Introducing Computational Approaches in Intermediate Mechanics

Text—somewhat expanded—of Invited Talk
Delivered at the Winter Meeting of
the American Association of Physics Teachers
Seattle, Washington   7 January 2007

David M. Cook

Department of Physics
Lawrence University
Box 599
Appleton, WI 54912

920-832-6721   david.m.cook@lawrence.edu

ABSTRACT

In the winter of 2003, we at Lawrence University moved Lagrangian mechanics and rigid body dynamics from a required sophomore course to an elective junior/senior course, freeing 40% of the time for computational approaches to ordinary differential equations (trajectory problems, the large amplitude pendulum, non-linear dynamics); evaluation of integrals (finding centers of mass and moment of inertia tensors; calculating gravitational potentials for various sources); and finding eigenvalues and eigenvectors of matrices (diagonalizing the moment of inertia tensor, finding principal axes), and to generating graphical displays of computed results. Further, students begin to use LaTeX to prepare some of their submitted problem solutions. Placed in the middle of the sophomore year, this course provides the background that permits faculty members as appropriate to assign computer-based exercises in subsequent courses. Further, students are encouraged to use our Computational Physics Laboratory *on their own initiative* whenever that use seems appropriate.

1 Introduction

Practicing physicists routinely confront a variety of tasks that support their research but are peripheral to their main objectives. While physicists in different subareas will probably disagree about the relative importance of particular tasks, most will agree that the more important and most frequent of these tasks involve visualizing functions of one, two, and three variables, solving algebraic and differential equations, evaluating integrals, finding roots, eigenvalues, and eigenvectors, acquiring, displaying, and analyzing experimental data, processing images, and preparing papers and talks. More often than not, pursuit of these tasks symbolically and analytically is at best tedious, difficult, and prone to error and at worst essentially impossible. In many cases, however, these tasks can be addressed by exploiting computational approaches. To facilitate their use of such approaches, practicing physicists of the twenty-first century must be acquainted with an operating system (preferably some flavor of UNIX or LINUX), a text editor, a spreadsheet, a computer algebra system, an array/number processor, a visualization tool, a standard computational language, a program for data acquisition, publishing software, presentation software, and maybe other packages for more specialized tasks like circuit simulation and image processing.

Undergraduate curricula that do not provide physics majors with at least an orientation to the applicable algorithms and a reasonable spectrum of the appropriate computational tools are failing to prepare their graduates for the activities that practicing scientists will find themselves doing frequently and extensively in the twenty-first century. The task before the physics educator is to design a physics curriculum that, without short-changing important topics from the historical curriculum, nonetheless includes exposure to and opportunity to build skill in the use of computational approaches to a selected set of representative problems in physics. While illustrations must, of course, be drawn from several areas of physics, the objective is as much to develop generalizable computational skills as to learn something of the physics of the situations used as examples.

These are the convictions that have driven and informed the curricular evolution at Lawrence University during the past couple of decades. In the broadest of terms, by graduation (and, with any luck, well before that time), we want each of our majors to have developed not only the ability to recognize when a computational approach may have merit but also the skill to pursue that approach confidently, fluently, effectively, knowledgeably, and—most of all—independently. In this talk, I want to focus on the sophomore mechanics course which we have modified to include computational elements and which provides the starting point in the development of our majors' exposure to serious computation in application to problems in physics.

2 Curricular Context

First, however, I want to describe briefly the curricular structure in which our efforts are embedded. Each year, full-time students at Lawrence take three courses in each of three ten-week terms. A one-term course is treated officially as the equivalent of a 3-1/3 semester-hour course, class periods are 70 minutes long, and introductory laboratories are three hours long. The ten physics and four mathematics courses required of a typical *minimum* major are shown in Table 1. The senior year is not as thin as it looks, because most majors elect more than the minimum number of physics courses and many will elect one term or more of independent study starting in the apparently empty fall term.

In broad outline, freshman prospective majors encounter *LoggerPro*, *Kaleidagraph*, and *Excel* for data acquisition and analysis, curve fitting, and graphical visualization. Fall-term sophomores in *Electronics* encounter *Multisim 7* for circuit simulation and continue to use *Kaleidagraph* and *Excel*; winter-term sophomores in *Computational Mechanics* experience a concentrated exposure to IDL, MAPLE, and LaTeX for solving ordinary differential equations, evaluating integrals, visualizing functions graphically, and preparing neatly printed problem solutions; and spring-term sophomores in *Electromagnetic Theory* continue to use IDL and MAPLE for graphical visualization and for numerical solution of Laplace's equation. Junior/senior courses in quantum mechanics, advanced laboratory, computational physics, mathematical methods of physics, plasma physics, and advanced mechanics make further explicit use of computers, though the extent of those uses varies with the instructor. In addition, at all points in our curriculum, students are free to use the resources of the Computational Physics Laboratory (CPL) whenever they deem such use to be appropriate, and many do so in *Advanced Laboratory*, senior independent projects, and many other contexts. Upper-class majors have 24/7 access to the CPL.

	Term I	**Term II**	**Term III**
Fresh	Freshman Studies Elective **Calculus I**	Freshman Studies ***Intro Classical Physics** **Calculus II**	Elective ***Intro Modern Physics** **Calculus III**
Soph	***Electronics** **Diff Eq/Lin Alg** Elective	***Computational Mechanics** Elective Elective	***E and M** Elective Elective
Junior	***Quantum** Elective Elective	**Physics Elective** Elective Elective	***Advanced Lab** **Physics Elective** Elective
Senior	Elective Elective Elective	**Physics Elective** Elective Elective	Elective Elective Elective
Available Physics Electives: Group 1: Thermal Physics, Optics, Advanced Mechanics, Advanced E and M, Mathematical Methods of Physics, Advanced Modern Physics, Plasma Physics, Solid State Physics, Laser Physics, Computational Physics, Tutorials. Group 2: Independent Studies/Capstone.			

Table 1: Typical Program of a Physics Major. Courses in bold type are required for a minimum major in physics; courses marked with an asterisk direct students explicitly to the computer and, in most cases, include some instruction in one or more of our computational resources.

3 Computational Mechanics

Prior to the academic year 2002–03, we offered a traditional sophomore course in *intermediate* mechanics and an *elective* sophomore course called *Computational Tools in Physics.* In 2002–03, the course *Computational Tools in Physics* was discontinued, and the *required* sophomore course titled *Computational Mechanics*, which replaced the course in classical mechanics, became the starting point in our nurturing of our students' abilities to take full advantage of the resources of the CPL. *Computational Mechanics* combines about 60% of the material covered in a traditional course in intermediate mechanics with about 40% of the orientation to computing covered in the discontinued course. The incorporation of the computational elements in a course that is *required* of sophomore majors assures that *all* majors have an early introduction to computational approaches, makes possible the assumption of that background in all subsequent courses, and supports independent use of our computational resources even if individual faculty members in later courses do not make

Week 01	Orientation to LINUX (including text editor) Kinematics/Dynamics of Translation/Rotation Impulse/Momentum/Work/Kinetic Energy Gravity/Electromagnetic Force/Friction/Tension
Week 02	Orientation to IDL/TGIF (basic capabilities, visualization)
Week 03	Equations of Motion (constant force/torque, force dependent only on t, ... only on x, ... only on v) Potential Energy, SHM, Equilibrium Work and Potential Energy in 3D
Week 04	Velocity-Dependent Forces Damped and Driven SHM; Resonance Coupled and Small Amplitude Oscillation
Week 05	HOUR EXAMINATION Orientation to LaTeX Central Forces/Effective Potential Energy, Orbital Equation
Week 06	Planets, Satellites, Comets, Scattering MID-TERM READING PERIOD
Week 07	Orientation to MAPLE Symbolic Solution of ODEs with MAPLE Algorithms to solve ODEs numerically
Week 08	Numerical Solution of ODEs with IDL
Week 09	HOUR EXAMINATION Symbolic Evaluation of Integrals with MAPLE
Week 10	Algorithms to Evaluate Integrals Numerically Numerical Evaluatation of Integrals with IDL
Week 11	FINAL EXAMINATION

Table 2: Weekly Schedule for *Computational Mechanics*.

explicit assignments involving those resources.

3.1 Outline of the Course

The main objectives of *Computational Mechanics* are conveyed by its catalog description, which asserts that it “introduces symbolic and numerical computation through examples drawn mainly from classical mechanics but also from classical electromagnetism and quantum mechanics” and that it “emphasizes computer-based approaches to graphical visualization, the solution of ordinary differential equations, the evaluation of integrals, and the finding of roots, eigenvalues and eigenvectors”. The main topics of the course are summarized in Table 2

The course begins with a tutorial exercise to acquaint students with the features of the workstations in the CPL and with the LINUX operating system. Concurrently, students review and extend their introductory studies of translational and rotational kinematics and dynamics, impulse, momentum, work, kinetic energy, and common forces. They spend the second week entirely in the CPL becoming acquainted with the general capabilities of IDL, especially for graphical visualization of scalar functions of one, two, and three variables, and with Tgif for generating drawings. In the third and fourth weeks, the course returns to the fundamental laws of mechanics to set up and solve the usual problems in one-dimensional motion via standard analytic techniques, and to extend the definition of potential energy and conservative forces to three dimensions. Then, in the fifth and sixth weeks, students encounter the elements of LaTeX and spend several days on the standard analytic approaches to the central force problem. The remainder of our ten-week term is spent mostly in computationally related activities, including an orientation to MAPLE, a discussion of numerical algorithms for solving ordinary differential equations (ODEs) and evaluating integrals, and an introduction to routines built into IDL for solving ODEs and evaluating integrals.

In the rearrangement that generated *Computational Mechanics* from its predecessor, a more traditional intermediate course in classical mechanics, only Lagrangian mechanics and rigid-body dynamics were relocated to another—and later—course to provide the time for the addition of computational approaches to graphical visualization, solving ODEs, evaluating integrals, and preparing documents. Other computational topics are incorporated in the introductory laboratories and in later courses. *Computational Mechanics*, which is a rapidly paced and intensive course, provides the starting point in which students are awakened to the merit of computational approaches and learn to use some of those approaches. Students in the course spend quite a bit of time in the CPL. Especially during the first few weeks of the term, when students are inexperienced with the available computational tools, we arrange for a more experienced student—often a student who worked with me in the previous summer—to be available in the CPL for four to six scheduled hours each week to help students in the course strengthen their wings so that, later in the term, they can be expected to fly more confidently on their own.

4 Examples

In the remainder of this talk, I want to describe several representative exercises that are assigned to students in *Computational Mechanics*. I will limit myself to computational exercises and—for the sake of treating more examples in a short time—to only sketching the coding, but please understand on the one hand that something like 50% of the assigned exercises and 60% of the course content deal with mechanics in traditional ways and on the other hand that students are expected—to be sure with help—to generate the entirety of the coding I will only sketch. While emerging from considerations in mechanics, the computational component of the course is organized not by topics in mechanics but by computational strategies such as graphical visualization, solving ordinary differential equations, and evaluating integrals. In structuring these components of the course, I therefore feel free to draw on examples from areas of physics outside of mechanics to illustrate the computational strategies.

4.1 Visualization: Black Body Radiation

After becoming familiar with the operating system, students first learn about strategies for generating graphical displays of functions of one and two independent variables. Gravitational and electrostatic potential energies, particle trajectories, diffraction patterns, quantum probability densities, and many other physical situations provide the functions. Among my favorites is the blackbody radiation spectrum,

$$u(\lambda, T) = \frac{8\pi ch}{\lambda^5}\frac{1}{e^{ch/(\lambda kT)} - 1}$$

I choose this example in part because it illustrates very effectively the importance of thinking about scaling when preparing a graphical display. Students seem to have difficulty with the broad notion of scaling and particularly with conceiving systematic strategies for casting expressions in a dimensionless form, but I think the skill to be sufficiently important that I try to assign many exercises in which students are forced to practice the skill. To cast this specific expression in dimensionless form, students pick an arbitrary wavelength λ_0 as a unit, recast the expression as a function of the dimensionless wavelength $\overline{\lambda}$ by setting $\lambda = \lambda_0\,\overline{\lambda}$, recognize along the way that temperature would conveniently be measured in units of $T_0 = ch/(\lambda_0 k)$ and spectral density in units of $u_0 = 8\pi ch/\lambda_0^5$ and conclude that they should plot the function $\overline{u} = u/u_0$ given by

$$\overline{u} = \frac{1/\overline{\lambda}^5}{e^{1/(\overline{\lambda}\,\overline{T})} - 1}$$

as a function of $\overline{\lambda}$ for various values of $\overline{T} = T/T_0$. Arguing—reasonably—that, in a dimensionless casting, the range $0.0 \leq \overline{\lambda} \leq 10.0$ and the value $\overline{T} = 1.0$ would be a good first trial, students using IDL might construct the IDL coding

```
lambda = findgen(201)/20.0
T = 1.0
u = (1.0/lambda^5)/( exp(1.0/(lambda*T)) - 1 )
u[0] = 0.0
plot, lambda, u, title='...', thick=..., ...
```

which, respectively, set a vector to contain the values [0.0,0.05, 0.10, ... 10.00], choose a reasonable temperature, evaluate the spectrum, correct for the indeterminate value at lambda = 0, and plot the desired graph of u versus λ. (The ellipses in the plot statement replaces several embellishments that students quickly learn to add to the basic `plot` command.)

A common error in this exercise is for students to omit the parentheses around λT in the exponent, thus evaluating not $1/(\lambda T)$ but, because of the priority of operations, T/λ, and receiving quite unreasonable graphs. If that error has been avoided (or corrected), then the resulting graph looks reasonable—save that the peak is very much crowded up against the vertical axis. I expect students at this point to realize that the interval $0.0 \leq \lambda \leq 10.0$ is simply too wide an interval for satisfactory display of the function. Examination of the first trial graph suggests that the interval $0.0 \leq \lambda \leq 1.0$ would be more sensible. That choice—not shown in the above coding—leads to a much more reasonable graph of this function.

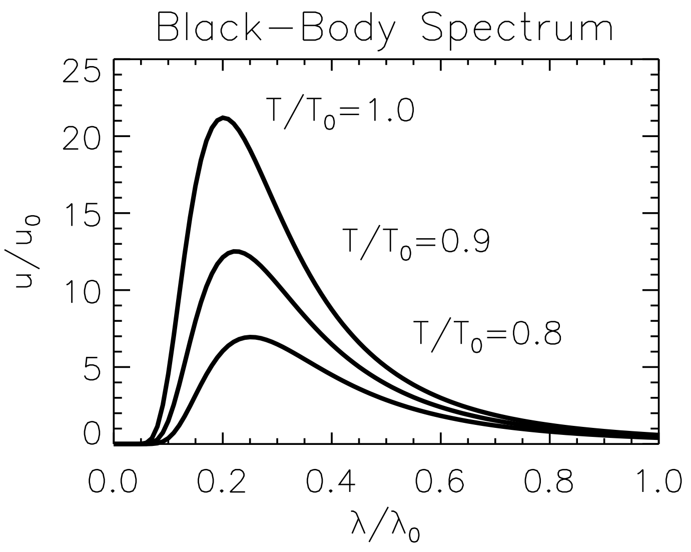

Figure 1: Black-Body Spectrum

Actually, I put in a statement—the penultimate statement `u[0]=0.0`—that students probably wouldn't put in on first pass. I would expect students to notice, however, that without that statement the graph doesn't start at $\lambda = 0$. I would further expect them to inquire why that point is omitted and, perhaps guided by a warning message, to discover first that the function is indeterminate at that point and second that IDL is able to deal with that indeterminacy in a way that, while not evaluating a limit, still allows the rest of the graph to be produced. The function, however, has a perfectly sensible limit—zero—at that point. To put that point on the graph, the statement setting $u[0] = 0$ is added—but typically only in hindsight (and only if the student happens to notice the gap in the graph).

The code above will produce a satisfactory graph on the screen. It takes a bit more tinkering to generate a graph that prints satisfactorily and, perhaps to decide on other temperatures and overplot a few more graphs. The student's final graphical result, however, might look like Fig. 1

Another part of this exercise asks students to generate a surface plot of the function u over the λ-T plane. The essential code to achieve that end is

```
lugen_grid, lambda, xrange=[0.0,1.0], nx=50, $
   T, yrange=[0.6,1.0], ny=20
u = (1/lambda^5)/( exp(1.0/(lambda*T)) - 1 )
u[0,*] = 0.0
device,decomposed=0 & loadct, 3
shade_surf, u, lambda, T, title='...', ...
```

which creates two arrays with values of λ and T, evaluates the function as a function of two variables, corrects for the indeterminacy at $\lambda = 0$, sets the color table for a surface plot, and creates that plot. The result after a bit of embellishment to put on labels is shown in Fig. 2. I do not know why the plot has the shaded background.

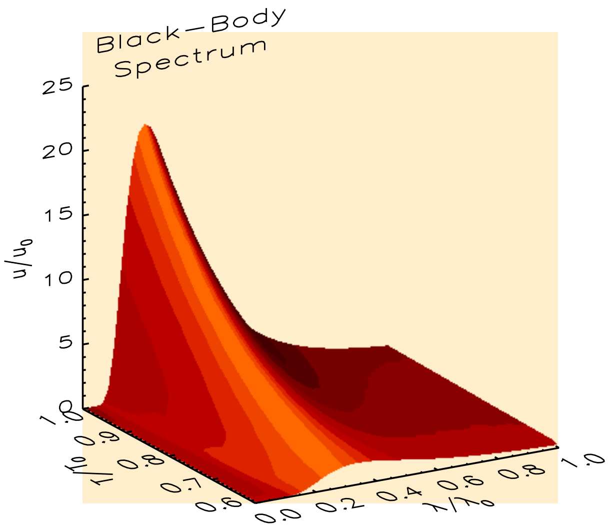

Figure 2: Surface Plot of Black-Body Spectrum

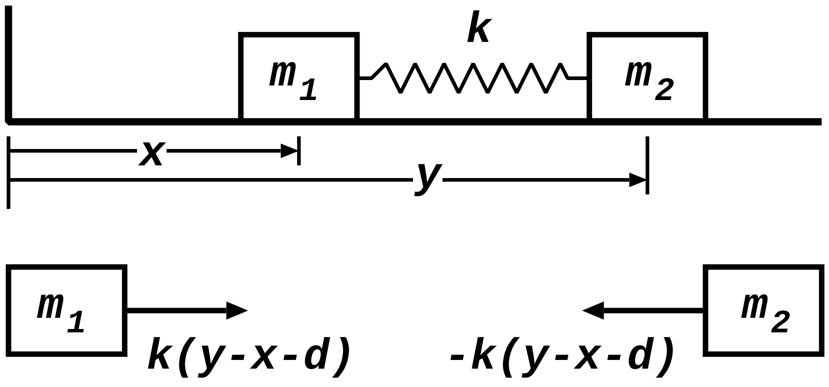

Figure 3: A System of Coupled Oscillators

4.2 Solution of ODEs Symbolically

The most ubiquitous mathematical task in classical mechanics involves the solution of ordinary differential equations, with common examples including driven, damped oscillations; trajectories in various gravitational, electrostatic, and magnetostatic force fields; and behavior of coupled oscillators. Students in *Computational Mechanics* address this task both symbolically and numerically. To illustrate the symbolic approach, I choose here to explore the motion of two blocks of different masses m_1 and m_2 free to slide along a one-dimensional, frictionless track but connected by a Hooke's-law spring of constant k and natural length d. The system is shown in Fig 3. The ultimate objective is to demonstrate the separation of the motion into the motion of the center of mass on which is superimposed the motion of one block relative to the other. Students quickly draw the force diagrams and determine the equations of motion to be

$$m_1 \frac{d^2x}{dt^2} = k(y - x - d) \quad ; \quad m_2 \frac{d^2y}{dt^2} = -k(y - x - d)$$

Then, they invoke the MAPLE statements

```
eq1 := diff( x(t),t$2 ) = k*( y(t) - x(t ) - d );
eq2 := diff( y(t),t$2 ) = -k*( y(t) - x(t ) - d );
eqs := { eq1, eq2 };
ics := { x(0)=x0, D(x)(0)=v0, y(0)=y0, D(y)(0)=w0 };
soln := dsolve( eqs union ics, { x(t), y(t) } );
```

to define the equations as a set, define the initial conditions as a set, and obtain a solution. Unfortunately, the solution is a major mess—and it takes some doing to beat the expressions into a reasonable and interpretable form. One interesting result can, however, be found with the statements

```
xx := eval( x(t), soln ); yy := eval( y(t), soln );
xcm := ( m1*xx + m2*yy )/( m1 + m2 );
```

which extract $x(t)$ and $y(t)$ as simple expressions and calculate the position of the center of mass, finding it—I suppress a few (not entirely simple) details—to be given by

$$x_{\text{cm}} = \frac{m_1 v_0 + m_2 w_0}{m_1 + m_2} t + \frac{m_1 x_0 + m_2 y_0}{m_1 + m_2}$$

Although it sometimes requires a hint or two, most students will recognize this result as describing motion at constant velocity (the coefficient of t) away from the initial position of the center of mass with the initial velocity of the center of mass!

Students are also asked to calculate the separation of the two blocks, the first step of which is simple enough and involves the MAPLE statements

```
r := yy - xx;
```

Beating the result into an interpretable form is complicated. Students should note, however, that the result contains several terms involving the cosine function, several involving the sine function, and one involving no time dependence. Further, students should notice that the sine and cosine functions all have the same argument, which I have abbreviated in the compacted output

$$r = d + (y_0 - x_0 - d) \cos \Omega t + \frac{w_0 - v_0}{\Omega} \sin \Omega t$$

as Ωt. Given class discussions, students should also recognize Ω as $\sqrt{k/\mu}$, where μ is the reduced mass of the two-body system.

In essence, students working this exercise find first hand that formal solution of the equations of motion for a representative system reveals, first, that the center of mass moves with constant velocity in the absence of external forces and the linear momentum of the system is constant and, second, that the motion of each block relative to the other is simple harmonic oscillation about the equilibrium separation!

With the equations of motion still in MAPLE's workspace, students can also ask about the normal modes of oscillation. The MAPLE statements

```
eq := eval( {eq1, eq2}, y(t)=z(t)+d );        (1)
tmp := cos(omega*t);                          (2)
guess := { x(t) = x0*tmp, z(t) = z0*tmp };    (3)
eqn := eval( eq, guess );                     (4)
eq3 := simplify( eqn[1]/cos(omega*t) );       (5)
eq4 := simplify( eqn[2]/cos(omega*t) );       (6)
tmp := solve(eq3, z0 );                       (7)
char := eval( eq4, z0=tmp );                  (8)
solve( char, omega^2 );                       (9)
```

yield the output

$$\omega^2 = 0, \frac{k(m_1+m_2)}{m_1 m_2} \quad \text{or} \quad \omega^2 = 0, \frac{k}{\mu}$$

for the frequencies of the two modes. In these statements, we recognize—line 1—that the natural length of the spring can be absorbed into a translated y, so the differential equations become homogeneous. Then we substitute—lines 2, 3, and 4—a *guessed* sinusoidal solution with an as yet unknown frequency, extract—lines 5 and 6—the two algebraic equations for the unknown amplitudes by dividing out the common sinusoidal factor, and systematically solve—lines 7, 8, and 9—the resulting pair of equations for ω^2. We find two solutions, one of which—the zero—conveys the steady motion at constant velocity (an oscillation at frequency zero, i.e., period infinity) and the other of which conveys the oscillation already seen in the full solution. The connection between center of mass motion and relative motion on the one hand and the two normal modes of oscillation on the other hand emerges naturally in this analysis.

Beyond revealing the nature of center of mass motion superimposed on top of relative motion as a way to think of the motion of a two-body system acted on by a central force, there are several lessons in this exercise. The most important of these is that programs like MAPLE really demand that you know where you are going and what to expect. Beating the results of the first step in a calculation into a reasonable and interpretable form often requires a fair bit of ingenuity and, above all, a lot of appreciation of where you think you should end up. A simple statement that says "simplify this expression and put it in the form I would like to see" does not usually exist. Indeed, I will admit that complexity associated with beating expressions into sensible forms has somewhat damped my original I would now say overly optimistic expectation about the value of computer algebra systems.

4.3 Solving ODEs Numerically

Discussion of numerical solutions to ODEs begins with a discussion of the simplest algorithms, of explicit coding of some of those algorithms, of the ways in which one can assess the accuracy of solutions generated numerically, and—qualitatively—of ways in which one might improve the algorithms to speed the calculation or increase its accuracy, given the hazards of doing arithmetic to finite precision on floating point numbers. Ultimately, however, students move to using fairly sophisticated, adaptive algorithms by exploiting ODE routines built in to IDL. Examples include many of the problems already discussed analytically but also non-linear and chaotic systems. One of my favorite examples is the large amplitude pendulum shown in Fig 4. When times are measured in units of $\sqrt{l/g}$, the

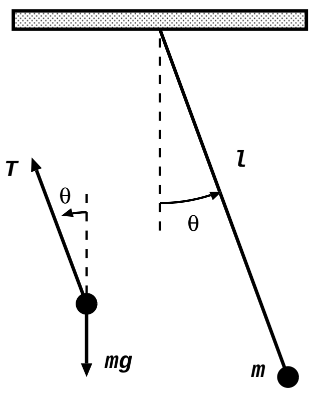

Figure 4: A Simple Pendulum.

equation of motion for this system is

$$\frac{d^2\theta}{dt^2} = -\sin\theta$$

In preparation for invoking an existing IDL routine, students must see this second-order equation as a pair of first order equations,

$$\frac{d\theta}{dt} = \omega \quad ; \quad \frac{d\omega}{dt} = -\sin\theta$$

associate the two dependent variables with the elements of a single vector in the program,

$$\theta \mapsto \mathsf{y[0]} \quad ; \quad \omega \mapsto \mathsf{y[1]}$$

and create the function subprogram

```
FUNCTION pendulum, t, y
RETURN, [ y[1], -sin(y[0] ]
END
```

that takes as input the (scalar) independent variable and (vector) dependent variable and returns a vector containing the value of the derivatives of the two dependent variables. From this point, one invokes the statements

```
amp = 10.0
amprad = !pi*amp/180.0
ludiffeq_23, 'pendulum', tt, yy, t0=0.0, tf=20.0, init=ics, tol=0.0001
yydeg = 180.0*yy[0,*]/!pi
plot, tt, yydeg, yrange=[-180.0,180.0], thick=3.0
```

to select a particular amplitude in degrees and convert it to radians, invoke the solver `ludiffeq_23` with arguments that specify the function defining the differential equations, the variables `tt` and `yy` to be used to store the results, the desired interval for the independent variable, the initial conditions, and a tolerance, and plot the resulting solution. Note

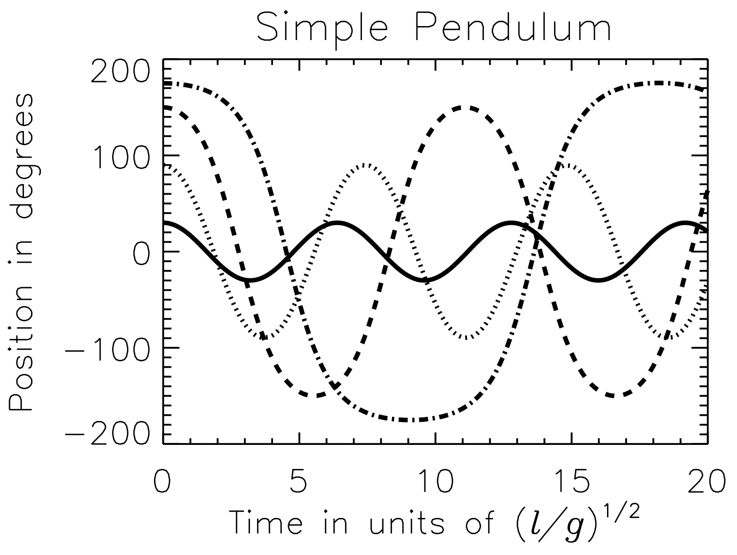

Figure 5: The Motion of a Pendulum for Various Amplitudes.

the conversion between degrees and radians so we can use degrees in talking about and plotting the solution but respect the need for trigonometric functions to be supplied with arguments in radians. Note also that students usually have to explore a bit before deciding on appropriate ranges for the variables involved.

With a bit of embellishment in procedure, students can overplot position as a function of time for several different amplitudes as in Fig 5—and note the clear dependence of period on amplitude. With the solution in hand for a particular amplitude, it is also easy to generate the graphs in Fig. 6, which show position versus time, angular velocity versus time, and angular velocity versus position (the phase plane) for a chosen amplitude.

I like this example because it is relatively easy to set up, it draws attention again to the importance of casting problems in dimensionless form, it offers an opportunity for students to worry about the quantitative accuracy of numerical operations (though I haven't illustrated that aspect here), it draws attention to the utility of the phase plane, and it alerts students to the existence of oscillations that are not simple harmonic.

4.4 Evaluating Integrals Symbolically

Examples involving integrals that can be done symbolically include finding assorted gravitational and electrostatic potential energies from their sources, normalizing some quantum wave functions, evaluating some quantum matrix elements, doing Fourier analyses, and determining coefficients in expansions of known functions in series of orthogonal polynomials. I elect to illustrate with an example that seeks the center of mass and the moment of inertia tensor of a thin plate having the shape of a cardioid, as shown in Fig. 7. With the cusp of the cardioid at the origin and the symmetry axis defining the x axis, the perimeter of this shape is defined in polar coordinates by the equation

$$r(\phi) = a(1 - \cos\phi)$$

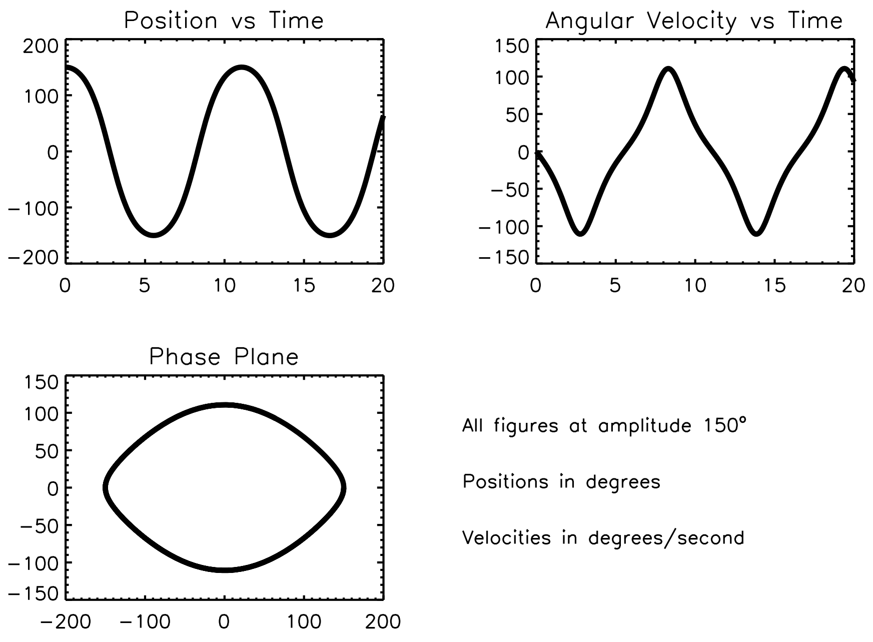

Figure 6: Graphs of θ versus t, ω versus t, and ω versus θ for Amplitude 150°.

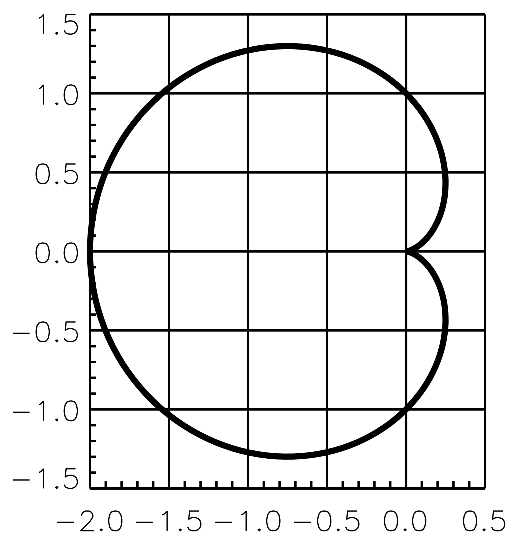

Figure 7: The Cardioid.Both coordinates are measured in units of a.

and the x, y, and z coordinates and the position vector of an element of this plate are given by

$$x = r\cos\phi \quad ; \quad y = r\sin\phi \quad ; \quad z = 0 \quad ; \quad \mathbf{r} = x\,\hat{\mathbf{i}} + y\,\hat{\mathbf{j}} + z\,\hat{\mathbf{k}}$$

As is laid out in class discussion, the position of the center of mass is given by

$$\mathbf{R}_{\mathrm{cm}} = \frac{1}{M}\int_0^{2\pi}\int_0^{a(1-\cos\phi)} \mathbf{r}\,\sigma\,r\,dr\,d\phi$$

and the moment of inertia tensor is given by

$$I = \int_0^{2\pi}\int_0^{a(1-\cos\phi)} \begin{pmatrix} y^2+z^2 & -xy & -xz \\ -yx & x^2+z^2 & -yz \\ -zx & -zy & x^2+y^2 \end{pmatrix} \sigma\,r\,dr\,d\phi$$

Here, σ (which may depend on r and ϕ) is the mass per unit area in the cardioid and M is the total mass of the cardioid. The main difficulties my sophomores encounter stem from their unwillingness to recognize that the order of integration in multiple integrals sometimes matters and that the necessary integrals in this case are not simply integrals over a circle of radius a centered at the origin (which are the limits many students will first apply). Once the students get the limits right, however, their evaluation of the desired quantities takes only a few MAPLE statements. They establish the Cartesian coordinates as functions of the polar coordinates and find the mass of the plate, assuming the mass density to be uniform, with the MAPLE statements

```
x := r*cos(phi); y := r*sin(phi); z := 0;
intM := sigma*r
M1 := int( intM, r = 0..a*(1-cos(phi)) );
mass := int( M1, phi = 0..2*Pi );
```

Then, with the statements

```
pos := [ x, y, z ];
intcm := map( q -> sigma*r*q, pos );
Rcm1 := map( int, intcm, r = 0..a*(1-cos(phi)) );
Rcm := expand( map( int, Rcm1, phi = 0..2*Pi );
```

they set the position vector, define the integrand for the center of mass, and evaluate the defining (double) integral to discover that

$$\mathbf{R}_{\mathrm{cm}} = \left[-\frac{5}{6}a, 0, 0\right]$$

Finally, with the statements

```
tens := array( [ [ y^2+z^2,   -xy,     -xz   ],
                 [   -yx,   x^2+z^2,   -yz   ],
                 [   -zx,     -zy,   x^2+y^2 ] ];
intI := map( q -> sigma*r*q, tens );
I1 := map( int, intI, r = a*(1-cos(phi)) );
Inert := map( int, I1, phi = 0..2*Pi );
Density := solve( M=mass, sigma );
Inertia := eval( Inert, sigma=Density );
```

they set the basic array for the moment of inertia tensor, evaluate the actual integrand, evaluate the double integral, and—for the sake of a simpler appearance that is obviously correct dimensionally—replace the mass density σ with its equivalent in terms of the total mass of the plate. The result is

$$I = \begin{bmatrix} 7Ma^2/16 & 0 & 0 \\ 0 & 49Ma^2/48 & 0 \\ 0 & 0 & 35Ma^2/24 \end{bmatrix}$$

Somewhat to my surprise, this moment of inertia tensor turns out to be diagonal in the chosen coordinates. I am not surprised that the x axis is a principal axis; I am a bit surprised that the y and z axes turn out also to be principal axes.

4.5 Evaluating Integrals Numerically

The remaining major computational topic in the course I am describing focuses on numerical evaluation of integrals. The off-axis potential energy of rings of mass or charge, the off-axis magnetic field of a current loop, and—given an integral definition—the value of special functions like the Bessel functions all provide relevant examples to illustrate the general technique. As with ordinary differential equations, students begin by examining the simplest algorithms and coding simple examples, but they ultimately move to adaptive algorithms built in to IDL. As a single example, I choose the integral

$$T(\alpha) = \frac{2}{\pi} \int_0^{\pi/2} \frac{d\beta}{\sqrt{1 - \sin^2 \frac{\alpha}{2} \sin^2 \beta}}$$

that, in units of $2\pi\sqrt{l/g}$, gives the period T of a simple pendulum as a function of its amplitude α. Before approaching this exercise, students will in class be led through the analytic development that derives this integral, including the important dimensionless casting presented in this slide. To be sure, MAPLE recognizes this integral as an elliptic function and will return that evaluation if asked to do so—and students can be motivated to learn about elliptic functions. At the same time, the evaluation of this integral numerically as a function of the amplitude of the pendulum's motion is instructive because it involves a parameter *in the integrand*, a position that requires a treatment more complicated than that needed when the parameter appears in the limits of integration.

To use IDL, we first define a function—called here `periodint`—to provide the integrand to the integration routine that will ultimately be invoked. A file containing the lines

```
FUNCTION periodint, beta
common param, alpha
RETURN, 1.0/sqrt( 1-sin(alpha/2.0)^2*sin(beta)^2 )
END
```

will play that role. Since the integration routine provides no means to pass the parameter α from the main command level through the routine into this function, which will be called from the routine (not from the main level), we need to invoke—and students need to

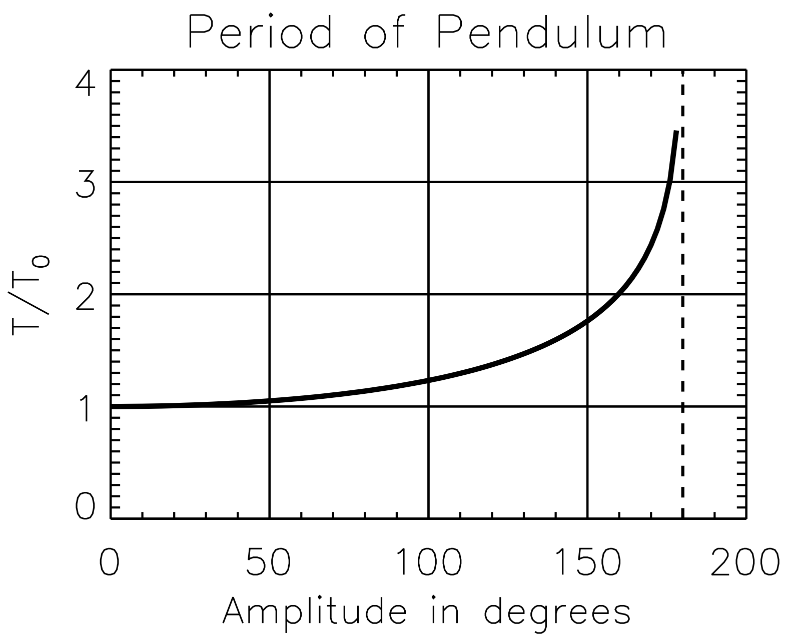

Figure 8: Period of a Pendulum versus Amplitude.

learn about—common storage, which allows access to selected variables in various program segments without passing the values as arguments through the succession of call statements connecting the segments. Once `periodint` is available, however, evaluation of the integral as a function of amplitude, reinstatement of the omitted factor of $2/\pi$, and plotting the result is quite quick, given what students bring to this exercise near the end of the course. With the statements

```
common, param, alpha
ampdeg = 2.0*findgen(90) & amprad = !pi*ampdeg/180.0
period = fltarr(90)
for i = 0, 89 do begin $
   alpha = amprad[i] & $
   period[i] = luqsimp( 'periodint', 0.0, !pi/2.0 ) & $
endfor
period = (2.0/!pi)*period
plot, ampdeg, period, title='...', ...
```

they set up the common area, define a vector containing values of the amplitude ranging in, say, two degree increments from[1] $0°$ to $178°$,create a parallel vector with the amplitudes in radians, create a vector of zeros for the storage of periods, in a loop evaluate the integral for each of the chosen amplitudes, incorporate the factor of $2/\pi$ omitted from the integrand itself, and plot the end result. Again with some embellishments to add labels and grid lines, the resulting graph is as shown in Fig. 8.

[1]We stay away from $\alpha = 180°$ to avoid a divergence in the integration.

5 Conclusions

That's the course and some samples of the exercises students do to practice their skills in using computational approaches to problems in physics. As I said at the beginning, our primary objective in incorporating an early—and substantial—explicit introduction to computation is to make sure our students develop sufficient skills early enough in their studies at Lawrence to assure that, ultimately, they will have the necessary knowledge and the personal confidence to pursue computational approaches fluently, effectively, and independently *on their own initiative.* That we have in substantial measure succeeded in this endeavor is evident in the way our upper-level students continue to use computational resources in their studies and in the extent to which the CPL has become an important facility in our total departmental program.

6 Acknowledgments

I cannot stop without acknowledging some $750K in outside support particularly from the National Science Foundation (1988, 1993, 2000), the W. M. Keck Foundation (1988, 1993, 2001), the Alfred P. Sloan Foundation (1990), and Lawrence University. Further, I must acknowledge support and encouragement over the past 20 years from my departmental colleagues (and especially John R. Brandenberger and the late J. Bruce Brackenridge), 26 Lawrence students who have worked as research assistants, numerous Lawrence students who have experienced computation in various courses, and the 70 faculty members from around the country who participated in one or another of four NSF-supported week-long workshops at Lawrence in the summers of 2001, 2002, and 2003.