tag{5.1}$$

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 `quadl` command using an inline function and passing it into `quadl`.

(b) Consider the simple function $f(x) = e^x$. Evaluate $f'(x)$ at $x = 1$ using both the forward and centered difference approximation to the first derivative. Write a loop that decreases h from 1 to 10^{-20} by dividing successively by 2 and calculate the error of the two derivative formulas (i.e., `abs(fp/exp(1)-1)` where `fp` is the numerical derivative) at each h. Make a `loglog` plot of the 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 5.2(c) 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. In Mathematica you can choose to work with as many digits as you want, but in Matlab you only have 15, 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.

Solving Differential Equations Numerically

Consider the motion of a projectile near the surface of the earth with no air resistance. The differential equations that describes the projectile are

$$\begin{aligned} \frac{dx}{dt} &= v_x & \frac{dy}{dt} &= v_y \\ \frac{dv_x}{dt} &= 0 & \frac{dv_y}{dt} &= -g \end{aligned} \tag{5.2}$$

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 in a numerical solution for a system like this is to think of time as being a discrete grid rather than a continuous quantity. It is easiest to have an evenly spaced time grid $[t_0, t_1, t_2, ...]$ with $t_0 = 0$, $t_1 = \tau$, $t_2 = 2\tau$, etc. We label the variables on this time grid using the notation $x_0 \equiv x(0)$, $x_1 \equiv x(\tau)$, $x_2 \equiv x(2\tau)$, etc. With this notation, we can write the equations in (5.2) using the (inaccurate) forward difference approximation of the derivative that you learned about in the reading:

$$\frac{x_{n+1}-x_n}{\tau} = v_{x,n} \qquad \frac{y_{n+1}-y_n}{\tau} = v_{y,n}$$
$$\frac{v_{x,n+1}-v_{x,n}}{\tau} = 0 \qquad \frac{v_{y,n+1}-v_{y,n}}{\tau} = -g \tag{5.3}$$

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 τ small enough, this approach can work OK.

By solving the equations in (5.3) we can obtain a simple algorithm for stepping our solution forward in time:

$$x_{n+1} = x_n + v_{x,n}\tau \qquad y_{n+1} = y_n + v_{y,n}\tau$$
$$v_{x,n+1} = v_{x,n} \qquad v_{y,n+1} = v_{y,n} - g\tau \tag{5.4}$$

This method of approximating solutions is called Euler's method. In general, it's not very good, especially over long times (many time steps). However, it works fine for our simple test system, and it provides a foundation for other better methods.

P5.3 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. Have your while loop exit when $x > 10$. Just store the new values over the top of the old ones in the loop, like this

```
x=x+vx*tau;
```

HINT: When you code the y equations, the order of assignment matter. Think about which y equation should come first.

(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, use the ideas from linear interpolation to determine a time step τ_1 such that an Euler step will take the ball to $y = 0$. Then take an Euler step with τ_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 τ 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.

Finally, make your model look more realistic by adding some loss to the bounce process, like this

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

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

Lab 6

The Pendulum

The equation of motion of a simple pendulum is

$$\ddot{\theta} = -\omega_0^2 \sin\theta \,, \tag{6.1}$$

where θ is the angle (in radians) between the pendulum and the vertical direction and ω_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. (6.1) 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 large amplitude behavior of the pendulum, which can be quite different from the simple harmonic oscillator.

P6.1 Prove that Eq (6.1) 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.

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

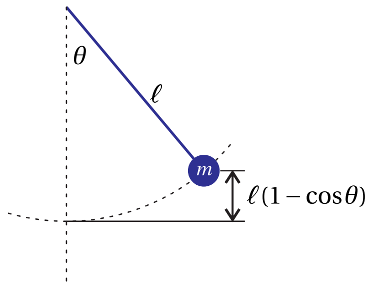


Figure 6.1 A simple pendulum comprised of a massless stick of length ℓ with a mass m at the end.

We can find the large-amplitude oscillation frequency of the pendulum by using an energy method.[1] The kinetic energy of the pendulum is $I\dot{\theta}/2$ and the potential energy is $mg\ell(1-\cos\theta)$ (see Fig. 6.1). The total energy of a pendulum can be found when the pendulum is at the maximum displacement, which we will denote by θ_0. At this point, the center of mass is at a height of $\ell(1-\cos\theta_0)$ above the equilibrium position, 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

$$\frac{1}{2}I\dot{\theta}^2 + mg\ell(1-\cos\theta) = mg\ell(1-\cos\theta_0) \tag{6.2}$$

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.

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

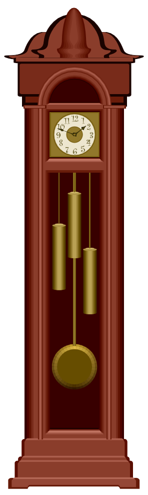


Figure 6.2 The frequency of a pendulum depends on the amplitude of oscillation. The variation of frequency with amplitude is smallest for low-amplitude oscillations, so its easier to get good accuracy with long pendulum and small angle oscillations as in a grandfather clock.

P6.2 Using paper and pencil, show that Eq. (6.2) can be written as

$$\omega_0 dt = \frac{d\theta}{\sqrt{2\cos\theta - 2\cos\theta_0}} \tag{6.3}$$

where you have separated variables.

To find the period of oscillation, we integrate both sides of Eq. (6.3) 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

$$\omega_0 \int_0^{T/4} dt = \frac{1}{\sqrt{2}} \int_0^{\theta_0} \frac{d\theta}{\sqrt{\cos\theta - \cos\theta_0}} \tag{6.4}$$

The time integral on the left is simple, but the θ 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. (6.4) as

$$T = \frac{4}{\omega_0} \int_0^{\pi/2} \frac{d\phi}{\sqrt{1 - \sin^2(\theta_0/2)\sin^2\phi}} \tag{6.5}$$

The ϕ integral in Eq. (6.5) is not any easier than the θ integral in Eq. (6.4), but it has come up in enough problems that it has been given a name: the complete elliptic integral of the first kind, called $K(m)$:

$$K(m) \equiv \int_0^{\pi/2} \frac{d\phi}{\sqrt{1 - m\sin^2\phi}} \tag{6.6}$$

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 period T of the pendulum as

$$T = \frac{4}{\omega_0} K\left(\sin^2(\theta_0/2)\right) \tag{6.7}$$

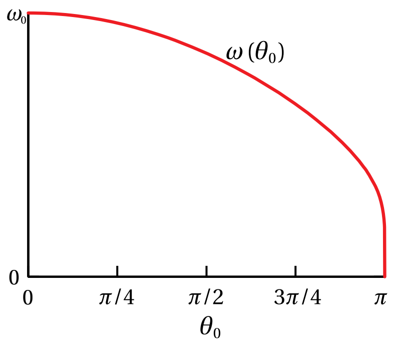


Figure 6.3 Oscillation frequency as a function of the maximum amplitude θ_0.

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 θ_0.

$$\omega(\theta_0) = \frac{\pi\omega_0}{2K\left(\sin^2(\theta_0/2)\right)} \tag{6.8}$$

Note that the natural oscillation frequency $\omega(\theta_0)$ of the pendulum depends on amplitude θ_0. This gives the pendulum some interesting characteristics.

P6.3 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 learn how to solve the pendulum equation numerically using Matlab.

Learning Matlab: Ordinary Differential Equations

P6.4 Read and work through *Introduction to Matlab*, Chapter 13. Type and execute all of the material in `typewriter font`. Then work the following problems.

(a) Set up Eq. (6.1) as a first order set of coupled equations and solve the system using Euler's method. Use $\omega_0 = 1$ and initial conditions $\theta(0) = \pi/2$ and angular velocity $\omega(0) = 0$, where $\omega(t) = \dot{\theta}(t)$. The example in *Introduction to Matlab* will be a big help. Plot your solution for 30 periods of oscillation and overlay a plot of a cosine function of matching amplitude and with a frequency $\omega(\theta_0)$ from Eq. (6.8). Evaluate how your accuracy changes as you vary τ.

(b) Modify your code from part (a) to use second-order Runge-Kutta. Plot your solution for 30 periods of oscillation and overlay a plot of a cosine function of matching amplitude and with a frequency $\omega(\theta_0)$ from Eq. (6.8). Evaluate how your accuracy changes as you vary τ, and compare with Euler's method.

(c) Now 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 $\omega(0) = 0$. Run your script and plot the solution for the following values of θ_0: 0.1, 0.5, 1.0, $\pi/2$, 0.9π, and 0.98π. For each value overlay a plot of a cosine function of matching amplitude and with a frequency $\omega(\theta_0)$ from Eq. (6.8). Verify that Eq. (6.8) gives the correct frequency, and that for large amplitudes the pendulum motion is not sinusoidal.

Important: As you worked through this material in *Introduction to Matlab*, you learned 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 `rhs6_4.m`, and change the call to `ode45` to correspond: `ode45(@rhs6_4,...)`. This will make it possible for you to come back later and see how you did each of the problems.

P6.5 Now let's drive the pendulum with an external torque, like this

$$\ddot{\theta} + \omega_0^2 \sin\theta = \alpha \sin\omega_\tau t\ . \tag{6.9}$$

Drive the pendulum at resonance for small amplitudes, with $\omega_0 = 1$, $\omega_\tau = 1$, and $\alpha = 0.1$. Since it is driven, you can just start it at rest: $\theta(0) = 0$ and $\dot{\theta}(0) = 0$. 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.

P6.6 Add some linear damping ($-\gamma\dot{\theta}$, $\gamma = 0.1$), to the right-hand side of Eq. (6.9). Use the same conditions as in (d) and watch how the motion changes.

Explain the damped behavior and explore how it depends on α. Also vary the driving frequency ω in the range $0.90\omega_0 \rightarrow 1.05\omega_0$ and explain why $\omega = \omega_0$ doesn't give the largest amplitude.

Lab 7

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

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt \tag{7.1}$$

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* $P(\omega)$

$$P(\omega) = |g(\omega)|^2 \tag{7.2}$$

$P(\omega)$ 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.

Learning Matlab: FFTs and Fourier Transforms

P7.1 Read and work through *Introduction to Matlab*, Chapter 14. Type and execute all of the material in `typewriter font`.

P7.2 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 = wavread('beethoven.wav');
```

Then construct the corresponding `t` time series by noting that the recording was sampled at 11025 points/second. Plot the signal versus time and plot its power spectrum versus ν (not ω) over the range 0-1000 Hz with both a linear scale and with `semilogy`. (You should get in the habit of looking at 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

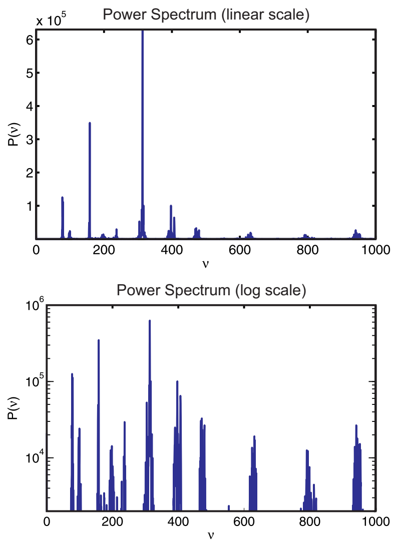


Figure 7.1 The power spectrum of the first four notes of Beethoven's 5th symphony.

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

by the violins/violas (400 Hz), cellos (200 Hz), and basses (100 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, 800 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. 7.2).

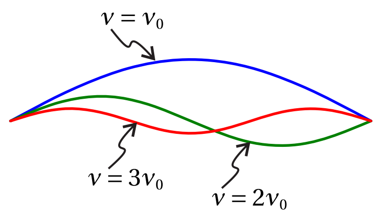


Figure 7.2 A string's fundamental mode of vibration has nodes at the ends and an antinode in the middle. However, the string can also vibrate in harmonic modes with nodes between the ends. When a musician drags a bow across a string, she excites mostly the fundamental, but the harmonics are also present. The frequencies of these modes are: ν_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 ω_0, but which only lasts for a finite time Δt. For example, consider the Gaussian function

$$f(t) = \cos(\omega_0 t) e^{-(t-t_0)^2/W^2} \tag{7.3}$$

which has a "bump" centered at t_0 with a width controlled by W. Because the signal oscillates at ω_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.

P7.3 Write a Matlab script to build $f(t)$ from Eq. (7.3), with t_0 chosen so that the bump is in the center of your time window. Plot $f(t)$ and its power spectrum for $\omega_0 = 200$ s^{-1} and $W = 10$, 1, and 0.1.

Choose appropriate values for your number of points N and your time step τ so that

(i) `fft` will run fast

(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 (Δt) and FWHM of the frequency peak ($\Delta\omega_{\text{plot}}$). Write these measurements in Table 7.1 for each value of W. Then deduce a rough linear "uncertainty" relation between the width of the time signal and the width of the frequency peak from this data.[3]

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

Table 7.1 Enter your data here

[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.

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

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 $\omega = 6000\ \mathrm{s}^{-1}$ to produce a spectrum with, say, a 1% width requires $\Delta\omega = (0.01)(6000\ \mathrm{s}^{-1}) = 60\ \mathrm{s}^{-1}$. Then the uncertainty principle tells us that this note can be produced by only holding it for the relatively short time of

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

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

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

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 Δt, 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 Δt, 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.

Windowing

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

P7.4 Modify Listing 14.1 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);
```

[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.

Plot the power spectrum versus ω 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. (7.3) via $\alpha \propto 1/W$—i.e. a bigger α 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.

Review of the Damped Harmonic Oscillator

In Lab 3 we studied the damped harmonic oscillator described by

$$\frac{d^2}{dt^2}y(t) = -\omega_0^2 y(t) - 2\gamma \frac{d}{dt} y(t)$$

and analytically found that the system oscillated at the frequency

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

P7.5 Use Matlab to numerically solve the damped harmonic oscillator for $\omega_0 = 1$ and $\gamma = 0.3$. Then verify that it oscillates at ω_d and not ω_0 by plotting the power spectrum. Use $y(0) = 10{,}000$ and $\dot{y}(0) = 0$ and run your model long enough to get good resolution.

HINT: Remember that the `fft` command requires evenly spaced data. For this problem it will work fine not to have a power of two for N.

Wave Propagation With Fourier Transforms

An interesting physics problem that can be addressed using Fourier transforms is pulse propagation through a dispersive medium. If you add a bunch of sinusoids of the form $g(\omega)e^{i\omega t}$ with appropriately chosen amplitudes $g(\omega)$, you can get them to interfere and produce any wave form with any temporal shape you like. If you know the temporal shape of the pulse at a given spatial point, say $x = 0$, you can get the amplitudes $g(\omega)$ for your coefficients by taking a Fourier transform:

$$g(\omega, x=0) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t, x=0)e^{i\omega t}dt \tag{7.4}$$

If you want to calculate the temporal form of the pulse at points besides $x = 0$, you have to add up traveling waves of the form $g(\omega)e^{i(kx-\omega t)}$. The different frequency components travel at their individual phase velocities, given by $v_p = \omega/k$. If there is no dispersion, k and ω are related by$k = \omega/c$ (where c is a constant with units of speed), so the phase velocity is the same for all frequency components: $v_p = c$. However, in many situations we have a different *dispersion relation* $k(\omega)$, so different frequency components move at a different speeds. As the frequency components shift phase relative to one another, the shape of the pulse evolves.

If the medium responds linearly to the waves, Fourier analysis provides an easy way to add up all of these frequency components with different phase velocities. If we freeze time, the phase change for each frequency component due to moving to a different point in space is given by kx. In complex notation, this means that the spectrum at a point x is related to the spectrum at $x = 0$ through

$$g(\omega, x) = g(\omega, x=0)e^{ikx} \tag{7.5}$$

If we take an inverse Fourier transform of this spectrum

$$f(t, x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\omega, x)e^{-i\omega t}d\omega = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\omega, x=0)e^{i(kx-\omega t)}d\omega \tag{7.6}$$

we can find the form of the pulse at an arbitrary x. Note that the last expression explicitly has the sum of traveling waves that we talked about conceptually.

P7.6 Let's use this technique to model the propagation of a water wave. Since Eqs. (7.4) and (7.6) are written as standard Fourier integrals, use the Matlab functions `ft.m` and `ift.m` as discussed in *Introduction to Matlab* rather than the regular `fft`. Use a time array that is 200 s wide with $N = 2^{16} = 65536$, like this

```
N = 2^16;
tmax = 200;
tau = tmax/(N-1);
t=0:tau:(N-1)*tau;
dw = 2*pi/tmax;
w = -(N/2)*dw:dw:dw*(N/2-1);
```

Notice that we've chosen a symmetric ω array because we will be using `ft.m` and `ift.m`.

Make your initial water pulse using $f(t,x=0) = e^{-(t-20)^2/0.5^2}$. This is a bump of water, sort of like the wave that a speedboat pushes away from it as it drives. (A boat makes two waves that propagate away from each other to make the wake—we'll just look at one of the waves.) Plot $f(t,x=0)$ to see its shape, and then, find the spectrum $g(\omega,x=0)$ of this pulse using `ft.m`.

Now find the spectrum of the pulse at $x=30$ by multiplying $g(\omega,x=0)$ by $e^{ik(\omega)x}$. The dispersion relation for water waves is

$$\omega^2 = gk\tanh(kd) \tag{7.7}$$

where g is the acceleration of gravity and d is the depth of the water. Since we need $k(\omega)$, Eq. (7.7) needs to be solved numerically. You know how to solve this equation using `fzero`, but since our ω array has 65,536 elements it would take a while to just do each element directly. In the interest of time, we've given you the matlab function `waterk.m` below that calculates $k(\omega)$ for you. Look it over briefly to understand how it works, then just use it.

Finally, find $f(t,x=30)$ by taking the inverse Fourier transform of $g(\omega,x=30)$ (using `ift.m`). Plot $f(t,x=0)$ and $f(t,x=30)$ on the same axis and explain what you see. Decide whether high or low frequency water waves travel faster in this model.

You have probably seen the dispersion of water waves behavior before. When the wave that defines the edge of a wake leaves the boat, it is mostly just a single bump of water. But it takes a lot of frequencies to make that bump. As the wave propagates, each frequency component travels at its own phase velocity and the bump spreads out and develops ripples. At the shore you get a long train of ripples rather than a single bump.

Listing 7.1 (waterk.m)

```
% function to calculate the water dispersion relation
function k = waterk(w,g,d)

% Make an approximate k. Error is less that 1e-15 for |kd| > 20
k = w.*abs(w) / g;

% Fix the errors for |kd| < 20 using dsolve
disp('Refining the dispersion relation.  Please be patient.');
disp('If your w array has a lot of small values, this can take a while.')

eq = inline('w^2 - g*k*tanh(k*d)','k','w','g','d');
for istep = 1:length(k)
    if (abs(k(istep)*d) < 20)
        k(istep) = fzero(@(k) eq(k,w(istep),g,d),k(istep));
    end
end
```

Lab 8

Pumping a Swing

There are two ways to pump a playground 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 ω_0 of the swing, you will be resonantly driving the pendulum, as we discussed in lab 6. This seems to be something that kids on a playground just do without knowing any physics at all.

To see how this works analytically, consider a the model of a swinger shown in Fig. 8.1.[1] The overall position of the swing is described by θ. 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 ϕ. If we make the assumption $m_2\ell_2 = m_3\ell_3$, the Lagrangian for this system simplifies to

$$L = \frac{1}{2}I_1\dot{\theta}^2 + \frac{1}{2}I_2(\dot{\theta}+\dot{\phi})^2 + Mg\ell_1\cos\theta \tag{8.1}$$

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

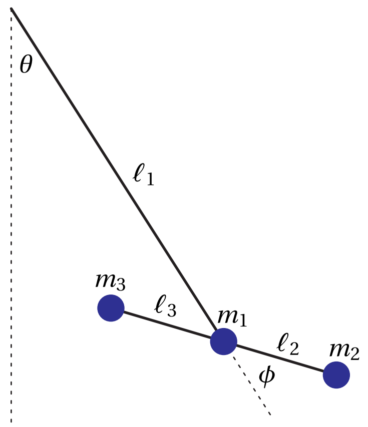


Figure 8.1 A simple model of a swing being pumped from the seated position.

[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).

P8.1 On paper, use the Lagrangian equation of motion for θ, 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 θ. Then assume that the swinger twists harmonically, with

$$\phi(t) = A + A\cos(\omega_\phi t) \tag{8.2}$$

and show that the equation of motion becomes

$$\ddot{\theta} + \omega_0^2 \sin\theta = \alpha\cos(\omega_\phi t) \tag{8.3}$$

where

$$\omega_0^2 = \frac{Mg\ell_1}{I_1 + I_2} \qquad \alpha = \frac{I_2 A\omega_\phi^2}{I_1 + I_2} \tag{8.4}$$

Compare with Eq. (6.9) and confirm that this is the equation for a driven pendulum.

P8.2 Add some friction to Eq. (8.3) 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 $\gamma = 0.1$ 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 8.1 (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
    xswing=l1*sin(xe(istep));
    yswing=-l1*cos(xe(istep));

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

    % 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]

$$\ddot{x} + \gamma \dot{x} + \omega_0^2 (1 + \epsilon \cos(\omega_p t)) x = 0 \,. \tag{8.5}$$

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://www.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 ℓ 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 (8.5) 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. (8.5) gives a reasonable approximation to the the motion of a swing.

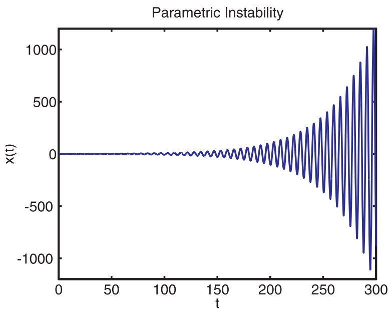


Figure 8.2 Pumped swing instability.

P8.3 (a) Use Matlab to solve Eq. (8.5) 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 ω_0 and the parametric drive at ω_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. 8.2. 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 ϵ, but that the instability growth rate is small for small ϵ.

(d) Show that when ω_p is not quite $2\omega_0$ the oscillator is stable for small ϵ, but that when ϵ 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 ϵ 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 ω_p and ω_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

$$x^3 = 1 + 0.1e^x \tag{8.6}$$

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:

$$x^3 = 1 \quad \Rightarrow \quad x_0 = 1 \tag{8.7}$$

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}:

$$x^3 = 1 + 0.1e^{x_0} \quad \Rightarrow \quad x^3 = 1 + 0.1e^1 \quad \Rightarrow \quad x_1 = (1 + 0.1e^1)^{1/3} \tag{8.8}$$

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

$$x^3 = 1 + 0.1e^{x_1} \quad \Rightarrow \quad x^3 = 1 + 0.1\exp\left[(1 + 0.1e^1)^{1/3}\right] \tag{8.9}$$

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

P8.4 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. 8.6 by continuing this step by step approach all the way to 15 significant figures in the Matlab command window by typing

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

and then using the ↑ key to repeat the second 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 $x = 1.090733542308225$; 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 $\omega_p = 2\omega_0$.

P8.5 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 ϵ 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 ϵ 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. (8.5) is solved by

$$x_0(t) = A\cos(\omega_0 t) \tag{8.10}$$

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. (8.5) down again, but with x_0 in place of x in the small term $\omega_0^2\epsilon\cos(\omega_p t)x$:

$$\ddot{x} + \gamma\dot{x} + \omega_0^2(1 + \epsilon\cos(\omega_p t))x_0 = 0 \quad \Rightarrow \tag{8.11}$$

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

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

P8.6 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

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

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 P8.5 and in your Mathematica solution?

In this first-order perturbation theory we see that in addition to the peak at ω_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

$$-\epsilon\omega_0\cos(\omega_p t))x_1 \ . \qquad (8.12)$$

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 ϵ. 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 ω_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 ω_0 (note that $-\omega_0$ is just as resonant as ω_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 $-\omega_0$, ω_0, ω_0, and $3\omega_0$. Three out of four come back to ω_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 ϵ, this choice for ω_p is less unstable.

P8.7 Explain all of the frequency peaks in your spectrum from P8.5 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

$$\ddot{\theta}+2\frac{\dot{\ell}}{\ell}\dot{\theta}+\gamma\dot{\theta}+\frac{g}{\ell}\sin\theta = 0 \ . \qquad (8.13)$$

If we let the length change sinusoidally at frequency ω_p by only the small amount ΔL about the constant length L_0, then

$$\ell(t) = L_0 + \Delta L \cos\omega_p t \,. \tag{8.14}$$

P8.8 Use Matlab to solve Eqs. (8.13) with (8.14) 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 ΔL 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 ΔL from zero up to around 0.3 m with $\omega_p = 2\omega_0$ and find the threshhold value of ΔL at which the swing becomes unstable. (Explain to your TA why the pendulum amplitude doesn't just keep getting bigger forever.)

Lab 9

Chaos

The van der Pol Oscillator

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

$$\ddot{x} - \epsilon(\ell^2 - x^2)\dot{x} + \omega_0^2 x = 0 \,. \tag{9.1}$$

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. (9.1) 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. 9.1 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.

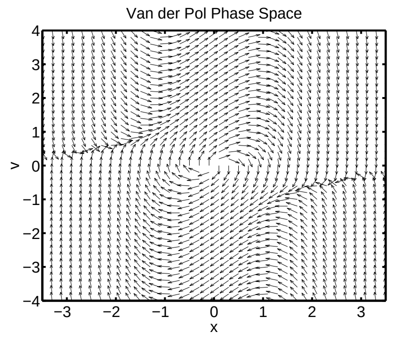


Figure 9.1 Flow in phase space for the Van der Pol oscillator with $\omega_0 = 1$, $\ell = 1$, and $\epsilon = 1$.

P9.1 (a) Use Matlab's `ode45` to solve Eq. (9.1) 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 $\omega_0 = 0$ to $\omega_0 = 20$ and note where the major peaks are.

[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).

P9.2 Now let's add a driving force to the van der Pol oscillator, like this:

$$\ddot{x} - \epsilon(\ell^2 - x^2)\dot{x} + \omega_0^2 x = A\cos\omega t\,. \tag{9.2}$$

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 P9.2 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:

$$\frac{d^2\theta}{dt^2} + \gamma\dot{\theta} + \omega_0^2 \sin\theta = A\cos\omega t\,. \tag{9.3}$$

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, ω, ω_0, and γ 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.

P9.3 A simple system in which chaos can be observed is a particle moving in a potential well with two low spots:

$$U(x) = -\frac{x^2}{2} + \frac{x^4}{4}\,. \tag{9.4}$$

(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

$$F_x = -\frac{\partial U}{\partial x} \tag{9.5}$$

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. 9.2. 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).

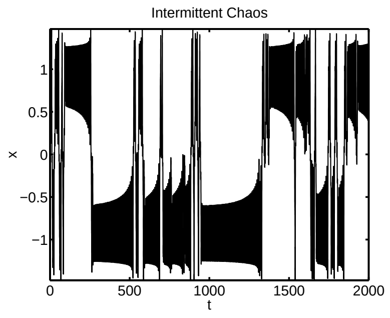


Figure 9.2 Intermittent random bouncing between the two wells.

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

The random switching back and forth between equilibrium positions observed in P9.3(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 P9.3(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?".

P9.4 (a) Make a phase space plot for the system in P9.3(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. 9.3 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 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. 9.3. 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 π. 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.

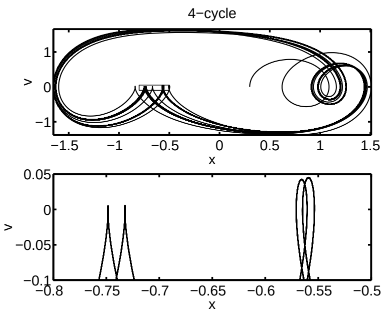


Figure 9.3 Full time history, then final 4-cycle state from the small window. There are fewer loops in the lower trace because it is the final state; the extra loops in the box in the upper trace are from early times.

Fractals

P9.5 (a) The Fibonacci sequence F_n is defined by

$$F_1 = 1 \quad ; \quad F_2 = 1 \quad ; \quad F_n = F_{n-1} + F_{n-2} \;\; \text{for } n \geq 3 \tag{9.6}$$

(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.

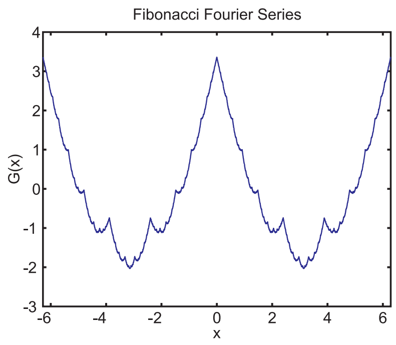


Figure 9.4 The function plotted in P9.5

(b) Now define an array `x` that goes from -2π to 2π 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

$$G(x) = \sum_{n=1}^{100} \frac{\cos(F_n x)}{F_n} \,. \tag{9.7}$$

Plot this function vs. x and carefully observe its shape (this function is shown in Fig. 9.4). 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 10

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 κ_1 and κ_2 for pendulum 1 and pendulum 2, respectively. This leads to a potential energy

$$U = \frac{1}{2}\kappa_1\theta_1^2 + \frac{1}{2}\kappa_2\theta_2^2 \, . \tag{10.1}$$

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

$$U_c = \frac{1}{2}\kappa_c(\theta_1 - \theta_2)^2 \tag{10.2}$$

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

$$U = \frac{1}{2}\kappa_1\theta_1^2 + \frac{1}{2}\kappa_2\theta_2^2 + \frac{1}{2}\kappa_c(\theta_1 - \theta_2)^2 \, . \tag{10.3}$$

P10.1 Combine this potential energy function with the kinetic energy

$$T = \frac{1}{2}I_1\dot{\theta}_1^{\,2} + \frac{1}{2}I_2\dot{\theta}_2^{\,2} \tag{10.4}$$

[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

$$\frac{\partial L}{\partial q_i} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) = 0 \tag{10.5}$$

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

$$\omega_0^2 = \frac{\kappa}{I} \quad ; \quad \omega_c^2 = \frac{\kappa_c}{I} \quad . \tag{10.6}$$

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

If you did P10.1 correctly, you should have arrived at the following four first-order differential equations for describing the motion of the coupled-pendulum system:

$$\dot{\theta}_1 = \omega_1 \tag{10.7}$$
$$\dot{\theta}_2 = \omega_2 \tag{10.8}$$
$$\dot{\omega}_1 = -\omega_0^2\theta_1 - \omega_c^2(\theta_1 - \theta_2) \tag{10.9}$$
$$\dot{\omega}_2 = -\omega_0^2\theta_2 - \omega_c^2(\theta_2 - \theta_1). \tag{10.10}$$

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 ω_0 is associated with the natural frequency of a single pendulum without any coupling, and the parameter ω_c is associated with the natural frequency of the middle section of tubing when attached to the two pendula.

P10.2 (a) Use Mathematica to solve Eqs. (10.7)-(10.10) 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 $\omega = 1.3$ 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 be looking at the beat plot in Fig. 10.1. 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.

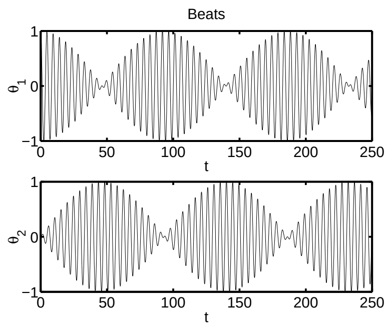


Figure 10.1 Energy passes back and forth between θ_1 and θ_2 due to beating.

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

(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 ω_+ and ω_- by

$$\omega_b = \omega_+ - \omega_- \,. \tag{10.11}$$

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

$$\ddot{\theta}_2 = -\gamma\dot{\theta}_2 + \cdots \tag{10.12}$$

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 θ_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.

P10.3 (a) Begin by making *both* pendulums be damped and driven as described in P10.2(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 θ_1 and θ_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 ω^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 ω_c? Comment on what your answer to this last question has to do with real clocks on a wall.

Lab 11

The Pendulum with a High Frequency Driving Force

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

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

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 ω. 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 ξ 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 ω large). All of the other high frequency terms are small compared to these two because ξ 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 ξ, $\cos\omega t$, 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_{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.

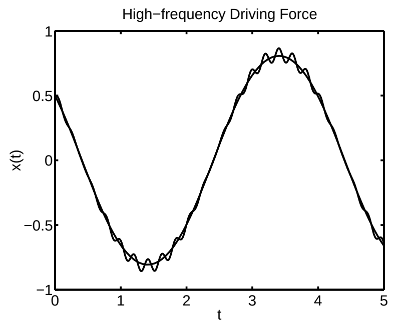


Figure 11.1

(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.

Learning Matlab: Publication Quality Plots

P11.2 (a) Read and work through *Introduction to Matlab*, Appendix A. Type and execute all of the material in `typewriter font`.

(b) Produce a publication quality EPS plot of some fake data and a fake model produced by

```
x=0:pi/200:2*pi;
f=sin(x).*exp(-x)+0.01*sin(15*x);
data  =  f  +  0.1*rand(1,length(x))-0.05;
```

Plot the data as red circles and the model as a blue line, and make your eps 16 cm wide and 8 cm tall. Make the range of the plot from $\theta = 0$ to $\theta = 6\pi$ and make sure the x-axis ticks are labeled correctly. This wide format is sometimes used when a figure need to span two columns in the printed article.

Import this EPS into a word processor and print it to verify that the sizing is correct. (Don't resize it after importing.)

NOTE: Many word processors don't render EPS files very well on screen, so the graph may look bad when viewed on-screen. However, when you print the document (or create a PDF of the document), the graph is rendered by the printer (or PDF viewer) and will look nice and sharp. In contrast, bitmap graphics look good on-screen but look bad when printed (unless they have very high resolution).

Lab 12

Two Gravitating Bodies

Consider two masses interacting through Newton's law of gravity: [1]

$$m_1\ddot{\mathbf{r}}_1 = -\frac{Gm_1m_2}{|\mathbf{r_1}-\mathbf{r_2}|^3}(\mathbf{r_1}-\mathbf{r_2}) \tag{12.1}$$

$$m_2\ddot{\mathbf{r}}_2 = -\frac{Gm_1m_2}{|\mathbf{r_1}-\mathbf{r_2}|^3}(\mathbf{r_2}-\mathbf{r_1}) \tag{12.2}$$

There are components of the motion described by these equations: $x_1(t)$, $y_1(t)$, $z_1(t)$, $\dot{x}_1(t)$, $\dot{y}_1(t)$, $\dot{z}_1(t)$, $x_2(t)$, $y_2(t)$, $z_2(t)$, $\dot{x}_2(t)$, $\dot{y}_2(t)$, $\dot{z}_2(t)$.

P12.1 (a) Use the equations of motion above, plus $\dot{x}_1 = v_{x1}$, etc., to obtain the 12 first order differential equations for this system. Write them down on paper.

(b) Have Matlab solve this system of equations using $G = 1$, $m_1 = 1$, $m_2 = 2$ and initial conditions

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

Run the solution from $t = 0$ to $t = 50$. After obtaining the solution arrays interpolate them onto new arrays equally spaced in time (`x1e, y1e, z1e, x2e, y2e,...` with `N=5*length(t)`).

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
```

[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.

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. 12.1 (after the appropriate rotation of your figure).

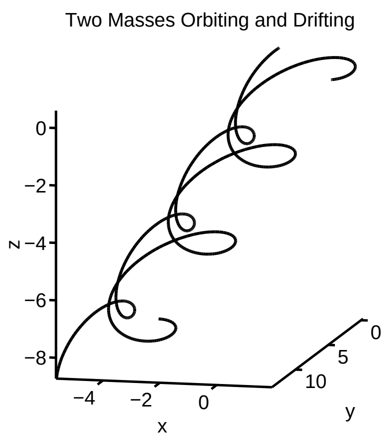


Figure 12.1 Two masses interacting via the inverse-square law.

We are about to ask you to do some calculations with 3-dimensional vectors and it will be easier if you turn your separate x, y, and z arrays into matrices instead. For instance, the $\mathbf{r}_1$ vector would be a matrix with 3 columns and as many rows as there were time steps in the output from `ode45`. The vectors for the 3-dimensional positions and velocities of both particles can be defined like this:

```
r1=[x1,y1,z1];
r2=[x2,y2,z2];
v1=[vx1,vy1,vz1];
v2=[vx2,vy2,vz2];
```

Let's also define versions of these vectors with the data equally spaced in time:

```
r1e=[x1e,y1e,z1e];
r2e=[x2e,y2e,z2e];
v1e=[vx1e,vy1e,vz1e];
v2e=[vx2e,vy2e,vz2e];
```

With them in this form the rest of this problem will be easy.

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:

$$\mathbf{R} = \frac{m_1\mathbf{r}_1 + m_2\mathbf{r}_2}{m_1 + m_2} \quad ; \quad \mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2 \tag{12.3}$$

P12.2 (a) Use Matlab to make 3d plots of $\mathbf{R}$ and $\mathbf{V} = \dot{\mathbf{R}}$ for the case you just ran and show that their motion is very simple. Using the matrices we had you define above, you can define the vectors $\mathbf{R}$, $\mathbf{r}$, $\mathbf{V}$, and $\mathbf{v}$ like this:

```
R=(m1*r1+m2*r2)/(m1+m2);
V=(m1*v1+m2*v2)/(m1+m2);
r=r1-r2;
v=v1-v2;
```

(equally spaced in time too:)

```
Re=(m1*r1e+m2*r2e)/(m1+m2);
Ve=(m1*v1e+m2*v2e)/(m1+m2);
re=r1e-r2e;
ve=v1e-v2e;
```

To make a 3-dimensional plot of these quantities use the command:

```
plot3(R(:,1),R(:,2),R(:,3))
```

Make sure you understand how the colon command works in this example.

(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.

(c) To see why the difference motion lies in a plane, compute the angular momentum in the center of mass frame

$$\mathbf{L} = m_1(\mathbf{r}_1 - \mathbf{R}) \times (\mathbf{v}_1 - \mathbf{V}) + m_2(\mathbf{r}_2 - \mathbf{R}) \times (\mathbf{v}_2 - \mathbf{V}) \tag{12.4}$$

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 you can plot each component of the angular momentum vs. time:

```
plot(t,L(:,1),t,L(:,2),t,L(:,3))
```

(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).

Kepler's Laws

We are now going to use Matlab to verify Kepler's laws. In Newton's more precise form, they are

Kepler's First Law The difference vector **r** sweeps out an ellipse with body number 2 at one focus. (Note that by definition **r** is the vector that points from m_2 to m_1, so $\mathbf{r} = 0$ is at the position of m_2).

Kepler's Second Law An imaginary line drawn from m_2 to m_1 (the $\mathbf{r}$ vector) sweeps out equal areas in equal times.

Kepler's Third Law The squares of the planetary periods are proportional to the cubes of the semi-major axes, or in more modern (and more precise) language, the period T of the orbit is given by

$$T = \frac{\pi}{\sqrt{2}} \frac{\mu^{1/2} G m_1 m_2}{(-E)^{3/2}} \tag{12.5}$$

where μ is the reduced mass

$$\mu = \frac{m_1 m_2}{m_1 + m_2} \tag{12.6}$$

and where E is the total energy in the center of mass frame:

$$E = \frac{\mu}{2} |\dot{\mathbf{r}}|^2 - \frac{G m_1 m_2}{|\mathbf{r}|} \tag{12.7}$$

(remember that $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$ and that $\dot{\mathbf{r}} = \mathbf{v} = \mathbf{v}_1 - \mathbf{v}_2$).

Before we begin verifying these laws, we need to simulate the solar system that Kepler studied. To do this, make m_1 very small compared to m_2, for example $m_2 = 1$ and $m_1 = 1e-8$.

Also, you can save yourself a lot of trouble by changing the initial conditions so that the motion lies only in the xy plane, and so that the ellipse is oriented squarely in this plane. To keep the motion in the xy plane set all z positions and velocities to zero. Then make the sun stationary by setting the other coordinates and velocities of m_2 to zero. Finally, give m_1 simple initial conditions that will make an elliptical orbit squarely oriented in the xy plane. (Go to the board and draw some pictures; with a little imagination you will be able to see what these initial conditions are like).

Note: You will need to be a little bit careful to make sure that the total energy is negative so that the orbits are confined. If your plots of the orbit look like parabolas headed to infinity and beyond, start with less initial velocity.

OK, now we are ready to verify Kepler's laws.

P12.3 (a) Verify Kepler's first law.

You will need to remember that if the semi-major axis of the ellipse is a and if the semi-minor axis of the ellipse if b, then the distance from the center of the ellipse to a focus is $f = \sqrt{a^2 - b^2}$.

To show that it is an ellipse first make sure that you are using initial conditions that make the orbit be squared up in the xy plane (i.e. have the axes of your ellipse lined up with the x and y axes). When you have an ellipse in the xy plane on your screen use `hold on` and the parametric form for an ellipse

$$x = a \cos s \quad y = b \sin s \quad s \in [0, 2\pi]$$

to lay an ellipse over your orbit to see if they match. You can find the values of a and b from your x and y data, then shift the ellipse by the focus distance to overlay the two ellipses.

Once you have a nice 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 the figure below.

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.

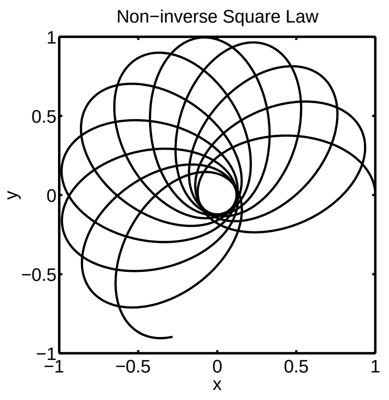


Figure 12.2 Precession of the orbit, non-inverse-square.

(b) Verify Kepler's Second law.

This law is hard to verify numerically if the vector sweeps out a large area per time interval. To make it easier make sure you have lots of equally spaced time points along your orbit from part (a). Use a cross product to calculate successive areas using a triangle approximation and see if they are close to equal. Be sure to use your equally-spaced **r** array `re` to do this calculation. As in part (a) it is good idea to look at this data by plotting the small areas vs. time to see how constant they are.

(c) Verify Kepler's Third law.

To find the period look at a plot of x vs. t, or some other quantity that varies periodically during an orbit. Zoom in on the graph to find the period of the orbit.

To find E recall that energy is conserved, so you only need to evaluate E at one point in time, which is easy to do by using the initial conditions.

You could find the period more precisely by using the methods discussed in *Introduction to Matlab*, Sec. 13.4. If you have time, you might want to read through this section to see how Matlab's event-finder works.

Lab 13

Hysteresis in Nonlinear Oscillators (two weeks)

The topic for this lab is *hysteresis* in nonlinear oscillators, an interesting and important effect. If you want to read more about this topic, see below. [1] The laboratory covers two lab periods and consists of lots of reading and things to do. You will need to use both the Mathematica and Matlab skills that you have learned over the semester in order to complete it.

Qualitative Analysis

Consider a damped-driven harmonic oscillator whose spring becomes weaker as the amplitude increases according to the equation of motion

$$\ddot{x} = -2\gamma\dot{x} + F(x) + A\cos\omega t\ , \tag{13.1}$$

where the restoring force $F(x)$ is given by

$$F(x) = -\frac{2}{3}\frac{\tanh(3x/2)}{\cosh^2(3x/2)}\ , \tag{13.2}$$

and where both linear damping ($-2\gamma\dot{x}$) and a sinusoidal driving force ($A\cos\omega t$) have been included. Note that the particle mass has been set to $m = 1$. So you don't panic at the ugliness of $F(x)$, let's plot it so you can see that it's nice and regular.

P13.1 Plot both the force $F(x)$ and the potential energy function $U(x)$ associated with this oscillator:

$$F(x) = -\partial U/\partial x \tag{13.3}$$

between $x = -3$ and $x = 3$.

Look at the potential energy plot now and notice that the bottom of the well looks pretty parabolic. This means that a low energy particle feels a force that is almost the same as the simple linear restoring force of the simple harmonic oscillator.

P13.2 To clearly understand this point, expand $F(x)$ in a Taylor series in small x up to 7th order and show that the natural frequency of oscillation for a particle with small displacement x is $\omega_0 = 1$.

HINT: Write down the equation of a simple harmonic oscillator and compare it to your expansion.

[1] R. Baierlein, *Newtonian Dynamics* (McGraw Hill, New York, 1983), p. 81-88, and G. Fowles and G. Cassiday, *Analytical Mechanics* (Saunders, Fort Worth, 1999), p. 108-110.

But if the particle were to gain more energy and move higher in the well, then it would be sampling the potential energy curve where it departs from a parabolic shape by opening out. This means that over part of the particle's orbit it feels less restoring force than the simple linear force $F = -x$ (see your plot of $F(x)$), and less restoring force means a lower oscillation frequency for the particle. This nonlinear effect, by which natural frequency becomes a function of oscillation amplitude (which we have seen before with the pendulum), has strange consequences for the resonance curve of this oscillator.

At the beginning of the course you made plots of the amplitude of a driven-damped harmonic oscillator and saw that when the damping was large there was only a small bump in the plot of the oscillation amplitude vs. driving frequency ω. But as the damping was decreased this bump became more and more peaked, until finally a sharp resonance curve was observed. In this lab we will study the same kind of resonance behavior for this nonlinear oscillator, but we will find that the resonance curves develop a very odd twist. To follow this story to its conclusion will require quite a bit of mathematics and computation, but before we tackle the technical details let's try to think about the problem qualitatively first.

To understand why nonlinearity makes so much difference in resonance, let's imagine two different scenarios, represented in Fig. 13.1

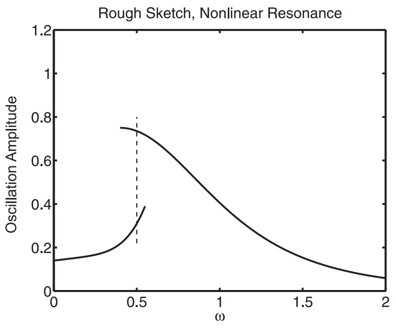


Figure 13.1 A rough sketch of a nonlinear resonance curve.

(i) Using a relatively small driving force and weak damping let's imagine starting at small driving frequency ω and then gradually increasing it toward resonance. At first the oscillator amplitude is low enough that simple harmonic oscillator behavior is observed, but as resonance is approached and the amplitude increases the resonant frequency shifts downward. Hence we might expect the resonance peak to occur at a lower value of ω than the value of $\omega_0 = 1$ that you calculated earlier in this section. This situation is indicated by the lower curve in Fig. 13.1.

(ii) Again using small driving force A and weak damping, let's imagine starting at high frequency and sweeping ω downward toward resonance. At first the amplitude is small and harmonic oscillator behavior is observed. But as we approach resonance and the amplitude increases the resonant frequency decreases, moving further away from us. We keep decreasing ω, the amplitude continues to grow because we are still approaching resonance, and the resonant frequency decreases still further. As this process continues we find ourselves at a frequency below $\omega_0 = 1$, and then even below the frequency where the early resonance occurred in scenario (i) (as shown by the upper curve in Fig. 13.1) but with the amplitude still high because we are continuing to chase the decreasing resonance frequency.

But this means that at some values of the driving frequency there are two possible oscillation amplitudes, as indicated by the dashed line in Figure 13.1. The curves in this figure look completely different from the ordinary resonance curves of the simple harmonic oscillator and the goal of this lab is to use computation to figure out how the upper and lower branches in Fig. 13.1 are related.

Mathematica Analysis

As a first attempt to understand the behavior of Fig. 13.1 it will be helpful to do some analysis with Mathematica. This analysis will only be approximate, but will help us see how the resonance curve might behave. The first approximation is to assume that x is small enough that it makes sense to only keep the first two non-zero terms in the Taylor expansion of $F(x)$ that you did in P13.2. Under this approximation, the force given in Eq. (13.2) is

$$F(x) \approx -x + 3x^3 \ .$$

The second approximation is to ignore the damping term in Eq. (13.1), so that our equation of motion becomes

$$\ddot{x} = -x + 3x^3 + A\cos\omega t \ . \tag{13.4}$$

Since we are interested in the driven response of this system, and since the driving force is proportional to $\cos\omega t$, we certainly expect the response $x(t)$ to also contain a term like $\cos\omega t$. But if $x(t) \propto \cos\omega t$, then because of the cubic term in the equation of motion its solution probably also involves a term like $\cos^3\omega t$. And since $\cos^3\omega t = \frac{3}{4}\cos\omega t + \frac{1}{4}\cos 3\omega t$, we then expect $x(t)$ to involve $\cos 3\omega t$ as well. Let's use these two suspected ingredients to approximately predict the amplitude of this steady oscillation. Assume that $x(t)$ is of the form

$$x(t) = c_1\cos\omega t + c_3\cos 3\omega t \ . \tag{13.5}$$

P13.3 (a) Use Mathematica to substitute this form for $x(t)$ into Eq. (13.4). Turn powers of cosine into terms containing $\cos(n\omega t)$, $n = 1, 3, 5, ...$, and then collect the terms that are proportional to $\cos\omega t$ and $\cos(3\omega t)$. We will ignore all of the higher order harmonics, hoping that they are small.

You will find that the collected form of the equation looks like this:

$$(\cdots)\cos(\omega t) + (\cdots)\cos(3\omega t) + \cdots = 0 \ .$$

Since $\cos(\omega t)$ and $\cos(3\omega t)$ are independent functions of time their coefficients $(\cdots)$. must separately be equal to zero, giving us two messy equations to determine the coefficients c_1 and c_3 in terms of A and ω.

(b) Assume that c_1 is small and that c_3 is even smaller to obtain from the $\cos 3\omega t$ component of the equation a simple approximate expression for c_3 in terms of c_1 and ω. In doing so make sure you only keep the terms that contain c_3 as a linear factor (drop any terms containing powers of c_3) and when you are considering what to do with two terms containing c_1^3 and $c_1^2 c_3$, keep c_1^3 and drop $c_1^2 c_3$ because we expect $c_3 \ll c_1$. You should find

$$c_3 = \frac{3c_1^3/4}{1-9\omega^2} \tag{13.6}$$

Under what conditions is c_3 *not* small as we assumed? In particular, what value of ω totally invalidates this assumption? This special value of ω will make an appearance later in this lab.

(c) Now use this approximation for c_3 in the $\cos\omega t$ equation to obtain an approximate cubic equation for c_1 involving ω and A by eliminating the higher order terms for c_1. Surprisingly, you will find that c_3 plays no role in the approximate equation for c_1 which is

$$c_1 - \omega^2 c_1 - \frac{9}{4}c_1^3 = A \tag{13.7}$$

(d) Have Mathematica solve for the three roots of this equation with $A = 0.136$ and put the formulas for the three values of c_1 in the variables `s1`, `s2`, and `s3`.[2]

(e) Now make two separate plots of `s1, s2, s3`:
(i) plot $|\text{Re}(c_1)|$ from $\omega = 0..2$ (all three roots on the same plot) and
(ii) plot $\text{Im}(c_1)$ from $\omega = 0..2$ (again, put all three roots on the same plot.
The second imaginary plot is important because if c_1 has an imaginary part, then this root has no physical significance in our problem, meaning that the corresponding parts of the curves in the real plot (i) should be ignored. (And the reason that we plot the absolute value $|\text{Re}(c_1)|$ is that c_1 is an amplitude, so we don't want to distinguish between positive and negative values).

(f) Now that you know which roots are physical, stare at the plot of the absolute value of $|c_1|$ and notice that for driving frequency ω below about 0.6 there are three real roots for c_1, but beyond 0.6 there is only one.

This change from one solution to three is the effect that makes this oscillator so interesting, and gives us a hint about how to complete Fig. 13.1. To see why, consider the following three scenarios for starting the oscillator at some driving frequency ω and then slowly changing it using the c_1 curves you have just made.

(i) Suppose that we start with a large driving frequency, then slowly decrease ω. Above 0.6 there is a single amplitude c_1, and as the frequency decreases

[2]Maple can solve a cubic like this, but the answers look horrible, so just assign each one to a separate variable like this: `sol:=solve(eq,c1): s1:=sol[1]: s2:=sol[2]: s3:=sol[3]:` (use colons at the end instead of semicolons so you don't have to look at pages and pages of blue output).

the curve predicts that c_1 should increase. When we reach 0.6 there are three possible solutions, but if we only make slow changes in the frequency, perhaps we will stay on the upper branch and c_1 will continue to increase.

(ii) Suppose that we start at low driving frequency and slowly increase ω toward 0.6. The amplitude c_1 will increase slowly along the lower branch, but soon the amplitude begins to increase rapidly because of the "early resonance" effect discussed in (i), and as we pass 0.6 the lower branch ceases to exist and the oscillator is forced to jump up to the upper branch, after which it decreases in amplitude as ω increases.

(iii) And what about the curve which is intermediate between high and low amplitude between $\omega = 0$ and $\omega = 0.6$? As you will see later, this branch is unstable and an oscillator that tries to live here will quickly either drop down to the lower branch or jump up to the upper one. And as you will also see later, if damping is added the two upper curves no longer both continue on to $\omega = 0$ as happens in your Mathematica curves. Instead they connect together, as you can see on the picture that graces the course web page (the red outlines are resonance curves for various values of the damping constant γ) or in Fig. 13.2 below. The surprising result is that the nonlinear resonance curves are like standard harmonic oscillator resonance curves that have been pushed over, like a flopped-over witches hat. During the rest of this lab you will be writing Matlab code to make such curves.

The important nonlinear effect in this behavior is that at frequencies below 0.6 you can't tell what the driven amplitude of the oscillator will be without knowing the details of how the driving frequency has been changing. This effect is called *hysteresis*, the Greek word for memory. This word is appropriate here because the state of the oscillator doesn't just depend on the current value of ω, but also on what ω used to be: the system remembers what path it has been following as ω has been changing.

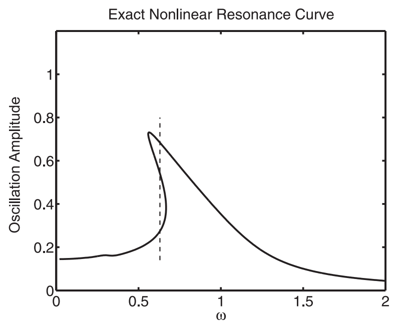


Figure 13.2

Matlab Calculation

As you were dropping terms in the previous section with wild abandon you may have wondered if the analysis was going to be any good. Well, the point of having a computer is to do difficult problems right, so let's see if we can coax Matlab into finding the correct steady response of the oscillator to the driving force, including both the full nonlinear form of the force and damping too.

The straightforward way to see how the oscillator responds is to give it some initial conditions $x(0)$, $v(0)$ and run it for a long time to see which final steady state it settles into. Unfortunately, doing it this way takes a long time, and it also won't allow us to explore the middle unstable branch of the resonance curve.

A better way to do this calculation is to notice that if the oscillator is to respond to the drive in a steady way then $x(t)$ and $v(t)$ of the oscillator must both be periodic with the same period as the drive: $T = 2\pi/\omega$. So for each driving frequency ω

there must be a set of initial conditions $x(0, v(0)$ that would produce the perfect final steady state. We could imagine looking for this special state by repeatedly choosing initial values for $x(0)$ and $v(0)$, solving the differential equation from 0 to T, and then checking to see if $x(T)$ and $v(T)$ match their initial values. If they don't we could keep adjusting $x(0)$, $v(0)$ until they do. If we're lucky, solving the differential equation over the short interval $[0, T]$ will make up for the initial value searching we do, and make the calculation run faster than just waiting for steady state to happen. But to be lucky, we will need a clever way of quickly finding the correct initial conditions, and Matlab's minimizer `fminsearch` turns out to do the job.

This will be the most involved programming project of the semester, so we will try to guide you through it step by step. You will not finish this lab today, but don't panic–we will continue this work into next week.

OK, it's time to write your own Matlab code to make a figure like Fig. 13.2 You will have to write a main script and two function M-files, and they are a little involved. The instructions that follow are a kit to solve this problem. We won't tell you everything you need to know, but most of it is here. As you go along you will be building a chain of 5 M-files and Matlab commands that call each other, like this:

```
lab13.m -> fminsearch -> leastsq13.m -> ode45 -> rhs13.m
```

Keep this chain in mind as you work through the rest of this lab.

P13.4 Write a main script called `lab13.m` that starts ω at ω_1 and ends it at ω_2 taking N steps along the way. This is how we will track the response of the oscillator as we scan ω through resonance going either up or down in frequency. This script, which we call `lab13.m`, should do the following things:

(a) Declare the driving frequency ω, the driving amplitude A, and the damping constant γ as global variables. After they are declared give them values $A = 0.136$ and $\gamma = 0.2$. Also declare the variable S to be global (you will see why later).

(b) Use `input` commands to enter values ω_1, ω_2 to define the driving frequency scan interval, and also use `input` to enter N, the number of scan points. Then calculate the step size in the scan $h = (\omega_2 - \omega_1)/N$ and use it to define an array of ω-values called `wscan` that will be used in the scan.

(c) Use `input` commands to enter initial guesses for $x(0)$ and $v(0)$.

(d) Write a loop that steps through the `w` array of ω values and asks Matlab's minimizing utility `fminsearch` to refine the initial guesses for $x(0)$ and $v(0)$ to find the correct initial conditions to make $x(t)$ and $v(t)$ be periodic with period $T = 2\pi/\omega$, the same as the period of the driving force. An example of what your loop should look like is given below.

Listing 13.1 (loop.m)

```
guess=[x0;v0];
for j=1:length(wscan)
    w=wscan(j)  % no semicolon so w will print on the screen
    T=2*pi/w;
    optionfmin=optimset('TolX',1e-6);
    answer=fminsearch(@leastsq13,guess,optionfmin) % no semicolon so that
                                                   % the correct (x0,v0)
                                                   % will be on the screen
    S   % print S, the minimum error from fminsearch, on the screen

    % now that fminsearch has found initial conditions for
    % a periodic orbit, calculate the orbit and find its
    % amplitude Amp(j) by finding the maximum value of x(t)
    % over a period.
    optionode=odeset('RelTol',1e-6);
    [t,yode]=ode45(@rhs13,[0,T],answer,optionode);
    x=yode(:,1);v=yode(:,2);

    % save the amplitude at each step of the scan
    Amp(j)=max(x);

    % plot the solution to make sure it is periodic
    plot(t,x,'b-',t,v,'r-')
    title(sprintf('\\omega = %g ',w));
    xlabel('t');ylabel('x(t), v(t)');

    pause(.1)

    % use the answer we just found as the guess to pass into fminsearch
    % when we go back up for the next value of w
    guess=answer;
end
% After the scan is finished plot amplitude vs. frequency
plot(wscan,Amp)
```

The code above is just a bare-bones version. As you use it you will encounter various problems which you will want to fix by adding new code to the loop above.

You will notice that in the loop in (d) `fminsearch` needs an M-file `leastsq13` and that `ode45` needs an M-file `rhs13`. Let's start with `leastsq13`. You may recall that `fminsearch` is a Matlab routine that varies a set of parameters until it finds a special combination of them that minimizes a scalar function of the parameters. In our problem we want to choose a scalar function that is a minimum when the orbit returned by `ode45` is periodic, i.e., the orbit satisfies $x(T) - x(0) = 0$ and $v(T) - v(0) = 0$. A scalar function (chosen to have units of velocity) that has a minimum when these conditions are satisfied is

$$S = \omega^2 (x(T) - x(0))^2 + (v(T) - v(0))^2$$

and this is the quantity that we want `leastsq13` to calculate so `fminsearch` can minimize it.

P13.5 (a) Write `leastsq13.m` following the pattern in Listing 13.2. We have left some blanks for you to fill in, indicated by the dots.

(b) Finally, create the function `rhs13` called by both the main script and by `leastsq13`. This is just the usual right-hand side function used by `ode45` and should look Listing 13.3. Make sure you use the force in Eq. (13.2) instead of the simple approximate force $F \approx -x + 3x^3$.

(c) Now it is time to debug and test the code we have written so far. Use $A = 0.136$ and $\gamma = 0.2$. To test the code, set $\omega_1 = 1.8$, $\omega_2 = 2$, and $N = 5$ so that we are working well above the resonance near $\omega = 1$. At this driving frequency the amplitude should be low and $x(t)$ should be 180 degrees out of phase with the driving force, so we expect something like $x(0) = -A/\omega^2$ and $v(0) = 0$ to give a periodic solution. Run and repeatedly debug the code you have written until you obtain a reasonable result. As a numerical check, with $\gamma = 0.2$ and $\omega = 2$ the correct initial values are $(x_0, v_0) = (-0.0423, 0.0225)$.

Listing 13.2 (leastsq13.m)

```
function S=leastsq13(guess)
global A w gamma S;

T=2*pi/w;
% fminsearch will give this function various initial conditions
% [x0;v0] through the input variable guess.  This function's
% job is to solve the differential equation with these proposed
% initial conditions and return a value for S so that fminsearch
% can know how close it is to choosing [x0;v0] that make S be
% a minimum. The differential equation is to be solved over time
% interval [0,T] with relative tolerance 1e-6.  After the solve,
% which is of the form [t,yode]=..., unpack yode into x and v.
.
.  write your own code here to call ode45
.
% now subtract the final and initial values to build S
% note that x(end) is Matlab syntax for the last element of x.
S=w^2*(x(end)-x(1))^2+(v(end)-v(1))^2;
return
```

Listing 13.3 (rhs.m)

```
function F=rhs13(t,y)
global A w gamma;

F=zeros(length(y),1);
% load the column vector with the derivatives of x and v
x=y(1);v=y(2);
F(1)=v;
F(2)=....(put the differential equation here)
return
```

Now we are going to use this set of codes to explore the true hysteresis diagram which we approximated in the Mathematica analysis.
Important: Use $A = 0.136$ throughout the rest of this lab.

P13.6 (a) Set $\gamma = 0.015$ and do a scan from $\omega_1 = .1$ to $\omega_2 = .4$. Use $N = 80$. Now that you are at low driving frequency using $x(0) = -A/\omega^2$ is a very bad idea. Use a small value instead, like $x(0) = 0.15$. $N = 80$ should be a large enough value that you can see the predicted resonant behavior at $\omega = 1/3$, as well as another small resonance at $\omega = 1/5$. This second resonance is expected because of the appearance of $\cos 5\omega t$ among the higher-order terms we neglected earlier. Other terms like $\cos 7\omega t$, $\cos 9\omega t$, etc. were also neglected. Can you see something at $\omega = 1/7$? At $\omega = 1/9$? Why are they harder to see, and why are the resonances shifted slightly from these values? (Zoom in on the plot to see them).

(b) Now run your loop from low frequency at $\omega_1 = 0.2$ up to high frequency at $\omega_2 = 2$ with the following value for the damping constant: $\gamma = 0.2$. Use $N = 40$. You will find that with this damping value there is no hysteresis, although the slightly bent-over shape of the resonance curve gives a hint of what is about to happen.

Now for the main event.

P13.7 Set the damping constant to $\gamma = 0.15$ and run your loop for 5 different sets of values of ω_1 and ω_2, as follows.

(a) Use $\omega_1 = 2$ and $\omega_2 = 0.7$ to follow the upper curve down in frequency from the low-amplitude states at high frequency. Use $N = 30$. Make sure that ω, (x_0, v_0), and S are printing on the screen so that you can scroll back up if you need to see them.

When your scan has finished, save `wscan` and `Amp` to disk so that later you can read all of the scan results back in and overlay their plots to make your own version of Fig. 13.2. For instance, if you wanted the results of this first scan (both amplitude and angular frequency) stored in a file named `scan1.mat`, you would type this at the Matlab command prompt (or put this code at the bottom of your `lab13` script:)

```
save scan1 Amp wscan
```

You will encounter a problem as you do this scan: `fminsearch` will sometimes lie to you by returning reasonable amplitudes when the value of S is too large to correspond to a solution. The problem is that `fminsearch` is a *minimizer*, not a solver, so when it finds a local minimum in parameter space, it thinks it has done its job properly and it quits. To take care of this problem, after the call to `fminsearch` in `lab13.m` test the value of S to see if it is too big. If S is larger than about 10^{-8} the solution has failed and you should break out of the scan loop using the `break` command.

Unfortunately, breaking out will now cause another problem: the frequency scan array `wscan` and the corresponding amplitude array `Amp` do not have the same length. A simple way to fix this problem is to redefine `wscan` to be the right length to match `Amp` after the bottom of the loop, like this:

```
wscan=wscan(1:length(Amp));
```

Another problem you will encounter is that it may be hard to stay on the upper curve because your code will want to jump down to the lower one before very much hysteresis is observed. This is caused by not giving `fminsearch` a sufficiently accurate initial guess. You can fix this by using linear extrapolation, like this:

Before the call to `fminsearch` set an old answer variable equal to the last `answer` (which contains refined values of `[x0;v0]`) so that it will be remembered before a new answer is found:

```
oldanswer=answer;
```

Then at the bottom of the loop replace the code

```
guess=answer
```

with linear extrapolation:

```
guess=2*answer-oldanswer
```

And finally, just above the top of the scanning loop initialize the `answer` variable so that extrapolation will work the first time by changing the line

```
guess=[x0;v0];
```

to

```
guess=[x0;v0];
answer=guess;
```

This will only work well if you supply a pretty good initial guess for (x_0, v_0).

(b) Now use $\omega_1 = 0.7$ and $\omega_2 = 0.55$ to finish the upper curve. Use $N = 30$. To start the code you will need to scroll up and see what (x_0, v_0) were in your first scan when you got to $\omega = 0.7$. Remember to save `wscan` and `Amp` to disk with a different file name than you used in (i).

(c) Now use $\omega_1 = 0.2$ and $\omega_2 = 0.7$ to follow the lower curve from low frequency up to the point where it curves up and jumps up to the upper branch. Use $N = 30$. Does this upward jump occur anywhere near the value predicted by the Mathematica plots you made earlier? Does the jump occur at a higher value of ω than the last good value on the upper curve in (ii)? If it does, then you have achieved hysteresis. Congratulations. Remember to save `wscan` and `Amp` to disk with different file names than you used in (i) and (ii).

(d) To explore the middle branch choose $\omega_1 = 0.6$, $\omega_2 = 0.55$, and use $(x_0, v_0) = (0.35, 0.31)$ with $N = 20$. This will produce the left half of the middle branch.

(e) Now choose $\omega_1 = 0.6$, $\omega_2 = 0.7$, and use $(x_0, v_0) = (0.35, 0.31)$ with $N = 20$. This will produce the right half of the middle branch, and is the last of the scans.

(f) Make the complete picture of the resonance curve by writing a script that overlays the results of all 5 scans on the same figure using the `load` commands given in (a) above.

To do this you will need to load each scan and plot its results. Reading them back in is a little tricky because Matlab not only saves the data in its `.mat` files, but it also stores the original name of the variable and restores it when you read it back in. For example, if you use the command

```
load scan1
```

then the amplitude and angular frequency values for scan 1 would be stored in the variable names `Amp` and `wscan`. If you then load `scan2`, its values for `Amp` and `wscan` will replace those for `scan1`. So the script to overlay all of the plots should be written like this:

```
load scan1
plot(wscan,Amp,'b*') % plot Amp vs. wscan from data set 1
hold on
load scan2
plot(wscan,Amp,'g*') % plot Amp vs. wscan from data set 2
.
.
.
```

Look up `save` and `load` in online help to see more details.

(g) Modify your code just slightly to study the stability of the oscillator states you found in this problem. In `lab13.m` change the code `[0,T]` in the line that calls `ode45` to `[0,100*T]` so that we can see the long-time behavior of the oscillator state found by `fminsearch`. Also change the line `pause(.1)` to `pause` so that `lab13.m` won't move on to the next point in the scan without your permission.

(i) Start a scan with $\omega_1 = 0.3$, ω_2 set to anything else, and $N = 2$. We don't care about ω_2 because we will only be looking at the first point, $\omega = \omega_1$. When the plots of $x(t)$ and $v(t)$ appear examine them carefully and decide whether this oscillator state on the lower curve is stable or unstable. Then use `Ctrl-c` to kill this run and move on to the next one.

(ii) Now start a scan with $\omega_1 = 1$ and see if this state on the upper curve is stable or unstable.

(iii) Finally, start a scan with $\omega_1 = 0.6$ using $(x_0, v_0) = (0.35, 0.31)$ and decide if this state is stable or unstable. You should find that it is unstable, but that the instability *saturates* by settling into a state on either the lower curve or the upper curve, both of which are stable. If you jumped up to the upper curve, modify the initial conditions in the line in `lab13.m` that calls `ode45` from `answer` to `0.999999*answer` and run it again. Or if you jumped down, use the factor `1.000001` instead. You should find that whether it jumps up or down depends very sensitively on the initial conditions, so if we were experimentally studying a system with this behavior we would probably not even know that there was a middle branch.

Well, that's it. You have now reproduced the bent-over resonance curve with hysteresis sketched in Fig. 13.2 and have found that the upper and lower branches are stable, but that the middle branch is unstable. And you have also found that when the damping is small there are small sub-resonances at $\omega = \omega_0/3$, $\omega = \omega_0/5$, etc.

Index