Ross L. Spencer and Michael Ware
Department of Physics and Astronomy
This is a laboratory course on using computers to solve ordinary differential equations of the kind that show up in physics. The course has three objectives.

1. The student will learn the basic ideas of nonlinear dynamics.

2. The student will learn the basics of the Matlab programming language.

3. The student will learn how to solve nonlinear differential equations of the kind that appear in nonlinear dynamics using both Maple and Matlab.

These three objectives are to be met by learning and applying programming techniques in Maple and Matlab to illustrate the ideas of nonlinear dynamics during weekly lab periods. The instructor and a teaching assistant will be present to highlight the important ideas and to coach the students through the laboratory exercises. This is not an independent study course. Experience has shown that students who try to work through this material on their own 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.

It is assumed that the student is familiar with Maple from the start so that our study of differential equations can begin in this language. For about the first half of the course the labs consist of (i) Maple exercises involving differential equations and (ii) assignments to work through sections of the text *Introduction to Matlab*. During the second half of the course we will leave Maple and 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. Students interested in mastering this subject should pursue independent reading and take more complete courses in the subject.

Students should consider buying the student versions of Maple and Matlab while you have a student ID and they are cheap. It would be very helpful for you as you do your homework and research to have these tools on your own computer.

Please contact Michael Ware (N263 ESC, 422-2186, michael_ware@byu.edu) with comments and suggestions on this manual.
# Contents

Preface                                          i  
Table of Contents                                iii  
1 Matlab and Differential Equations in Maple     1  
2 Matlab, Maple, and Baseball                    3  
3 More Matlab and Maple                          5  
4 Matlab, Maple, and Phase Space                 7  
5 Calculus with Matlab                           11  
6 Matlab and Fourier Transforms                  13  
7 Matlab and Pumping a Swing                     23  
8 Matlab and the Pendulum                        27  
9 Matlab, Differential Equations, and Chaos      31  
10 Coupled Nonlinear Oscillators                 37  
11 The Pendulum with a High Frequency Driving Force 41  
12 Two Gravitating Bodies                        45  
13 Hysteresis in Nonlinear Oscillators (two weeks) 51  
Index                                           63
Lab 1

Matlab and Differential Equations in Maple

Why Study Matlab?

In Physics 230 you learned how to tackle physics problems using Maple, and by now you have probably come to appreciate its utility. Maple has an amazing degree of flexibility: it does hard integrals, keeps track of messy algebra, knows a myriad of special functions, has over 100 data types, calculates numerically with as many digits as you want, and handles a host of other tasks. With some time and effort, you could use Maple to solve every problem in this course. Why, then, do we bother with Matlab?

When it comes to numerically modeling dynamical systems, Maple’s incredible flexibility comes back to bite you in one extremely important area: speed. Maple is the numerical equivalent of a Winnebago. Hauling all of Maple’s utility into a numerical modeling problem is like hauling your shower, bed, microwave, satellite TV, and kitchen sink up a steep mountain grade. Its OK once in a while (if you’re not in a hurry), but not something you want to do repeatedly. This usually makes Maple a poor choice for all but the simplest of numerical modeling problems.

The 1960s muscle car of the numerical world is Fortran\(^1\). It may look kind of dated, but step on the gas and that thing computes! Once you get over the thrill of speed, however, your notice that an old muscle car also comes with other features: a mono AM radio, two windows for air conditioning, and the necessity to become (or make friends with) a good mechanic. In other words, with Fortran you get the speed, but pay the price in increased coding time and a lack of some of the built-in extras (e.g. plotting and bundled, well-documented function libraries). If you have a serious calculational challenge, you’ll want to take a look at Fortran, but it is a bit too involved for the purposes of this class.

Now enter Matlab, a modern mid-sized sedan of numerics. Matlab computes much faster than Maple, and comes with a large array of built-in functionality (plotting, special functions, etc.). While Matlab gives up some speed in comparison to Fortran, the code development time in Matlab is so much shorter that for typical computational jobs (processing data and small to medium scale modeling) it is usually an excellent choice of tools, and the one that we will primarily use to model dynamical systems in this class.

We will start by studying some simple systems using the numerical abilities of Maple while we learn Matlab through a series of basic exercises. (Although slow, Maple’s numerical tools can be useful, and you ought to know how to use them. We will also have plenty of opportunity to use Maple’s symbolic math abilities.) As you gain familiarity with Matlab, we’ll start using it to model more complicated dynamical systems.

As we go through the initial Matlab exercises, please keep in mind that even though you could often do the exercises in Maple (sometimes more simply than in Matlab), the point is to learn how to drive a new tool, not simply to get the exercise done. We will eventually come to problems that are challenging but straightforward in Matlab, but which would be

---

\(^1\)Fortran is an acronym derived from The IBM Mathematical Formula Translating System, developed back in the 1950s. Modern Fortran has evolved significantly since then, and is currently the tool of choice for most serious computational physics.
downright painful in Maple. These types of problem are quite common in most physics careers, and you will be glad to have the tools to deal with them.

OK, enough with the automotive analogies. Let’s get started:

1.1. (a) Read and work through Introduction to Matlab, Chapters 1-3. Type and execute all of the material in typewriter font.

(b) Write a Matlab program 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., $3.1415927e+00$.

1.2. Read and execute the examples in Introduction to Maple for Physics Students, Chapter 7, the first two sections on first and second order differential equations. Along the way do problems 7.1, 7.2, 7.3, 7.4, and 7.6 in Introduction to Maple for Physics Students. This is a computer book, available at the Physics 230 or 330 web sites at www.physics.byu.edu. (We will come back to the harmonic oscillator in Labs 3-5. You can read about the mathematics of the simple harmonic oscillator in the references below.²)

Lab 2

Matlab, Maple, and Baseball

2.1. (a) Read and work through Introduction to Matlab, Chapters 4-6. Type and execute all of the material in typewriter font.

(b) 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.

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

\[
 f(x, y) = e^{-|x-y|} \left( 1 + \frac{1}{5} \cos \left( \frac{x}{2} \right) \right) \left( 1 + \frac{4}{3 + 10y^2} \right) . 
\]

Plot it over the xy square from -5 to 5 in \( x \) and from -6 to 6 in \( y \) and label the \( x \) and \( y \) axes. Look carefully at your plot and make sure the labels correspond to the correct axes.

(d) Use Matlab’s `dot` command to find the angle between the vectors \( \mathbf{A} = [1, 2, 3] \) and \( \mathbf{B} = [-3, 2, 1] \). (You will need to know how to find the magnitude of a vector to do this problem. Look in the index of Introduction to Matlab under “Magnitude of a Vector.”) 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 \( m = 2.3 \).

2.2. Read and execute the examples in Introduction to Maple for Physics Students, Chapter 7, the sections titled Systems of differential equations and Numerically solving differential equations.

2.3. Do Problem 7.8 in Introduction to Maple for Physics Students.

(Lab continues on the next page)
2.4. (a) Do the air baseball problem in *Introduction to Maple for Physics Students*, Problem 7.9. With air friction included the trajectory is no longer parabolic, as illustrated in Fig. 2.1. You should produce similar trajectories as you do this problem. For more information about the subject of air drag see below. ¹

(b) Physicists studying baseball say that backspin (which makes the ball float by deflecting air downward through the Bernoulli effect) is also essential for record hits. One expert says 105 mph (47 m/s) with optimal backspin gives a range of about 400 feet (122 m). Try several initial angles to find the maximum range you can get from your model in part (a) (which doesn’t include backspin) with an initial velocity of 105 mph. The amount you fall short shows the importance of backspin.

3.1. (a) Read and work through *Introduction to Matlab*, Chapters 7-8. Type and execute all of the material in typewriter font.

(b) Define the Matlab matrices

\[
A = \begin{bmatrix} 1,2,3;4,5,6;7,8,9 \end{bmatrix}
\]

and

\[
B = \begin{bmatrix} 1,4,5;9,6,3;2,3,1 \end{bmatrix}.
\]

Also define the row vector

\[
v1 = [1,1,2]
\]

and the column vector

\[
v2 = [0.40824829 ; -0.81649658 ; 0.40824829].
\]

(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) 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. 3.1). 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.

![Graph](image)

**Figure 3.1** This third-order fit is not very good; you will do better.

3.2. Read and execute the examples in *Introduction to Maple for Physics Students*, Chapter 7, the section titled *How does a differential equation make a curve?* and do problems 7.12 and 7.13.

3.3. Do Problem 7.5 in *Introduction to Maple for Physics Students*. (For more information about the driven, damped harmonic oscillator, see below.\(^1\))

---

Lab 4

Matlab, Maple, and Phase Space

4.1. (a) Read and execute the examples in *Introduction to Matlab*, Chapter 9.

(b) Write a loop that prints the first 20 terms of the recursion relation

\[ \begin{align*}
    a_1 &= 1 \\
    a_{n+1} &= \left( \frac{n}{(n-1/2)(n+1/2)} \right) a_n.
\end{align*} \]

Plot them with semilogy to see how they fall off with \( n \), i.e., find out if \( a_n \) falls off like \( e^{-n} \), \( 1/n^p \), \( 1/n! \), etc.

(c) Write a loop that performs the sum

\[ S(N) = \sum_{n=1}^{N} \frac{1}{n}. \]

Run it for increasingly large \( N \) and show by numerical experimentation that the sum diverges, but only weakly. Then modify your code to compute \( S(N) - \ln(N) \) and show that as \( N \) becomes large this quantity approaches a limit. This limit is called \( \gamma \), or Euler’s constant. (Note: 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.) Look up the value of this constant in Maple and compare it with your Matlab result.

(d) Define an array of \( x \)-values like this: \( x=0:.01:5 \). Then use loop and logic commands to load \( f \) with the function

\[ f(x) = \begin{cases} 
    e^x, & 0 \leq x < 1 \\
    e \times \cos(x-1), & 1 \leq x \leq 5
\end{cases}, \quad (4.1) \]

and plot \( f(x) \) vs. \( x \).

Phase Space

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, here 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. You will make several pictures like this during the course of this lab.

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:

\[
\frac{dx}{dt} = v \quad ; \quad \frac{dv}{dt} = -\omega_0^2 x .
\] (4.2)

4.2. Use Maple to plot \(x(t)\) and \(v(t)\) for a simple harmonic oscillator (on the same graph) from \(t = 0\) to \(t = 3\) by choosing \(\omega_0 = 2\pi\) and \(x_0 = 1\); make three plots using three different values of \(v_0\): \(v_0 = 0\), \(v_0 = 40\), and \(v_0 = -20\). Make sure you understand qualitatively why the plots look like they do.

Now modify your plot command so that it plots \(v(t)\) vs. \(x(t)\) using \(t\) as a parameter, i.e., make a phase-space plot of the orbit. The change from a 2-trace plot to a parametric plot is very simple. If the plot of \(x(t)\) and \(v(t)\) is done with the command

\[
\text{plot}([x(t),v(t)],t=0..3)
\]

then to make a parametric plot all you have to do is move the right square bracket over to include \(t=0..3\) as well:

\[
\text{plot}([x(t),v(t),t=0..3])
\]

Overlay all three parametric phase-space plots on the same frame by assigning each of the three plots to the variable names \(p_1\), \(p_2\), and \(p_3\), and then using \texttt{display} to combine them, like this:

\[
p1:=\text{plot}(...)
p2:=\text{plot}(...)
p3:=\text{plot}(...)
\text{with(plots):display(p1,p2,p3)}
\]

When you make this work you will be looking at three different orbits in phase space. Physicists often use this \([x,v]\) space to visualize dynamical systems.

If you think of \(dx/dt\) and \(dv/dt\) in Eq. (4.2) 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 \(v\).

Now you will make some flow plots using Maple. (They can also be made by Matlab using the \texttt{quiver} command, but in the interest of time we will only do it in Maple.)

4.3. (a) Maple has a built-in command that draws both trajectories and flow arrows: \texttt{DEplot}. To learn how to use it go to the Physics 230 book and work through the material following Problem 7.9. There you will see how to simultaneously produce a trajectory and a flow field. Explore online help for \texttt{DEplot} and make a three-trajectory phase space plot with flow arrows following the pattern in problem 4.2. Use \(\omega_0 = 2\pi\) and run for just 0.5 s of time. You will have to think carefully about how big the plot window should be in \(x\) and \(v\). When you understand it, explain it to the TA.

(b) Now change your plot in (a) to include linear damping with damping constant \(\gamma = 1.3\):

\[
\frac{dx}{dt} = v \quad ; \quad \frac{dv}{dt} = -\omega_0^2 x - \gamma v .
\] (4.3)

Explain how the arrows show what the solution will do.

(c) Change your plot in part (b) to include air-damping with \(\gamma = 1.3\) and \(\bar{v} = 1\) in the following differential equation:

\[
\frac{dx}{dt} = v \quad ; \quad \frac{dv}{dt} = -\omega_0^2 x - \gamma |v|/\bar{v} .
\] (4.4)

Explain how this picture looks different from the one in part (b), and why.
Lab 5

Calculus with Matlab

5.1. (a) Read and work through Introduction to Matlab, Chapters 10-11. Type and execute all of the material in typewriter font.

(b) Consider the simple function \( f(x) = e^x \). Show by numerical experimentation in Matlab at \( x = 1 \) that the centered difference approximation to the first derivative in Section 10.1 of Introduction to Matlab is indeed more accurate than the forward difference formula in the same section. (The forward formula just involves \( x + h \) and \( x \) while the centered one uses \( x + h \) and \( x - h \)).

(c) Write a loop that decreases \( h \) from 1 to \( 10^{-20} \) by dividing successively by 2 and calculate the error of the centered derivative formula (i.e., \( \text{abs}(fp/exp(1)-1) \)) where \( fp \) is the numerical derivative) at each \( h \). Make a loglog plot of the error vs. \( h \) (it should have a minimum around \( 10^{-5} \)). Show your plot to the TA and explain why very small values of \( h \) always make the approximate derivative be wrong, giving zero instead of a good approximation to \( f' \) (the section below may be helpful for this).

Roundoff

The effect illustrated in exercise 5.1(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: \( b - a = .000000006632 \). 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 Maple 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.

5.2. Use the simple mid-point rule code in Sec. 10.2 of Introduction to Matlab to numerically do the integral

\[
\int_0^2 x^2 e^{-x} \cos x dx .
\] (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 integral in Maple. Then do the same integral using Matlab’s \texttt{quadl} command by putting an in line function in your midpoint program and passing it into \texttt{quadl}.

5.3. (a) The Fibonacci sequence \( F_n \) is defined by

\[
F_1 = 1 \quad ; \quad F_2 = 1 \quad ; \quad F_n = F_{n-1} + F_{n-2} \quad \text{for} \quad n \geq 3
\] (5.2)
Figure 5.1 The function plotted in Problem 5.2

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

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

\[
h = \frac{4\pi}{50000}; \\
x = [-2\pi:h:2\pi];
\]

Then write a loop that evaluates the Fourier-like series

\[
G(x) = \sum_{n=1}^{100} \frac{\cos(F_n x)}{F_n}.
\]

Plot this function vs. \( x \) and carefully observe its shape (this function is shown in Fig. 5.1 above). 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 \(^1\) and such curves are important in chaos theory.

5.4. (a) Read and work through Introduction to Matlab, Chapter 12. Type and execute all of the material in typewriter font.

(b) Write a loop that makes a list of the first 40 zeros of the Bessel Function \( J_0(x) \). Find these zeros by writing a loop to load them using Matlab’s \texttt{fzero} command. You will have to give \texttt{fzero} a search range instead of just an initial guess, and this will be easier if you remember that the zeroes of \( J_0(x) \) are separated by about \( \pi \). Look through your list of zeros and make sure that there are no repeated values.

\(^1\) 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 6

Matlab and Fourier Transforms

Note: we will be talking about frequencies in this lab and in physics they come in two types: frequency $\nu$ (measured in Hz, or cycles/second) and angular frequency $\omega = 2\pi\nu$ (measured in radians/second). Both are usually called frequencies, but can be kept separate by looking at the context and at the symbols used. There are lots of symbols for frequency ($f$ and $\nu$ are common choices) but $\omega$ is exclusively used for angular frequency.

6.1. Read and work through Introduction to Matlab, Chapter 13. Type and execute all of the material in typewriter font.

6.2. First, let’s just practice interpreting the spectrum of some data. On the class web site is a file called “Beethoven.wav” that has the first four notes of Beethoven’s 5th symphony. If your computer has speakers, go ahead and listen to it (if not, get someone in the class to sing it for you). Let’s look at the signal and its spectrum.

Loading a wav file is slightly different than loading a text file. The code

$f = \text{wavread('beethoven.wav');}$

puts the waveform in $f$. The time series isn’t in the file, but you can construct it by noting that this particular recording was sampled at a rate of 11.025 kHz (i.e. 11025 points/second in $f(t)$). Find the timestep $\tau$, construct the time series, and plot the signal versus time. Then take the $\text{fft}$ and plot the power spectrum versus $\nu$ (not $\omega$) over the range 0-1000 Hz. Your spectrum should look like this:

Now we need to make sense of the spectrum. The short notes at the beginning are the note “G” (repeated three times) played in octaves 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 decide what their relative amplitudes mean.

Plotting the power spectrum with $\text{semilogy}$, as is done in the right plot, is a great way to see small structure in your spectrum. You should always take a look at the log plot to see what structure that may not be evident on a linear scale. In our spectrum we see smaller peaks around 234 Hz, 468 Hz, 624 Hz, 800 Hz, and 936 Hz. Come up with an explanation for each of these peaks by thinking about vibrational modes on a
string. When you are ready, explain to the TA which peaks are associated with which notes in the four note theme and why some peaks are big and some are small.

Big Hint: The “fundamental” mode of vibration for a string has nodes at either end (where the string is fastened) and an antinode in the middle. However, it can also vibrate in other “harmonic” modes with nodes at places besides the ends. The second harmonic mode has a node in the middle of the string, the third harmonic has two nodes that split the string in thirds, etc. The frequencies of these modes are: fundamental = ν₀, second harmonic = 2ν₀, third harmonic = 3ν₀, etc. When a musician drags a bow across a string, she excites mostly the fundamental, but the higher harmonics are also present.

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 which we can understand by using the fft. (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.) To avoid mathematical complications we will use here a simplified version of the uncertainty principle applied to time signals and their corresponding spectra in ω-space.

Suppose that we have a time signal which has a frequency ω₀, but which only lasts for a finite time Δt. For example, consider the function of time

\[ f(t) = \sin(\omega_0 t)e^{-(t-t_0)^2/W^2} \]  

(6.1)

The function multiplying the sine is called a Gaussian, and it makes a “bump” centered at t₀ with a width controlled by W. Because the signal oscillates at angular frequency ω₀ we would expect to see a peak in the spectrum at ω = ω₀. But we will also find that the frequency peak has a well-defined width, and this width is related width of the signal in time through the uncertainty principle. To make the problem a little more concrete, let’s study the spectrum of a low frequency signal with ω₀ = 200 S⁻¹ (ν ≈ 32 Hz, about the frequency of the lowest C on a piano).

Choosing an Appropriate Time Series for the FFT

Before we can study the problem numerically, we have to design a time series for this function so that Matlab can calculate its spectrum correctly. Designing a time series means to choose values of τ and N that will allow the fft to display the features of the spectrum correctly (no aliasing) and with enough resolution (small enough dω) to see the widths of the peaks. The key spectrum parameters are dω and ωc, which are given by

\[ d\omega = \frac{2\pi}{N\tau} \quad ; \quad \omega_c = \frac{\pi}{\tau} \]

Recall that ωc is the critical or Nyquist frequency, which is the highest frequency we can see without aliasing troubles.
We first need to decide what frequency range we want to study. In our case, we would like to study frequencies around the peak at \( \omega_0 = 200 \). To make sure that the \texttt{fft}'s aliased peaks don’t mix with the real peaks we’ll want to choose a critical frequency such that we can see frequencies well above \( \omega_0 = 200 \) s\(^{-1} \), say up to \( \omega = 400\pi \approx 1250 \) s\(^{-1} \). Once we have chosen an acceptable \( \omega_c \), \( \tau \) is determined by

\[
\omega_c = \frac{\pi}{\tau} \quad \Rightarrow \quad \tau = \frac{\pi}{\omega_c} = \frac{\pi}{400\pi} \approx 0.0025 \text{ s}
\]

For a more complicated signal, you might need to adjust your choice after looking at the spectrum.

The next question to ask is what kind of resolution do we need along the \( \omega \) axis, i.e., what should the value of \( d\omega \) be? We need to choose it small enough that we can separate all of the peaks in the spectrum, and also resolve their widths, if we need this information. In our case, we expect a pretty simple spectrum, and just want to see the width of the peak at \( \omega = 200 \) s\(^{-1} \). We will start with \( d\omega = 0.2 \), which will give us lots of detail since \( 0.2 \ll 200 \).

Once we have decided on a value for \( d\omega \), it determines the value of \( N \) through the relation

\[
d\omega = \frac{2\pi}{N\tau} \quad \Rightarrow \quad N = \frac{2\pi}{d\omega\tau} = \frac{2\pi}{(0.2)(0.0025)} \approx 12566
\]

Because the FFT works better with powers of 2, we raise \( N \) up to the next power of 2:

\[ N = 2^{14} = 16384 \]

Notice that this choice means that time will run from \( t = 0 \) to \( t_{\text{final}} = (N-1)\tau = 40.9575 \) s. If we decide we need better resolution than this choice of \( N \) allows, we can enhance the resolution by sampling for a longer time (i.e., increasing \( N \)).

OK, now we are ready to do a problem.

6.3. Write a Matlab script that uses \( \tau = 0.0025 \) and \( N = 16384 \) to build a time array and the function \( f(t) = \cos(\omega_0 t)e^{-\frac{(t-t_0)^2}{W^2}} \) with \( \omega_0 = 200, \ t_0 = 20, \) and \( W = 1 \) using

```matlab
t=0:tau:(N-1)*tau
f=...(put your code here)
```

Then take the \texttt{fft} and plot both \( f(t) \) and \( P(\omega) \). To see where the uncertainty principle is lurking in these plots, write down the FWHM of the time signal and the frequency peak at \( \omega = 200 \). Then decrease the width of the time signal by changing \( W \) to 0.1 and run the program again. What happened to the width of the frequency peak? Try several other values for \( W \) and deduce a relation between the width of the time signal and the width of the frequency peak. Keep the time signal narrow enough that it goes to zero before reaching the edge of the time window.

We can also analyze this problem analytically. If we define the width in time \( \Delta t \) to be the half-width at half-maximum (HWHM) of the time pulse, then

\[
e^{-\frac{(\Delta t)^2}{W^2}} = \frac{1}{2} \quad \Rightarrow \quad \Delta t = W\sqrt{\ln 2}
\]
The Fourier transform of the time signal is

\[ g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-(t-t_0)^2/W^2} \cos(\omega_0 t) \, dt = \frac{W}{2\sqrt{2}} \left( e^{-W^2(\omega-\omega_0)^2/4} + e^{-W^2(\omega+\omega_0)^2/4} \right). \]

Notice that we get the customary positive and negative frequency peaks (which turn out to be Gaussians in frequency). If we define the width in frequency \( \Delta \omega \) to be the half-width at half-maximum (HWHM) of the positive peak, then

\[ e^{-W^2(\Delta \omega)^2/4} = \frac{1}{2} \quad \Rightarrow \quad \Delta \omega = \frac{2\sqrt{\ln 2}}{W}. \]

(You plotted the power spectrum \( P(\omega) = |g(\omega)|^2 \), so the width you measure on screen will differ somewhat from this.) Notice that as \( W \) increases, \( \Delta t \) gets big and \( \Delta \omega \) gets small, and vice-versa as \( W \) decreases. Their product is given by

\[ \Delta \omega \Delta t = 2 \ln 2 \]

This expression is an example of an uncertainty principle, but it is specific to a Gaussian time signal and our (rather arbitrary) definition of \( \Delta t \) and \( \Delta \omega \). A more general form looks like\(^1\)

\[ \Delta \omega \Delta t \geq (\sim 1) \]

The reason we can’t have a single definite number for this product is that different shapes for the time signal \( f(t) \) give different answers. To see that this is true, take a minute now and change the time function in your \( f(t) \) to \( e^{-|t-t_0|/W} \cos(\omega_0 t) \) and check how the new uncertainty product \( \Delta \omega \Delta t \) compares to the old one.

Show your code and your plots to a TA, and use them to explain the uncertainty principle.

You have actually experienced this idea illustrated in this problem before, but just probably haven’t realized it. You may have noticed that flutes can play much faster than tubas can. 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 \( \omega = 6000 \text{ s}^{-1} \) to produce a spectrum with, say, a 1% width requires \( \Delta \omega = (0.01)(6000) = 60 \). But then the uncertainty principle tells us that this note can be produced by only holding it for the relatively short time of

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

\(^1\)Our HWHM definition for the widths doesn’t give the correct uncertainty principle used in quantum mechanics. Heisenberg’s uncertainty principle is based on widths defined by the standard deviations of the \( t \) and \( \omega \) signals. But this only affects the number on the right that we called \( \sim 1 \); even when done correctly it still has the same basic form \( \Delta \omega \Delta t \geq \text{constant} \). And notice that even when done correctly we still don’t have equality, but only inequality. It seems somehow fitting that the uncertainty principle should itself be uncertain. See N. Asmar, *Partial Differential Equations and Boundary Value Problems* (Prentice Hall, New Jersey, 2000), p. 488-491, and F. S. Crawford, Jr. *Waves, Berkeley Physics Course*, Vol. 3, p. 295-307.
where we have arbitrarily chosen \( \Delta \omega \Delta t = 1 \) to make the calculation. But when a tuba plays a low note around \( \omega = 200 \text{ s}^{-1} \), the same calculation using \( \Delta \omega = (200)(.01) = 2 \) gives a note-duration of only

\[
\Delta t \approx \frac{1}{\Delta \omega} \approx \frac{1}{2} = 0.5 \text{ s}
\]

Now tubas can play faster than this, but if you listen carefully, when they do their sound becomes rather “muddy”, which simply means that the note isn’t very pure, corresponding to a wide peak in its spectrum. The length of the tuba also contributes to the “muddiness” 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. Despite all these setbacks, your ear/brain system helps you out with some fancy signal processing 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. (At tuba frequencies, your ear/brain system can perceive pitch for pulses containing only a few cycles.)

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

### Windowing

Now that we understand the uncertainty principle, we can address the problem of the peaks in Example 13.2a not having the right heights because they are not wide enough. As you may recall, when you plotted the power spectrum for a signal containing 5 frequencies, the sizes of the peaks did not correspond to the squares of the signal amplitudes as they should have. If you don’t remember this, run the example again and try to see the problem.

The reason that this didn’t work right is that the Fourier transform of an undamped sine or cosine is a delta function peaked at the signal frequency. Since this function is so narrow, it is unlikely that its peak sits right at a point in the \texttt{fft} spectrum, so what you think is the peak point is actually slightly off-center, making the peak heights be incorrect.

To fix this problem, we use a technique called \textit{windowing}, where we artificially constrain the signal to be narrow in time. As the signal narrows in time, the peaks in its spectrum broaden so they don’t fall between the cracks in the frequency array. Let’s see how this works:

### 6.4. Modify Example 13.2a in \textit{Introduction to Matlab} so that the height of the largest peak in the power spectrum is normalized to 1 (i.e. plot \( P / \max(P) \) instead of \( P \)) and replace the time signal with the following

\[
f = \sin(t) + 0.5 \sin(3t) + 0.4 \sin(3.01t) + 0.7 \sin(4t) + 0.2 \sin(6t);
\]
To increase the width of the frequency peaks we multiply the time signal by a “window” function. There are many types of windowing functions (see Matlab help on the window command for a list of ones that Matlab has built in). We’ll use the Gaussian window because you’ve already seen it once today in Eq. (6.1) (the function it is windowing is $\sin(\omega_0t)$). To apply the window function to $f$ you use the code

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

The parameter `alpha` is specific to a Gaussian window, and is related to Eq. (6.1) via $\alpha \propto 1/W$ (it is the inverse of the width).

Let’s start off by making the window really narrow so that the peaks are good and broad. To do this, use `alpha=25` and plot the resulting $f$ and normalized $P$. If you look the peaks at $\omega = 1, 4, 6$, the relative amplitudes are now right on. But what happened to the peaks at $\omega = 3, 3.01$? We’ve made the peaks so broad that they’ve smooshed into each other. This phenomenon is called leakage because the window has caused our spectral peaks to leak into neighboring frequencies. Leakage is even present when we don’t multiply $f$ by a windowing function, since there is always an implied rectangular window around a signal that only lasts for a finite time.

Let’s reduce the widths of our peaks by making window broader in time, say `alpha=2`. Now we can see all the peaks, but we are back to the problem of having the relative amplitudes be off because the peak widths are too narrow. Thus when you use the windowing technique there is a tradeoff between how well you know the relative heights of peaks, and how well you can resolve closely spaced peaks. For our signal, we can come to a good compromise between these effects at about `alpha=8`.

Show the TA your plots and code, explain the concepts of windowing and leakage, and tell how they relate to resolving closely spaced peaks.

Wave Propagation With Fourier Transforms

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

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

(6.2)

$g(\omega, x = 0)$ tells you the amplitudes and phase that you need for each of your sinusoids, but in a complex form through Euler’s formula

$$e^{i\omega t} = \cos(\omega t) + i\sin(\omega t).$$

(6.3)

If you want to see how the temporal form of the pulse evolves at other points besides $x = 0$, you have to add up traveling waves of the form $\cos(kx - \omega t)$ and $\sin(kx - \omega t)$. 
The crests and troughs of each frequency travel at the phase velocity, given by $v_p = \omega/k$. In a non-dispersive system we have $k = \omega/c$, so the phase velocity is the same for all frequency components, i.e. $v_p = c$. However, in most interesting situations we have a different dispersion relation $k(\omega)$, so that the peaks and troughs of the frequencies move at different speeds. As the frequency components shift relative to one another, the shape of the pulse changes.

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 terms of complex notation, this means that the spectrum at a point $x$ is related to the spectrum at $x = 0$ through

$$g(\omega, x) = g(\omega, x = 0)e^{ikx} \quad (6.4)$$

If we take an inverse Fourier transform of this spectrum

$$f(t, x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\omega, x)e^{-i\omega t}d\omega = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\omega, x = 0)e^{i(kx - \omega t)}d\omega = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\omega, x = 0) [\cos(kx - \omega t) + i\sin(kx - \omega t)] d\omega \quad (6.5)$$

we can find the form of the pulse at an arbitrary $x$. Note that in the last line we explicitly have the sum of traveling waves that we talked about conceptually. Also note that to make this work, it will be easiest to use the Matlab functions *ft.m* and *ift.m* (as discussed in the last section of chapter 13 in *Introduction to Matlab*) rather than the regular *fft*.

OK, enough theory, let’s do a problem. The dispersion relation for water waves is

$$\omega^2 = gk \tanh(kd) \quad (6.6)$$

where $g$ is the acceleration of gravity and $d$ is the depth of the water. This is a transcendental equation, so we can’t write an explicit $k(\omega)$, but we know how to solve it numerically.

6.5. (a) Before we can solve the dispersion relation numerically, we have to choose an array of frequencies to consider. As we discussed at the beginning, the frequency array is determined by our choice of $N$ and $t_{\text{max}}$. Let’s have a time array that is 200 s wide with $N = 2^{16} = 65536$. Our time and frequency arrays are then given by

- $N = 2^{16}$;
- $t_{\text{max}} = 200$;
- $\tau = t_{\text{max}}/(N-1)$;
- $t = 0: \tau : (N-1) \times \tau$;
- $d\omega = 2\pi / t_{\text{max}}$;
- $\omega = -(N/2) \times d\omega : d\omega : (N/2-1)$;

$$\omega^2 = gk \tanh(kd) \quad (6.6)$$
Notice that we’ve chosen a symmetric $\omega$ array because we will be using $\text{ft.m}$ and $\text{ift.m}$ for this problem.

Numerically solving Eq. (6.6) for $2^{16}$ frequencies is painful, so let’s be smart about this. Plot $\tanh x$. Note that at large positive $x$ we have $\tanh x \approx 1$ and at large negative values of $x$ we have $\tanh x \approx -1$. Now plot $|1 - \tanh x|$ from $x = 0$ to $x = 20$ with semilogy. Note that by $x = 20$, $\tanh x$ equals 1 to 15 digits, so as far as Matlab is concerned they are exactly the same at that point. Thus, for $|kd| > 20$, we can simply use

$$k = |\omega|/g$$  \hspace{1cm} (6.7)

(changing $\omega^2$ to $|\omega|$ captures the sign change). Use this approximation to create a wavenumber array $k$ for all the frequencies in our $w$ array. Set the depth to $d = 10$ m.

(b) We need to fix our approximation for $|kd| < 20$. Using Eq. (6.7) we can translate this into a condition on $k$ to a condition on $\omega$:

$$|k| < \frac{20}{d} \Rightarrow |\omega| < \sqrt{\frac{20g}{d}} \Rightarrow |\omega| < 4.5$$  \hspace{1cm} (6.8)

We can use Matlab’s $\text{fzero}$ command to solve Eq. (6.6) if we put it in the form $\omega^2 - gk \tanh(kd) = 0$. We also need to supply $\text{fzero}$ with a guess that is near the solution. We can get a good guess by noting that for small $x$, $\tanh x \approx x$. Using this approximation with Eq. (6.6) gives the guess $k \approx \omega/\sqrt{gd}$.

Now we just need to loop through and solve it for all the $\omega$’s that satisfy Eq. (6.8). Here is a code fragment with some blanks to get you started:

% Fix the errors for $|\omega| < 4.5$

$$\text{eq} = \text{inline}('\omega^2 - g*k*\tanh(k*d)','k','\omega','g','d');$$

start = . . . % the array index for $w = -4.5$
finish = . . . % the array index for $w = 4.5$
for istep = start:finish
    guess = w(istep)/sqrt(g*d);
    k(istep) = $\text{fzero}(@(k) \text{eq}(k,w(istep),g,d),guess);$
    disp((istep-start)/(finish-start));
end

(c) With our $k(\omega)$ now solved, we are ready to propagate our pulse. At $x = 0$, let’s just have one bump (kind of like a bow wave from a boat) that comes around $t = 20$ s, defined by $f(t, x = 0) = e^{-(t-20)^2/0.5^2}$. Find the spectrum $g(\omega, x = 0)$ of this pulse using $\text{ft.m}$. Next, find the spectrum of the pulse at $x = 30$ by multiplying $g(\omega, x = 0)$ by $e^{ikx}$. Finally, find the $f(t, x = 30)$ by taking the inverse Fourier transform of $g(\omega, x = 30)$ (using $\text{ift.m}$). Plot $f(t, x = 0)$ and $f(t, x = 30)$ on the same axis and explain what you see.

You have probably seen this type of behavior before. When a bow wave leaves the front of your boat, it is mostly just a single bump of water. But it takes a
lot of frequencies to make that bump. As the pulse propagates, the frequencies behave differently and the bump spreads out and develops ripples. At the shore you get a long train of waves rather than a single bump.
Lab 7
Matlab and Pumping a Swing

The so-called parametric oscillator equation is
\[ \ddot{x} + \gamma \dot{x} + \omega_0^2 (1 + \epsilon \cos (\omega_p t)) x = 0 \, . \]  
(7.1)

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 called a parametric oscillator. This may seem strange, but you are very familiar with at least one example of such an oscillator. If you have ever sat in a swing and moved your legs back and forth to “pump” it, you were making a parametric oscillation. The parameter you were changing was the distance from the top of the swing to the center of mass. Using the Lagrangian formulation of mechanics to obtain the equation of motion of a pendulum whose length \( \ell(t) \) is changing with time yields

\[ \ddot{\theta} + \frac{\ell}{\ell} \dot{\theta} + \frac{g}{\ell} \sin \theta = 0 \, . \]  
(7.2)

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

\[ \ell(t) = L_0 + \Delta L \cos \omega_p t \, . \]  
(7.3)

Substituting this form into Eq. (7.2) and expanding in small \( \Delta L \), keeping terms only through first order, yields the following approximate equation of motion

\[ \ddot{\theta} + \omega_0^2 \left( 1 - \frac{\Delta L}{L_0} \cos \omega_p t \right) \sin \theta - 2 \frac{\Delta L}{L_0} \omega_p \dot{\theta} \sin \omega_p t = 0 \, , \]  
(7.4)

where \( \omega_0^2 = g/L_0 \). This is almost like the parametric oscillator equation above (Eq. (7.1)); we just need to make the small angle approximation \( \sin \theta \approx \theta \), drop the term containing \( \dot{\theta} \), and add linear damping. (And a camel would look like a horse if you took off the humps and performed plastic surgery on its ugly face). Well, it’s not that bad; Eq. (7.1) does capture the basic physics of Eq. (7.4) (as we will see in the pendulum lab coming up), so we will start our study of pumping a swing by using the parametric oscillator equation (7.1).

7.1. (a) Begin by asking Maple to solve the differential equation above, Eq. (7.1). Use for initial conditions \( x(0) = 0 \) and \( v(0) = 1 \). You will discover that Maple can’t do it symbolically, so put a line in your worksheet that assigns values to all of the parameters in the differential equation (7.1), solve it numerically, and plot the solution \( x(t) \) for a long enough time that you can see that nothing but a bunch of wiggles is happening. Use \( \omega_0 = 1, \gamma = 0, \epsilon = 0.1, \) and \( \omega_p = 1.1 \). You will probably find that Maple will give you an error having to do with too many

\[^{1}\text{L. D. Landau and E. M. Lifshitz} \text{ Mechanics} \text{ (Pergamon Press, New York, 1976), p. 80-83, and M. Abramowitz and I. A. Stegun} \text{ Handbook of Mathematical Functions} \text{ (Dover, New York, 1971), Chap. 20.}\]
function evaluations. To work around this you need to set an option in `dsolve`, like this: `dsolve(.....,type=numeric,maxfun=-1)`. (The option command `maxfun=-1` tells `dsolve` not to quit because the numerical solver has taken too many steps, in its opinion). You will also need to use a large number of plotting points in `odeplot` by using it in the form `odeplot(...,numpoints=1000)`.

(b) Now let’s see if we can reproduce what you do when you pump a swing. As you come forward you raise your legs, raising your center of gravity. When you go through the bottom you swing them down, lowering it, and when you go back you tuck them up under, raising it again. So during one swing period you raise and lower your center of gravity twice, which means that your parameter oscillation frequency $\omega$ is twice the natural frequency $\omega_0$. Run your Maple code again with $\omega_p = 2$ and watch what happens. (You should reproduce Fig. 7.1 below).

![Parametric Instability](image)

**Figure 7.1** Pumped swing instability.

When you see the swing-pumping effect on your screen, come up to the swing demonstration at the front of the room and see it happen in a real physical system.

Let’s explore this effect a bit more.

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

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

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

To gain some insight into why $\omega_p = 2\omega_0$ is unstable it is helpful to look at the power spectrum of $x(t)$ for the parametric oscillator. A file called `parametric.txt` has been made that contains columns of $t$ and $x(t)$ with $\omega_0 = 1$, $\omega_p = 1.3$, $\gamma = 0$, and $\epsilon = 0.3$. This file was generated with one of Matlab’s numerical differential equation solvers, which work like Maple’s `dsolve` but are somewhat faster. You will learn to use them before too much longer.
7.2. Use Matlab to read the file in and then take it apart into $t$ and $x(t)$. After the file has been read in take the FFT of the data and display its power spectrum using a semilogy plot.

You should immediately notice the big peak at $\omega_0$. This is not a surprise because what we have is an oscillator at this frequency plus a small perturbation of size $\epsilon$. But if you look for a peak at $\omega = 1.3$, you won’t find one, even though there are plenty of other peaks. Your job now is to explain why these other peaks are where they are. Read the hint below, do the math on paper, and explain it to the TA.

**A big hint:**

Since $\epsilon$ is small and the damping is weak, let’s begin by ignoring them both ($\epsilon = 0$ and $\gamma = 0$). Then note that with these simplifications Eq. (7.1) is solved by

$$x_0(t) = Ae^{i\omega_0 t}. \quad (7.5)$$

Now we will proceed by perturbation theory, which goes like this. Make a more precise guess at the solution by writing

$$x(t) \approx x_0(t) + x_1(t), \quad (7.6)$$

where $|x_1(t)| \ll |x_0(t)|$; substitute this form into Eq. (7.1) (using the equation above for $x_0(t)$), simplify, and drop the term of order $\epsilon x_1(t)$ because it involves the product of two very small quantities. The result is the following equation for $x_1(t)$:

$$\ddot{x}_1 + \omega_0^2 x_1 = -\omega_0^2 \epsilon \cos(\omega_p t)Ae^{i\omega_0 t}. \quad (7.7)$$

Notice that this is just the equation of a driven harmonic oscillator, and to see what the driving frequency is use the identity

$$\cos(\omega_p t) = \frac{e^{i\omega_p t} + e^{-i\omega_p t}}{2}$$

to turn the right hand side of Eq. (7.7) into complex exponentials. Just do the math and see what frequencies turn up; can you see them in your spectrum?

This means that in addition to the main term in $x(t)$ that has frequency $\omega_0$ there are two other, smaller, contributions at these other two frequencies. But now these new terms have to be multiplied by the $\cos(\omega_p t)$ when we go to the form

$$x(t) \approx x_0(t) + x_1(t) + x_2(t). \quad (7.8)$$

Can you guess what happens next and find the next two frequencies 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, when the frequencies $\omega_p$ and $\omega_0$ interact to make sum and difference frequencies $\omega_p + \omega_0 = 3\omega_0$ and $\omega_p - \omega_0 = \omega_0$, we produce a component at the resonant frequency $\omega_0$. This allows the system to feed back on itself and become unstable.
This effect now allows us to see why $\omega_p = \omega_0$ is not the most unstable choice. The reason is that with this choice the sum and difference frequencies are 0 and $2\omega_0$, neither one of which is resonant. But in the next order of approximation we will have sum and difference frequencies $0 + \omega_p = \omega_0$, $0 - \omega_p = -\omega_0$, $2\omega_0 + \omega_p = 3\omega_0$, and $2\omega_0 - \omega_p = \omega_0$. Three of these terms resonate with $\omega_0$ so there is a possibility of instability due to nonlinear resonance. But because it takes two approximation steps to get to this resonance, and since each step involves another power of $\epsilon$, this frequency is less unstable.

7.3. Run your Maple differential equation solver with $\omega_0 = 1$, $\omega_p = 1$, $\gamma = 0$, and $\epsilon = 0.1$ and verify that the system is unstable, but less so than it is for $\omega_p = 2$. 
8.1. Read and work through *Introduction to Matlab*, Chapters 14-15. Type and execute all of the material in typewriter font.

The equation of motion of a simple pendulum is

\[ \ddot{\theta} = -\omega_0^2 \sin \theta , \]  

(8.1)

where \( \theta \) is the angle between the pendulum and the vertical direction. (Remember that we are doing calculus here, so \( \theta \) must be in radians.) This is a nonlinear equation, so to this point in your physics career you have probably used the small angle approximation \( \sin \theta \approx \theta \) to simplify Eq. (8.1) into a simple harmonic oscillator. If the pendulum is a simple massless stick of length \( \ell \) with all of the mass at the end of the stick, the small-amplitude oscillation frequency is \( \omega_0 = \sqrt{g/\ell} \). In the more complicated case of an extended object with moment of inertia \( I \) about the pivot point, and distance \( L \) from the pivot point to the center of mass of the object, the small-amplitude frequency is \( \omega_0 = \sqrt{MgL/I} \).

It doesn’t take a very large amplitude before the small angle approximation falls apart, so the simple harmonic oscillator description of a rigid pendulum often falls short. Now that we have the tools to do the problem correctly, we can solve Eq. (8.1) numerically to study large amplitude behavior. This is our first example of a nonlinear equation of motion, and because it is nonlinear the pendulum behaves quite differently from the simple harmonic oscillator. We will explore some of its interesting properties in this lab.

8.2. (a) Let’s begin by proving that Eq (8.1) 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. (Use pencil and paper; Maple will just slow you down). When this happens, we say that the differential equation is nonlinear.

(b) Now let’s find the period of the pendulum by using an energy method.\(^1\) The total energy of a pendulum can easily be found when the pendulum is at the maximum displacement, which we will denote by \( \theta_0 \). At this point, the center of mass is at a height of \( L(1 - \cos \theta_0) \) above the equilibrium position, so the total

---

\(^1\) G. Fowles and G. Cassiday, *Analytical Mechanics* (Saunders, Fort Worth, 1999), p. 318-320
energy is $MgL(1 - \cos \theta_0)$. As the pendulum oscillates, energy shuttles back and forth between kinetic and potential according to

$$\frac{1}{2} I \dot{\theta}^2 + MgL(1 - \cos \theta) = MgL(1 - \cos \theta_0) \quad (8.2)$$

Rearrange this equation to find

$$\frac{1}{2} \dot{\theta}^2 = \omega_0^2 (\cos \theta - \cos \theta_0) \quad (8.3)$$

Separate the variables $t$ and $\theta$ by solving $(8.3)$ for $d\theta/dt$, choosing the positive square root, and then putting the equation into the form $dt = d\theta/f(\theta)$. Then integrate both sides 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).

The resulting $\theta$-integral is difficult, so do it in Maple and try to find a nice expression for the pendulum period in terms of a function $K(m)$, called the complete elliptic integral of the first kind. When you do the integral just as it comes, Maple won’t give you a pretty result. You can help it out by telling it what range the angle $\theta$ takes on, like this: `assume(theta0>0,theta0<Pi)`, where we are using $\theta_0$ as the symbol for $\theta$. This will give you a simpler answer, but not the one we want. To find the simplest answer use a $u$-substitution $u = \theta/2$ and the identity $\cos 2u = 1 - 2\sin^2 u$ for both $\theta$ and $\theta_0$. Remember to change your upper limit from $\theta_0$ to $u_0$, where $u_0 = \theta_0/2$, and use `assume(u0>0,u0<Pi/2)`. Doing the integral with $u$ will give you a nice answer in terms of the complete elliptic integral $K$. Be sure that your answer for the period $T$ depends on $\omega_0$.

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 $\theta_0$. The fact that the natural oscillation frequency depends on amplitude gives the pendulum some interesting characteristics. 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 $\theta_0 = \pi$.

(c) Now set up Eq. (8.1) as a first order set of coupled equations and use Maple’s `dsolve(...,type=numeric)` command to make a procedure that solves it with $\omega_0 = 1$ and initial conditions $\theta(0) = \theta_0$ and $\omega(0) = 0$, where $\omega(t) = \dot{\theta}$. Use `odeplot` to plot the solution for the following values of $\theta_0$: 0.1, 0.5, 1.0, $\pi/2$, 0.9$\pi$, and 0.98$\pi$. For each value overlay a plot of a cosine function of matching amplitude and with a frequency found from your period formula from part (b). Verify that (i) the period formula in part (b) is correct and (ii) for large amplitudes the pendulum motion is not sinusoidal.

(d) Now let’s drive the pendulum with an external torque, like this

$$\ddot{\theta} + \omega_0^2 \sin \theta = \alpha \sin \omega t \quad (8.4)$$

Use $\omega_0 = 1$, $\omega = 1$ (resonance at small amplitudes), 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, then add some linear
damping \((-\gamma \dot{\theta}, \gamma = 0.1)\), to the right-hand side of Eq. (8.4) and watch how the motion changes. Explain the damped behavior and explore how it depends on \(\alpha\). Also vary the driving frequency \(\omega\) in the range \(0.90\omega_0 \rightarrow 1.05\omega_0\) and explain why \(\omega = \omega_0\) doesn’t give the largest amplitude.

![Parametric Pendulum Instability](image)

**Figure 8.1** Parametrically unstable damped pendulum.

(e) Now make the pendulum be parametrically unstable by oscillating its length at \(2\omega_0\) using the equation of motion given in the parametric oscillator lab and printed again here:

\[
\ddot{\theta} + 2\frac{\dot{\ell}}{\ell} \dot{\theta} + \frac{g}{\ell} \sin \theta = 0
\]  

(8.5)

where

\[
\ell(t) = L_0 + \Delta L \cos \omega_p t .
\]  

(8.6)

Run your numerical solution again with initial amplitude \(\theta_0 = 0.1, \dot{\theta}(0) = 0, g = 9.8, L_0 = 1, \Delta L = 0.03, \text{ and } \omega_p = 2\sqrt{g/L_0}\) and recall that \(\omega_0 = \sqrt{g/L_0}\) and watch the pendulum go unstable.

Be careful to run for a long time (like 800 seconds) and use lots of plotting points (\texttt{numpoints}=2000) and you will see something interesting happen. Recall that when you did this with the harmonic oscillator, the oscillation amplitude went to infinity. Why doesn’t this happen with the pendulum?

Now add damping to the equation of motion (use a simple linear damping term \(-\gamma \dot{\theta}\) on the right-hand side of the equation of motion with \(\gamma = 0.01\)) and run the simulation again. (You should see a plot qualitatively like the one in Fig. 8.1 above). Explain physically why the pendulum behaves like this.
Lab 9

Matlab, Differential Equations, and Chaos

In this laboratory you will learn how to solve differential equations in Matlab and then study the evolution of a number of dynamical systems. We will use Matlab’s differential equation solvers extensively for the rest of the course, so make sure you spend the time to clearly understand how they work.

9.1. Read and work through Introduction to Matlab, Sections 16.1-16.5. Copy and execute all of the material in typewriter font.

Important: As you work through this material you will learn how to use an M-file named rhs.m to solve differential equations. When you go on to the problems below, and to the problems in labs 10, 11, etc., please don’t keep using the name rhs.m over and over. Invent a unique name, like rhs9.2.m or rhs9.3.m, and change the call to ode45 to correspond: ode45(@rhs9.3,...). This will make it possible for you to come back later and see how you did each of the problems in the remaining labs.

The van der Pol Oscillator

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

\[ \ddot{x} - \epsilon (\ell^2 - x^2) \dot{x} + \omega_0^2 x = 0. \] \hspace{1cm} (9.1)

This is a simple model differential equation for systems that have an external source of energy which causes the resting state \((x = 0, \dot{x} = 0)\) to be unstable, but which also have sufficient damping that the instability cannot grow to arbitrarily large amplitude. Begin by studying the equation above and convincing yourself that the resting state is indeed unstable, but that large amplitude motion is damped (on average). (Note: you can’t see that \(x = 0, \dot{x} = 0\) 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 \(x = 0.0001, \dot{x} = 0\). The phase-space portrait, 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 obviously not uniformly inward, however, and later in this lab you will see the effect of the squeezed inward flow patterns visible in the figure. (The figure has \(\omega_0 = 1, \ell = 1\), and \(\epsilon = 1\)).

9.2. Use Matlab’s ode45 to solve Eq. (9.1) numerically for several different choices of initial conditions and for the following set of values of \(\epsilon\): 0.3, 1., 20. Use \(\omega_0 = 1.3, \ell = 1\), and use options=odeset(’RelTol’, 1e-4) 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).

---

For each case make both a plot of $x$ vs. $t$ as well as a phase space plot of $v$ vs. $x$. Also plot the power spectrum of $x(t)$ with `semilogy` using an axis command to display the spectrum from $\omega = 0$ to $\omega = 20$.

**Limit Cycles and Attractors**

The phase space curve on which the solutions settle is called a *limit cycle*, 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 at 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).

**9.3.** Now let’s add a driving force to the van der Pol oscillator, like this:

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

(9.2)

Using the parameters of part (a) with $\epsilon = 2$ and $\omega_d = 1.4$, 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 Problem 9.3 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*. This type of chaos has to do with the evolution of dynamical variables (position and velocity) over time. In chaotic systems, the dynamical variables 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 variation in parameters or initial conditions will affect the motion. Usually the best you can do is to find statistics that predict what types of behaviors the system will exhibit on average.

Chaotic systems have been known and studied for a long time. For instance, it comes as no surprise that when you have $6 \times 10^{23}$ 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:

\[
\frac{d^2 \theta}{dt^2} + \gamma \dot{\theta} + \omega_0^2 \sin \theta = A \cos \omega_d t .
\]  

This system has only two degrees of freedom, way less than $6 \times 10^{23}$, and all of the functions that appear in it are nice and smooth. But for certain choices of $A$, $\omega_d$, $\omega_0$, and $\gamma$ the solutions of this differential equation are almost as unpredictable as the motion of an atom in a gas.
Chaos is hard to study because that old standby of physical theory, the formula, is not of much help. If we had a formula for the solution of this differential equation its behavior would be perfectly predictable and un-chaotic. Since the dynamics in chaotic systems are not represented by analytic formulas, their solution had to wait for computers to be invented and to become powerful. The computers that you have at home and that 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 1960’s and 1970’s, so we have all the computing power we need to at least be introduced to this fascinating field.

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

\[ U(x) = -\frac{x^2}{2} + \frac{x^4}{4}. \]  

(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

\[ F_x = -\frac{\partial U}{\partial x} \]  

(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 \texttt{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 \( F_{\text{damp}} = -m\gamma \dot{x} \) 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).

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

The random switching back and forth between equilibrium positions observed in problem 9.4(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
ω = 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 problem 9.3(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?”.

9.5. (a) Make a phase space plot for the system in problem 9.4 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):

```matlab
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 part (c). 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 $\pi$. Such a phase shift could occur by having the oscillator start up in a different way, which might easily happen if your initial conditions were not exactly the same as mine. This flipped-over state is to be expected on physical grounds. My 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.

That is all the time we’ll have to study chaos in this class. If you want to learn more, try Clint Sprott’s website at the University of Wisconsin, Madison.\(^2\) His list of links to other places at the bottom of this website is especially interesting.

\(^2\) http://sprott.physics.wisc.edu/chaosta/
Lab 10

Coupled Nonlinear Oscillators

So far we have only looked at single oscillators, but two or more oscillators can also be hooked together, in which case we say that they are coupled.¹ Consider, for instance, two pendulums hanging from the same piece of horizontally-stretched rubber tubing. If one pendulum is held fixed and the second is displaced from equilibrium, the second one experiences a restoring torque from two separate sources: (a) gravity and (b) the rubber tubing. If both pendulums are displaced together each one experiences gravity and restoring torque from the tubing, but the tubing between the two plays no role because they both twist it in the same direction. But if the pendulums are displaced in opposite directions then the tubing between them is flexed, causing an extra restoring torque. This difference in restoring force between the “together” and “opposite” motions is the cause of the two slightly different frequencies that produce the beating you will see throughout this lab.

Coupled Equations of Motion via Lagrangian Dynamics

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

$$U = \frac{1}{2} \kappa_1 \theta_1^2 + \frac{1}{2} \kappa_2 \theta_2^2. \quad (10.1)$$

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

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

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

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

10.1. Combine this potential energy function with the kinetic energy

$$T = \frac{1}{2} I_1 \dot{\theta}_1^2 + \frac{1}{2} I_2 \dot{\theta}_2^2 \quad (10.4)$$

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

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

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

\[ \omega^2 = \frac{\kappa}{I} \quad ; \quad \omega_c^2 = \frac{\kappa_c}{I} . \]  

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

If you did problem 10.1 correctly, you should have arrived at the following four first-order differential equations for describing the motion of the coupled-pendulum system:

\[ \dot{\theta}_1 = \omega_1 \]  
\[ \dot{\theta}_2 = \omega_2 \]  
\[ \dot{\omega}_1 = -\omega^2 \theta_1 - \omega_c^2 (\theta_1 - \theta_2) \]  
\[ \dot{\omega}_2 = -\omega^2 \theta_2 - \omega_c^2 (\theta_2 - \theta_1) . \]

We write them as a first order set so that Matlab can solve them. The four variables are: the angular positions of the two pendulums \( \theta_1(t) \) and \( \theta_2(t) \) (remember that \( \theta = 0 \) corresponds to the pendulum hanging straight down) and, the angular velocities of the two pendulums \( \omega_1(t) \) and \( \omega_2(t) \). The parameter \( \omega \) is associated with the natural frequency of a single pendulum without any coupling, and the parameter \( \omega_c \) is associated with the natural frequency of the middle section of tubing when attached to the two pendula.

10.2. (a) Use Maple 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 modify the Matlab function \texttt{rhs.m} to solve this system. Use \( \omega = 1.3 \) and \( \omega_c = 0.3 \); for initial conditions let everything be zero except \( \theta_1(0) = 0.3 \). Use Example 16.5a in \textit{Introduction to Matlab} as a template to set these differential equations up to be solved by \texttt{ode45} and make plots of \( \theta_1(t) \) and \( \theta_2(t) \), one above the other using \texttt{subplot} \(^2\) like this:

\begin{verbatim}
 subplot(2,1,1)
 plot(te,th1e)
 subplot(2,1,2)
 plot(te,th2e)
 pause
\end{verbatim}

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 called beats (you studied this in Physics 123) and it is caused by the presence of two different frequencies in the dynamics.

(c) Run the solution out for long enough that when you take the FFT of \( \theta_1(t) \) you can see the two peaks in the power spectrum corresponding to the two frequencies whose mixing causes the beats. Verify that the beat frequency \( \omega_b = \frac{2\pi}{T_b} \).

\(^2\)For more information about subplots, look it up via \texttt{help subplot} using online help in Matlab.
Figure 10.1 Energy passes back and forth between $\theta_1$ and $\theta_2$ due to beating.

(where $T_b$ is the time for one of the oscillators to be at maximum amplitude, go to zero amplitude, then come back to maximum amplitude again) is related to the two peaks in the spectrum $\omega_+$ and $\omega_-$ by

$$\omega_b = \omega_+ - \omega_-. \quad (10.11)$$

Also verify that the two frequencies you observe in the FFT are the two frequencies predicted by your Maple 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

$$\ddot{\theta}_2 = -\gamma \dot{\theta}_2 + ... \quad (10.12)$$

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 $\theta_1$ is positive and $\dot{\theta}_1$ is negative. This 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.

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.
**Note:** 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.

Now we are ready to study a very famous problem in dynamics. In the 1600’s 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.

### 10.3.

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

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

When you do these runs, start pendulum 1 with $\theta_1 = 1$ and $\dot{\theta}_1 = 0$ and try various choices for the initial conditions of pendulum number 2. When they become entrained, check the phase difference between the two clocks. (A visual inspection is probably sufficient). In your numerical experiments, how many different phase relationships do you observe? (Try overlaid plots of $\theta_1$ and $\theta_2$ to see the phase relationships). Do your in-phase and out-of-phase entrained states have the same frequencies?

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

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

\[
\omega > > \frac{2\pi}{T} \tag{11.3}
\]

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

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

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

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 \(X(t)\) changes only slightly during one period of the high frequency motion. Substituting Eq. (11.2) into Eq. (11.1) and expanding in small \(\xi\) through first order gives

\[
\ddot{X} + \ddot{\xi} = -\frac{\partial V}{\partial x} \bigg|_{x=X} - \xi \frac{\partial^2 V}{\partial x^2} \bigg|_{x=X} + A(X) \sin \omega t + \xi \frac{\partial A}{\partial x} \bigg|_{x=X} \sin \omega t. \tag{11.5}
\]

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

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

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

\[
\xi(t) = -\frac{A(X)}{\omega^2} \sin \omega t \quad ; \quad \dot{\xi}(t) = -\frac{A(X)}{\omega} \cos \omega t .
\]  

(11.7)

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

\[
\ddot{X} = -V'(X) - \frac{A(X)}{2\omega^2} \frac{dA}{dX} .
\]  

(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 \( X \):

\[
X(0) = x_0 \quad ; \quad \dot{X}(0) = v_0 + A(x_0)/\omega .
\]  

(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.

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

\[
y_{\text{support}} = b \sin \omega t .
\]  

(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 \( a \), then the pendulum will experience a downward gravitational force \( -ma \). Hence we may write for the effective acceleration of gravity acting on the pendulum

\[
a = \ddot{y}_{\text{support}} = -\omega^2 b \sin \omega t \text{ so that the total acceleration, including ordinary gravity is}
\]

\[
g_{\text{eff}} = g - a = g - \ddot{y}_{\text{support}} = g + b\omega^2 \sin \omega t ,
\]  

(11.11)

which then leads to the equation of motion

\[
\ddot{\theta} = -\omega_0^2 \sin \theta - \frac{b\omega^2}{L} \sin \theta \sin \omega t ,
\]  

(11.12)
where $\omega_0^2 = g/L$. This equation of motion matches Eq. (11.2) if we write

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

(11.13)

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

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

(11.14)

OK, we are finally ready to calculate.

11.1. (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.

![Figure 11.1](image)

(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.

(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

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

(11.15)
(integrate both sides of this equation to obtain $V(\theta)$). Then plot this potential from $\Theta = 0$ to $\Theta = 2\pi$ for various values of $b$ in the range $b = 0$ to $b = .1$ and notice what happens at $\Theta = \pi$ as $b$ increases. Then use calculus to find the critical value of $b$ at which the straight-up pendulum first becomes stable and use your code from part (b) to verify that this threshold value is correct.

These low frequency effective forces that arise from high-frequency non-linear effects are called *ponderomotive forces* and they show up all the time in physical problems. This problem is just a small taste of a very large field.
Lab 12

Two Gravitating Bodies

Consider two masses interacting through Newton’s law of gravity: 1

\[
\begin{align*}
    m_1 \ddot{r}_1 &= -\frac{G m_1 m_2}{|\mathbf{r}_1 - \mathbf{r}_2|^3} (\mathbf{r}_1 - \mathbf{r}_2) \\
    m_2 \ddot{r}_2 &= -\frac{G m_1 m_2}{|\mathbf{r}_1 - \mathbf{r}_2|^3} (\mathbf{r}_2 - \mathbf{r}_1)
\end{align*}
\]

(12.1) 

(12.2)

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

12.1. (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

\[
\begin{align*}
    x_1(0) &= 1, & x_2(0) &= -1 \\
    y_1(0) &= 0.5, & y_2(0) &= -0.3 \\
    z_1(0) &= -0.3, & z_2(0) &= 0.6 \\
    v_{x1}(0) &= 0.65, & v_{x2}(0) &= -0.45 \\
    v_{y1}(0) &= 0.2, & v_{y2}(0) &= 0.3 \\
    v_{z1}(0) &= 0.1, & v_{z2}(0) &= -0.3.
\end{align*}
\]

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

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

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

As your script runs you should see your masses doing an intricate gravitational dance, and the final picture should look just like the one in Fig. 12.1 (after the appropriate rotation of your figure).

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 $r_1$ 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:

$$ r_1 = [x_1, y_1, z_1]; $$
$$ r_2 = [x_2, y_2, z_2]; $$
$$ v_1 = [v_{x1}, v_{y1}, v_{z1}]; $$
$$ v_2 = [v_{x2}, v_{y2}, v_{z2}]; $$

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

$$ r_{1e} = [x_{1e}, y_{1e}, z_{1e}]; $$
$$ r_{2e} = [x_{2e}, y_{2e}, z_{2e}]; $$
$$ v_{1e} = [v_{x1e}, v_{y1e}, v_{z1e}]; $$
$$ v_{2e} = [v_{x2e}, v_{y2e}, v_{z2e}]; $$

With them in this form the rest of this problems 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 $r_1$ and $r_2$ for the center of mass position and the relative position of $m_1$ with respect to $m_2$:

$$ R = m_1 r_1 + m_2 r_2 \quad ; \quad r = r_1 - r_2 \quad (12.3) $$

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

\[
\mathbf{L} = m_1(\mathbf{r}_1 - \mathbf{R}) \times (\mathbf{v}_1 - \mathbf{V}) + m_2(\mathbf{r}_2 - \mathbf{R}) \times (\mathbf{v}_2 - \mathbf{V})
\]  

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 \( \mathbf{L} \) you can easily evaluate the angular momentum as a matrix (rows are time, columns are \( x, y, z \) components):

\[
\mathbf{L}=m1*cross(r1-R,v1-V)+m2*cross(r2-R,v2-V);
\]

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

\[
\text{plot}(t,L(:,1),t,L(:,2),t,L(:,3))
\]

(d) Show graphically that \( \mathbf{L} \) 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 \texttt{dot} command you will need to make a slight change to the syntax we used above with the \texttt{cross} command. The \texttt{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 \( \mathbf{r} \) 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).
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 $r = 0$ is at the position of $m_2$).

**Kepler’s Second Law**  An imaginary line drawn from $m_2$ to $m_1$ (the $r$ 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

$$T = \frac{\pi}{\sqrt{2}} \frac{\mu^{1/2}Gm_1m_2}{(-E)^{3/2}}$$

(12.5)

where $\mu$ is the reduced mass

$$\mu = \frac{m_1m_2}{m_1 + m_2}$$

(12.6)

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

$$E = \frac{\mu}{2} |\dot{r}|^2 - \frac{Gm_1m_2}{|r|}$$

(12.7)

(remember that $r = r_1 - r_2$ and that $\dot{r} = v = v_1 - v_2$).

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.

12.3.  (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. 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.

![Non-inverse Square Law](image)

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

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

(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 (i). 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. 16.6. 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. The laboratory covers two lab periods and consists of lots of reading and things to do. You will need to use both the Maple 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

\[ \ddot{x} = -\gamma \dot{x} + F(x) + A \cos \omega t, \]  

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

\[ F(x) = \frac{2 \tanh (3x/2)}{3 \cosh^2 (3x/2)}, \]  

and where both linear damping \( -\gamma \dot{x} \) 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 ugliness of \( F(x) \), let’s plot it so you can see that it’s nice and regular.

13.1. Plot both the force \( F(x) \) and the potential energy function \( U(x) \) associated with this oscillator:

\[ F(x) = -\partial U/\partial x \]  

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.

13.2. To clearly understand this point, expand \( F(x) \) in a Taylor series in small \( x \) and find that the natural frequency of a particle with small displacement \( x \) is \( \omega_0 = 1 \).

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 particles 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 $\omega$. 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 the figure below.

(i) Using a relatively small driving force and weak damping let’s imagine starting at small driving frequency $\omega$ 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 $\omega$ 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 $\omega$ 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 $\omega$, 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.

Maple Analysis

As a first attempt to understand the behavior of Fig. 13.1 it will be helpful to do some analysis with Maple. This analysis will only be approximate, but will help us see how the resonance curve might behave. The first approximation is to assume that it makes sense to only keep the first two non-zero terms in the Taylor expansion of $F(x)$.

13.3. Use Maple to show that the force given in Eq. (13.2) is approximately

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

for small values of $x$.

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

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

13.4. (a) Use Maple to substitute this form for $x(t)$ into Eq. (13.4), use `combine` to turn powers of cosine into terms containing $\cos n\omega t$, $n = 1, 3, 5, \ldots$, and then collect all of 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). Maple has a nice command that will do term collection for you. To use it, make sure that your differential equation containing all of the $\cos n\omega t$ terms is in the form $\text{eq}:=\ldots=0$. Then use Maple’s `collect` command like this:

```maple
collect(eq,[cos(w*t),cos(3*w*t)]);
```

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 $\omega$. 
(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 \( \omega \). 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

\[
c_3 = \frac{3c_1^3}{4} \cdot \frac{1}{1 - 9\omega^2}
\]  

(13.6)

Under what conditions is \( c_3 \) not small as we assumed? In particular, what value of \( \omega \) totally invalidates this assumption? This special value of \( \omega \) 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 \( \omega \) 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

\[
c_1 - \omega^2 c_1 - \frac{9}{4} c_3^3 = A
\]  

(13.7)

(d) Have Maple solve for the three roots of this equation with \( A = 0.136 \). Maple can solve a cubic like this, but the answers look horrible, so just assign each one to a separate variable like this (use colons at the end instead of semicolons so you don’t have to look at pages and pages of blue output):

\[
\text{sol:=solve(eq,c1):}
\]

\[
s1:=\text{sol}[1]:
\]

\[
s2:=\text{sol}[2]:
\]

\[
s3:=\text{sol}[3]:
\]

Each of \( s1, s2, \) and \( s3 \) is a formula for one of the three solutions for \( c_1 \).

(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 \( \omega \) 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 \( \omega \) 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 \( \omega \). Above 0.6 there is a single amplitude \( c_1 \), and as the frequency decreases 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 \( \omega \) 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 \( \omega \) 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 Maple 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 \( \gamma \)) 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.

![Figure 13.2](image)

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 \( \omega \), but also on what \( \omega \) used to be: the system remembers what path it has been following as \( \omega \) has been changing.
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 $\omega$ 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 \texttt{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:

$$
\text{lab13.m} \rightarrow \text{fminsearch} \rightarrow \text{leastsq13.m} \rightarrow \text{ode45} \rightarrow \text{rhs13.m}
$$

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

13.5. Write a main script called \texttt{lab13.m} that starts $\omega$ at $\omega_1$ and ends it at $\omega_2$ taking $N$ steps along the way. This is how we will track the response of the oscillator as we scan $\omega$ through resonance going either up or down in frequency. This script, which we call \texttt{lab13.m}, should do the following things:

(a) Declare the driving frequency $\omega$, the driving amplitude $A$, and the damping constant $\gamma$ as global variables. After they are declared give them values $A = 0.136$ and $\gamma = 0.4$. Also declare the variable $S$ to be global (you will see why later).

(b) Use \texttt{input} commands to enter values $\omega_1, \omega_2$ to define the driving frequency scan interval, and also use \texttt{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 $\omega$-values called \texttt{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 $\omega$ 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.

Loop Example

```matlab
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
    % 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.

13.6. (a) Write `leastsq13.m` following the pattern below (we have left some blanks for you to fill in, indicated by ...).

```matlab
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
```

(b) Finally, write the M-file `rhs13` used by the main script and by `leastsq13`. This is just the usual right-hand side function used by `ode45` and should look like this:

```matlab
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
```

Note: 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 $\gamma = .4$. 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 $\gamma = 0.4$ and $\omega = 2$ the correct initial values are $(x_0, v_0) = (-0.0423, 0.0225)$.

Now we are going to use this set of codes to explore the true hysteresis diagram which we approximated in the Maple analysis.

**Important:** Use $A = 0.136$ throughout the rest of this lab.

13.7. (a) Set $\gamma = 0.03$ 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 $\omega = 1/3$ from Sec. 13.2, 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 in Sec. 13.2. 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 $\omega_1 = .2$ up to high frequency at $\omega_2 = 2$ with the following value for the damping constant: $\gamma = 0.4$. 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.

13.8. Set the damping constant to $\gamma = 0.3$ and run your loop for 5 different sets of values of $\omega_1$ and $\omega_2$, 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 $\omega$, $(x_0, v_0)$, 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:)

```matlab
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 \texttt{fminsearch} in \texttt{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 \texttt{break} command.

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

\begin{verbatim}
    wscan=wscan(1:length(Amp));
\end{verbatim}

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 \texttt{fminsearch} a sufficiently accurate initial guess. You can fix this by using linear extrapolation, like this:

Before the call to \texttt{fminsearch} set an old answer variable equal to the last \texttt{answer} (which contains refined values of $[x_0;v_0]$) so that it will be remembered before a new answer is found:

\begin{verbatim}
    oldanswer=answer;
\end{verbatim}

Then at the bottom of the loop replace the code

\begin{verbatim}
    guess=answer
\end{verbatim}

with linear extrapolation:

\begin{verbatim}
    guess=2*answer-oldanswer
\end{verbatim}

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

\begin{verbatim}
    guess=[x0;v0];
\end{verbatim}

to

\begin{verbatim}
    guess=[x0;v0];
    answer=guess;
\end{verbatim}

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 $\texttt{wscan}$ and $\texttt{Amp}$ 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 Maple plots you made in Sec. 13.2? Does the jump occur at a higher value of $\omega$ than the last good value on the upper curve in (ii)? If it does, then you have achieved hysteresis. Congratulations. Remember to save $\texttt{wscan}$ and $\texttt{Amp}$ 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:

```matlab
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) Finally (hang on, we’re almost finished) 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 $\omega_1 = 0.3$, $\omega_2$ set to anything else, and $N = 2$. We don’t care about $\omega_2$ 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 for the initial guess $(x_0, v_0) = (0.35, 0.31)$ 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

Amplitude, 5
Attractor, 32

Baseball with air, 4
Bessel function zeros, 12
Butterfly effect, 35

Chaos, 31
Coupled oscillators, 37

DEplot, 9
Derivatives, numerical, 11
Differential equations with Matlab, 31
Dynamical chaos, 31, 34

Entrainment, 33, 36, 40
Euler’s constant, γ, 7

FFT, 13
Fibonacci numbers, 11
Fourier transform, 13
Fractal curve, 12

Gravitating bodies, 45

High-frequency driving forces, 43
Hysteresis, 51

Integrals, numerical, 11
Intermittency, 35

Kepler’s laws, 48

Limit cycle, 32

Nonlinear damped driven oscillator, 51

Parametric oscillator, 23
Pendulum, 27
Pendulum period, 27
Pendulum, driven, 28
Pendulum, parametric instability, 29

Period-doubling route to chaos, 35
Phase shift, 5
Phase space, 7
Ponderomotive forces, 44
Pumping a swing, 23

Quiver, 9

Roundoff, 11

Simple harmonic oscillator, 5
Strange attractor, 32

Zeros, Bessel functions, 12