Computation in classical mechanics

Todd Timberlake[a)]
Department of Physics, Astronomy, and Geology, Berry College, Mount Berry, Georgia 30149

Javier E. Hasbun[b)]
Department of Physics, University of West Georgia, Carrollton, Georgia 30117

One way to introduce computation into the physics curriculum is to include it in a standard upper-level physics course. A course in intermediate classical mechanics is well-suited for this purpose. We discuss our approach and examples of student projects on solving differential equations and Liouville's theorem, projectile motion on a rotating Earth, motion of a charged particle in electric and magnetic fields, and approximate analytical and numerical solutions for a classical model of a molecule. The projects introduce students to these physics topics and develop the students' computational skills. © *2008 American Association of Physics Teachers.*
[DOI: 10.1119/1.2870575]

I. INTRODUCTION

The purpose of this article is to suggest an approach for incorporating computation into upper-level classical mechanics courses. Many of the topics in such a course are well-suited to a computational approach, and some important topics[1,2] are inaccessible without computation. Classical mechanics provides a wealth of opportunities to use computational tools such as numerical integration, root-finding, numerical solutions of ordinary differential equations, and visualizations of motion in two and three dimensions.

If computation is to be introduced into a standard physics course, it is critical that the computational work support the physics content rather than replace it. One way to get students computing quickly without having to invest much instruction time is to use a standard software package (we use MATHEMATICA and MATLAB). Students can take advantage of the built-in functionality of these programs for numerical computation and for graphics, but we prefer to have students begin by implementing algorithms on their own. This approach minimizes the amount of time spent discussing computation, leaving most of the class time for physics. (We typically spend two to three class meetings discussing software and various algorithms.)

Using even a small amount of class time to discuss computation inevitably means that some physics content must be cut. We focus our computational instruction on a few simple algorithms, rather than covering the most sophisticated algorithms available. The algorithms are presented in a way that allows us to introduce various physics topics, so that we teach about computation as our students are learning new physics. In the same spirit we infuse our physics lectures with computation by presenting computational illustrations of physics concepts. The code for these computational demonstrations is made available to students so they can experiment with it on their own. We assign computational projects that extend the examples we have discussed in the lecture or introduce new topics. We usually require students to submit their code along with any relevant plots and a brief written description of their results. In some cases students are expected to write a formal report (typeset in LATEX with figures and tables) describing their work.

These projects form the heart of our approach to incorporating computation into classical mechanics. The remainder of this article describes four examples of projects that we use.

II. ALGORITHMS AND PHYSICS

To teach students about computation it is essential to introduce them to a few algorithms for carrying out basic computational tasks. Students need to be aware that the choice of algorithm can critically affect the success of the computation. Here we present a way to introduce students to the Euler and the Euler–Cromer algorithms[3,4] for numerically solving ordinary differential equations, while simultaneously introducing them to various physics topics.

For a point particle moving in one dimension and subject to a net force F the Euler–Cromer algorithm is[4]

$$v_{n+1} = v_n + F_n \Delta t / m, \tag{1a}$$

$$x_{n+1} = x_n + v_{n+1} \Delta t, \tag{1b}$$

where x_n, v_n, and F_n represent the position, velocity, and net force at time $t_n = t_0 + n\Delta t$. The Euler–Cromer algorithm is a seemingly minor modification of the Euler algorithm:

$$x_{n+1} = x_n + v_n \Delta t, \tag{2a}$$

$$v_{n+1} = v_n + F_n \Delta t / m. \tag{2b}$$

Students are shown how to implement each algorithm and produce a sequence of positions, velocities, and energies. The most effective way to view the results is to construct plots of the trajectories in phase space, as shown in Fig. 1 for the simple harmonic oscillator. These results illustrate one of the important deficiencies of the Euler algorithm, namely, that it is unstable and does not conserve energy. The Euler–Cromer algorithm, in contrast, produces steady oscillations and conserves the total energy averaged over each cycle of the oscillation. These results[4,5] illustrate the importance of choosing the proper algorithm.

Why does the Euler algorithm fail while the Euler–Cromer algorithm succeeds? The answer to this question leads us to Liouville's theorem, a topic usually discussed in the context of Hamiltonian mechanics.[6] Liouville's theorem states that the dynamics in a conservative system will preserve phase
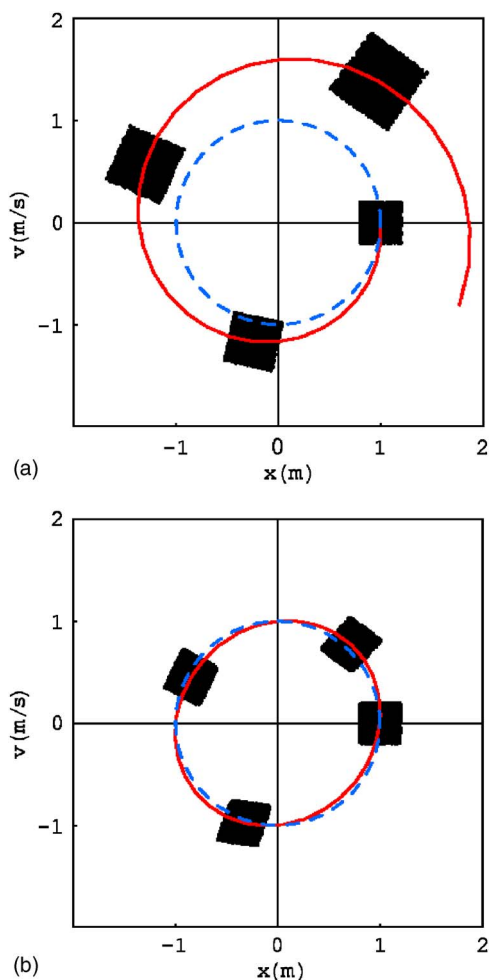
Fig. 1. (Color online) Comparison of (a) Euler and (b) Euler–Cromer algorithms for a simple harmonic oscillator with $k=1$ N/m, $m=1$ kg, $\Delta t=0.2$ s, $x(0)=1$ m, and $v(0)=0$. The dashed line shows the exact orbit. The solid line shows the solution generated by each algorithm. Also shown is a set of trajectories initially distributed at random in a square centered on $x=1$ m, $v=0$. The locations of these trajectories resulting from each algorithm are shown at $t=0$, 1.8, 3.6, and 5.4 s. The period of the oscillator is $T=2\pi$ s.

space volume (or area for a system with one degree of freedom). A standard result from multivariable calculus shows that the infinitesimal phase space area element $dx_n dv_n$ will be transformed under the application of either algorithm to

$$dx_{n+1}dv_{n+1} = |J|dx_n dv_n, \tag{3}$$

where

$$J = \begin{pmatrix} \dfrac{\partial x_{n+1}}{\partial x_n} & \dfrac{\partial x_{n+1}}{\partial v_n} \\ \dfrac{\partial v_{n+1}}{\partial x_n} & \dfrac{\partial v_{n+1}}{\partial v_n} \end{pmatrix} \tag{4}$$

is the Jacobian of the algorithm and $|J|$ is the determinant. According to Liouville's theorem a conservative system should have $|J|=1$ so that phase space area will be preserved. Students can easily show that $|J|=1$ for the Euler–Cromer algorithm and $|J|=1+k\Delta t^2/m$ for the Euler algorithm. Hence, the Euler algorithm does not preserve phase space area, but causes the area occupied by any group of trajectories to grow because $|J|>1$.[7] The change of phase

space area can be illustrated by creating a collection of initial conditions randomly distributed in some small region of phase space. The Euler algorithm will cause the area occupied by this collection of points to grow as shown in Fig. 1(a), while the Euler–Cromer algorithm keeps the area constant, as shown in Fig. 1(b).

The trajectories produced by the Euler algorithm can be obtained from the Euler–Cromer algorithm if we add a force of the form $F=(k\Delta t)v_n$. That is, the Euler algorithm has the effect of introducing an artificial velocity-dependent driving force. Students can convince themselves of this result by deriving the equations for the Euler–Cromer algorithm with the additional force term and comparing these to the equations generated by the Euler algorithm for the simple harmonic oscillator.

We present the foregoing material as a lecture, supplemented with computer demonstrations. At the end of this lecture students are given the code and explore the computations themselves. In particular, students are asked to investigate the effect of reducing the time step Δt. If time permits, they can also compare the results of these algorithms with the results obtained using the built-in ordinary differential equation solver supplied with the software package. In addition, students are assigned a homework problem in which they determine $|J|$ for the Euler–Cromer algorithm when the force is a general function of position and velocity. The result ($|J|=1+(\partial F/\partial v_n)\Delta t/m$) shows that the presence of velocity-dependent forces disrupts the preservation of phase space area. Hence the condition for energy conservation, (which for systems with one degree of freedom is that the force depends only on position), is equivalent to the condition for the preservation of phase space area.

Students are then asked to complete a project in which they modify the algorithms for the harmonic oscillator by adding a linear drag force of the form $F_{\text{drag}}=-bv$ with $b>0$. They use both algorithms to generate phase space trajectories for a variety of parameter values. They also generate plots showing how the phase space area of a collection of trajectories evolves under the action of each algorithm. Additionally, students are asked to compute $|J|$ for each algorithm and each set of parameters. The Euler–Cromer algorithm gives $|J|=1-b\Delta t/m$, and the Euler algorithm gives $|J|=1-b\Delta t/m+k\Delta t^2/m$. The parameter sets are chosen so that students will see underdamped, overdamped, and critically damped motion when they use the Euler–Cromer algorithm. Depending on the parameter values the Euler algorithm can produce trajectories in which the energy decreases, increases, or remains constant on average. In all cases students are expected to comment on whether or not the results are physically realistic. This exercise provides students with experience in interpreting phase space plots and highlights the importance of critically examining the results of any computation to ensure that they are physically reasonable. It also illustrates Liouville's theorem by showing that dissipative motion is associated with the decrease of phase space area.

Students then repeat the same analysis for the quadratic drag force $F_{\text{drag}}=-cv|v|$ with $c>0$. They can show that for this case $|J|=1-2c|v_n|\Delta t/m$ for the Euler–Cromer algorithm and $|J|=1-2c|v_n|\Delta t/m+k\Delta t^2/m$ for the Euler algorithm. Trajectories generated by the Euler algorithm will spiral inward if $|v_n|>k\Delta t/(2c)$, and spiral outward if $|v_n|<k\Delta t/(2c)$. The result is that the trajectories converge to a
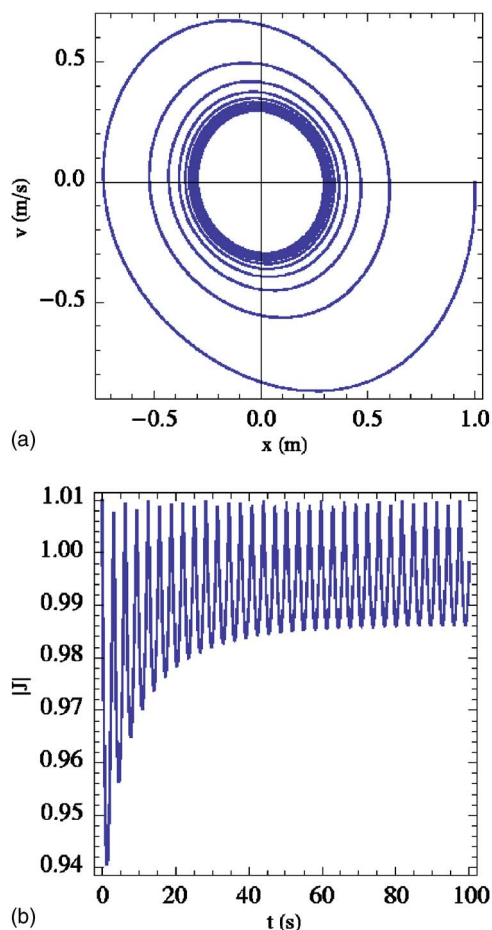
Fig. 2. Plots of (a) the phase space trajectory and (b) the determinant of the Jacobian as a function of time generated by the Euler algorithm for the harmonic oscillator with quadratic drag. The initial conditions are $x(0) = 1$ m and $v(0)=0$. The values of the parameters (as defined in the text) are $k=1$ N/m, $m=1$ kg, $c=0.4$ kg/m, and $\Delta t=0.1$ s.

limit cycle, a phenomenon that is often observed in nonlinear oscillators [see Fig. 2(a)]. Figure 2(b) shows the value of $|J|$ for this trajectory over the same time interval. As the particle oscillates, the value of $|J|$ also oscillates about unity, showing that the limit cycle is stabilized by the balance of the dissipative motion (due to the drag force) and the artificial driving force introduced by the Euler algorithm. The Euler–Cromer algorithm produces the proper physical behavior, but the motion becomes less dissipative as the oscillator's speed decreases. One consequence of this effect is that the harmonic oscillator with quadratic drag does not display critical damping. As part of the project we ask students to explore different parameter values to see if they can produce anything that looks like critical damping in this case. This exercise introduces students to the concept of a limit cycle and the subtle differences between linear and quadratic damping.

Presenting algorithms in this way allows the instructor to introduce physics topics that might not otherwise be discussed in the course, such as Liouville's theorem and limit cycles. This integration of computational methods and physics content can be applied to algorithms for other important numerical problems. For example, some algorithms for numerical root-finding (such as the Newton–Raphson method[8]) can be used to introduce the concept of iterated functions

with stable attractors. This concept is important for understanding the dynamics of dissipative systems like the logistic map.[9]

III. REAL WORLD PHYSICS

Classical mechanics courses are often confined to studying only idealized situations that lead to analytical solutions. Computational assignments can allow students to move beyond these artificial problems to solving "real world" problems. An example is the motion of a projectile on a rotating Earth. The equation of motion in a reference frame fixed to Earth is

$$m\ddot{\mathbf{r}} = m\mathbf{g} + 2m\dot{\mathbf{r}} \times \mathbf{\Omega} + m(\mathbf{\Omega} \times \mathbf{r}) \times \mathbf{\Omega}, \tag{5}$$

where m is the mass of the projectile, $\mathbf{r}$ is the projectile's position, $\mathbf{g}$ is the gravitational field, and $\mathbf{\Omega}$ is Earth's angular velocity. In Cartesian coordinates with the origin at Earth's center and the z-axis through the North (rotational) pole, we can write

$$\ddot{x} = -gx/r + \Omega^2 x + 2\Omega\dot{y}, \tag{6a}$$

$$\ddot{y} = -gy/r + \Omega^2 y - 2\Omega\dot{x}, \tag{6b}$$

$$\ddot{z} = -gz/r, \tag{6c}$$

where $r=\sqrt{x^2+y^2+z^2}$ and $\Omega=|\mathbf{\Omega}|$. The terms proportional to Ω^2 are due to the centrifugal force, and the terms linear in Ω are due to the Coriolis force. Equation (6) cannot be solved analytically, but a computational approach allows us to solve the equations and visualize the results. We assign a project in which students are asked to compute the trajectory of a projectile fired due east from Rome, Georgia with an initial speed of 1600 m/s and a launch angle of 40° above the horizontal.[10] The latitude of Rome is 35.256°N so in spherical polar coordinates (with the origin at Earth's center) the initial polar angle is $\theta_0=55.744°$; for convenience we take the initial azimuthal angle to be $\phi_0=0°$. These initial conditions can be converted into Cartesian coordinates for the numerical computations, and the results can be converted back to spherical polar coordinates. Students are shown an example of how to numerically compute the trajectory (using MATHEMATICA's built-in ODE solver) on a nonrotating Earth and plot the results. To study the relative size of the effects due to each of the two inertial forces students compute a trajectory in which they include the Coriolis force, a trajectory in which they include the centrifugal force, and a trajectory in which both inertial forces are included. Air resistance is neglected, so it is not quite real world physics.

Many students are surprised at the results for the nonrotating Earth because the value of θ changes, indicating that the projectile did not really travel due east. Students can be led to understand that a projectile in this case will follow the path of a great circle that is tangent to the line of constant latitude that passes through the launch point. Only at the equator will a projectile fired due east actually travel due east, even in the absence of effects from Earth's rotation. Projectiles fired due east from a northern latitude will land at a slightly more southern latitude. The deflection due to the centrifugal force results in a slightly increased time of flight (and a slightly increased range) as well as a small southward deflection. The Coriolis force causes a similar but larger deflection. Students should determine the direction of each in-
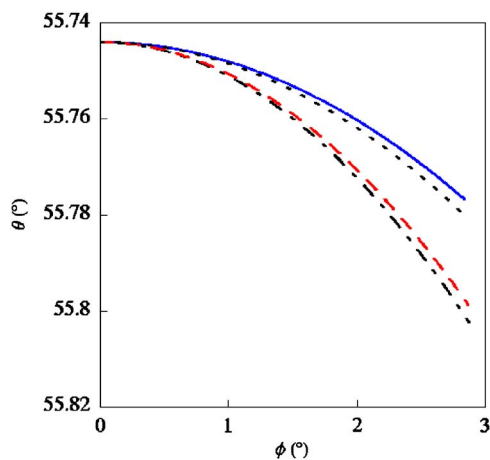
Fig. 3. (Color online) The path of a projectile fired due east from Rome, Georgia. The solid curve shows the path under the influence of gravity only. The dotted curve includes the effects of the centrifugal force. The dashed curve includes the effects of the Coriolis force, and the dot-dashed curve includes the effects of both inertial forces. In each case the curve shows the path of a point on the Earth's surface directly beneath the projectile, plotted in terms of the polar (θ) and azimuthal (ϕ) angles. The scaling of the axes greatly exaggerates the curvature of the projectile's path.

ertial force (using a globe is helpful) and match this direction to the computed deflection. The numerical results will show that the deflection due to the combined inertial effects is almost equal to the sum of the deflections from each effect individually. This simple additive behavior might seem counterintuitive because the inertial forces are coupled, but students can be led to understand that the centrifugal force is effectively constant (it depends on the distance from Earth's center, which does not change appreciably during the projectile's flight) and very small (so it doesn't significantly alter the velocity components which determine the Coriolis force). Figure 3 shows the motion of a point on Earth's surface directly below the projectile for each of the cases studied.

This project can be extended in several ways. Students can be asked to predict what would happen if the projectile had been fired due west rather than due east. Most students will correctly predict that the overall deflection due to Earth's rotation will be northward, but they may not realize that the projectile will still land at a latitude south of its launch point because of the southward deflection that occurs even in the absence of Earth's rotation. Students also might not realize that the westward-fired projectile will have its time of flight and range shortened by the effects of Earth's rotation. The project can be extended by accounting for air resistance or the fact that Earth is an oblate spheroid rather than a sphere.

Computation allows for the solution of more realistic problems in less familiar contexts as well. Consider the motion of a charged particle in constant electric and magnetic fields. Computation allows students to explore the motion of a charged particle in electromagnetic fields of any geometry. Figure 4 shows a JAVA application[11] based on an earlier MAT-LAB script, which simulates the motion of a charged particle in uniform electric and magnetic fields. Students input parameters such as the field components, initial velocity components, and the charge and mass of the particle. The program then numerically solves the equations of motion and generates a plot of the three-dimensional trajectory as well as graphs of each coordinate as a function of time. Students can investigate the dynamics by reproducing the simple circular and helical motions described in most texts, and then can move on to more interesting motions by changing the com-
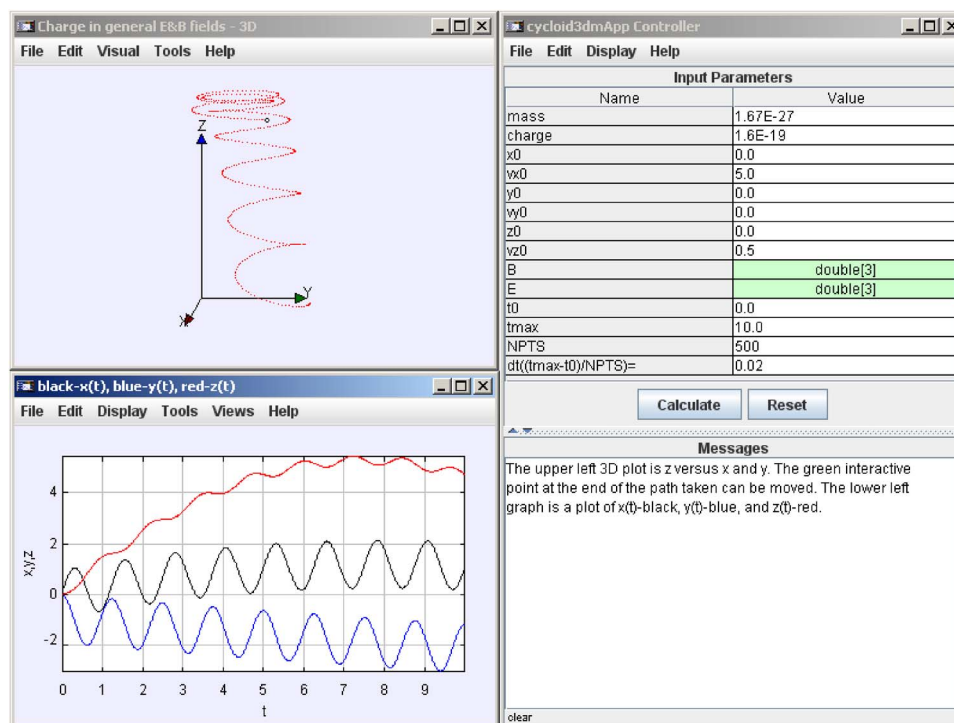

Fig. 4. (Color online) Output of an Open Source Physics program for the motion of a charged particle in the presence of constant electric and magnetic fields. The values of the electric and magnetic field components are $E=[5\times10^{-9}, 10^{-9}, -3\times10^{-9}]$, and $B=[10^{-8}, -10^{-9}, 5.13\times10^{-8}]$ in units of V/m, and Tesla, respectively. The particle is a proton with the initial conditions $x_0=0$, $v_{x0}=5$ m/s, $y_0=0$, $v_{y0}=0$, $z_0=0$, and $v_{z0}=0.5$ m/s.

ponents of the fields. They can also explore the effects of changing the particle's mass and charge, thereby gaining an understanding of how a particle can be identified by its track in a cloud chamber. These investigations can be extended to nonuniform fields by modifying the method that specifies the derivatives.

IV. COMPUTATION AND APPROXIMATION

When faced with a problem that has no analytic solution, there are two general approaches—solve the problem numerically or find an approximate solution. Physics students need to be familiar with both approaches and should also be aware of how these two approaches can complement each other. The approximate solution is applicable to a variety of initial conditions and parameter values, and the numerical solution must be reconstructed for each new set of values. An approximate solution is useful only if it provides the desired level of accuracy, which might be difficult to assess. One approach is to construct an approximate solution and compare it to a numerical solution for specific sets of initial conditions and parameter values. Such a comparison can reveal the limitations of the approximation and can also serve to detect errors in the analysis that led to the approximate solution. This approach assumes that an accurate numerical solution can be found.

As an example of combining approximation and computation, consider a one-dimensional model of a diatomic molecule with the potential

$$V(x) = u_0 a_0^2 \left(\frac{a_0}{x^3} - \frac{1}{x^2} \right), \tag{7}$$

where u_0 is a unit of energy, a_0 is a molecular distance, and x represents the distance between the two atoms in the molecule. We can determine the bond length x_b for this molecule by letting $V'(x_b)=0$ with the result $x_b=3a_0/2$. To obtain an approximate solution for the vibrations of the molecule about this equilibrium length we expand the force $F(x)=-V'(x)$ to second order in a Taylor series about $x=x_b$:

$$F(x) \approx -\frac{32 u_0}{81 a_0^2}(x - x_b) + \frac{256 u_0}{243 a_0^3}(x - x_b)^2. \tag{8}$$

If we neglect the second-order term, then the motion will be simple harmonic with angular frequency $\omega_0 = \sqrt{32 u_0/81 \mu a_0^2}$, where $\mu = m_1 m_2/(m_1 + m_2)$ is the reduced mass of the two atoms. If we take the molecule to be at rest at $t=0$ with the atoms a distance x_0 apart, this linear approximation leads to

$$x(t) \approx x_l(t) = x_b + (x_0 - x_b)\cos(\omega_0 t). \tag{9}$$

To obtain a more accurate approximation we rewrite Newton's second law using Eq. (8) to find

$$\ddot{x} + \omega_0^2(x - x_b) \approx \frac{8 \omega_0^2}{3 a_0}(x - x_b)^2. \tag{10}$$

By using the method of successive approximations[12] we construct an approximate solution of Eq. (10):

$$x(t) \approx x_{\text{nl}}(t) = x_b + A_1 \cos(\omega_0 t) + \frac{4 A_1^2}{9 a_0}[3 - \cos(2\omega_0 t)]. \tag{11}$$

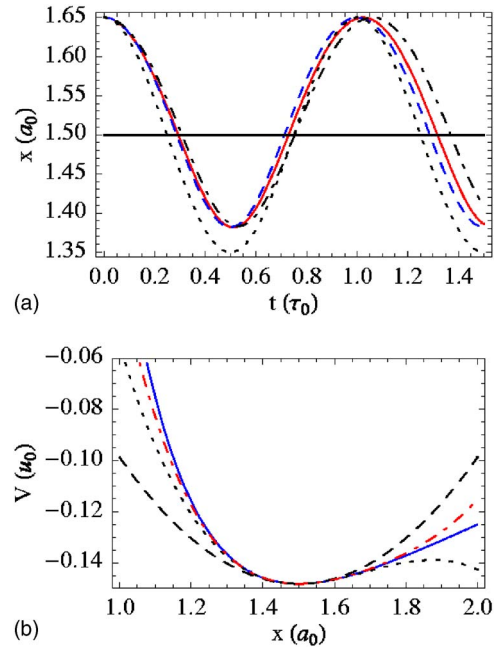From the requirement that $x_{\text{nl}}(0)=x_0$ we find

Fig. 5. (Color online) The solid curve in (a) shows the numerical solution for the atomic separation as a function of time in a model of molecular vibration. Also shown are the results of a linear approximation (dotted), a nonlinear approximation (dashed), and the numerical solution of Eq. (10) (dot-dashed). The initial atomic separation is $x_0=1.65$ and the equilibrium separation is $x_b=1.5$ (indicated by the horizontal line). The solid curve in (b) shows the exact potential $V(x)$ [see Eq. (7)]. Also shown are approximate potentials derived from a Taylor series expansion of $V(x)$ to second-order (dashed), third-order (dotted), and fourth-order (dot-dashed).

$$A_1 = \frac{9 a_0}{16}\left[-1 + \sqrt{1 + \frac{32(x_0 - x_b)}{9 a_0}} \right]. \tag{12}$$

The approximate solutions given in Eqs. (9) and (11) can be compared to the numerical solution of

$$\mu \ddot{x} = F(x) = -V'(x) = u_0 a_0^2 \left(\frac{3 a_0}{x^4} - \frac{2}{x^3} \right). \tag{13}$$

It is convenient to define units so that length is measured in terms of a_0, energy in terms of u_0, and time in terms of $\tau_0 = 2\pi/\omega_0$. In these units $u_0=1$, $a_0=1$, $\omega_0=2\pi$, and $\mu = 8/(81\pi^2)$. Figure 5(a) shows a comparison of the three solutions for $x_0=1.65$, and the numerical solution of Eq. (10). The nonlinear approximation does a better job of capturing the asymmetry in the oscillations than the linear approximation, which is not surprising because the second-order Taylor series expansion of $V(x)$ is symmetric about $x=x_b$ and the third-order expansion of $V(x)$ mimics the asymmetry of the full potential [see Fig. 5(b)]. Both the nonlinear and linear approximations overestimate the frequency of the oscillations. This overestimate is surprising for the nonlinear approximation, because the third-order expansion of the potential is less steep than the full potential in the vicinity of the equilibrium point, meaning that the restoring force for the third-order potential is weaker than for the full potential. We would expect an approximate solution that makes use of the third-order expansion of $V(x)$ to produce a frequency that is lower than the true frequency. A numerical solution of Eq. (10) produces oscillations with a frequency less than that of the numerical solution for the full potential [see Fig. 5(a)].

So the overestimation of the frequency in Eq. (11) is a result of the method of successive approximations. If we include the fourth-order correction, we find a curve that matches the complete potential over a wider range, as shown in Fig. 5(b); a numerical solution using this fourth-order potential more closely matches the exact solution. By examining the effects of higher-order terms in the Taylor series expansion of $V(x)$ students can see how each additional term produces an improved approximation to the potential, and also see how including each additional term leads to an improved solution for $x(t)$. This problem shows how computation can be used to analyze the validity of approximations and solutions derived from them.

V. DISCUSSION

Instructors looking for additional projects and other computational materials can obtain a variety of materials for use with MATLAB[13] or MATHEMATICA.[14] One of us (JH) has authored a textbook on classical mechanics that explicitly incorporates computation and suggestions for computational projects.[15] Recent editions of more traditional classical mechanics texts[1,12,16] also contain suggestions for computational projects. There are also a variety of computational physics texts[3,17] that are sources for project ideas.

[a]Electronic mail: ttimberlake@berry.edu

[b]Electronic mail: jhasbun@westga.edu

[1] John R. Taylor, *Classical Mechanics* (University Science Books, Sausalito, CA, 2005), Chap. 12.

[2] Todd Timberlake, "A computational approach to teaching conservative chaos," Am. J. Phys. **72**, 1002–1007 (2004).

[3] Harvey Gould, Jan Tobochnik, and Wolfgang Christian, *An Introduction to Computer Simulation Methods: Applications to Physical Systems* (Addison–Wesley, New York, NY, 2007), 3rd ed., p. 75.

[4] Alan Cromer, "Stable solutions using the Euler approximation," Am. J. Phys. **45**, 455–459 (1981).

[5] See for example, Ref. 3, pp. 45–46.

[6] Louis N. Hand and Janet D. Finch, *Analytical Mechanics* (Cambridge U.P., Cambridge, 1998), pp. 202–204.

[7] Algorithms that preserve phase space area, like the Euler–Cromer algorithm, are called *symplectic* algorithms (see Ref. 3, pp. 82–84). The Euler algorithm is not symplectic.

[8] See for example, Ref. 3, p. 191.

[9] See for example, Ref. 3, pp. 141–166.

[10] This extraordinary initial speed is the muzzle velocity of the "Paris Gun" that was used by the German army to bombard Paris during World War I. See ⟨www.daviddarling.info/encyclopedia/P/Paris_Gun.html⟩ and ⟨www.britannica.com/eb/article-9105701/Paris-Gun⟩ for more details.

[11] See ⟨www.opensourcephysics.org⟩ and Wolfgang Christian, *Open Source Physics: A User's Guide with Examples* (Pearson Addison–Wesley, San Francisco, CA, 2007).

[12] Javier E. Hasbun, *Classical Mechanics with MATLAB Applications* (Jones and Bartlett, Sudbury, MA, 2008), Sec. 4.14.

[13] MATLAB code is available upon request by e-mailing the author (J.H.). Open Source Physics applications not included in Ref. 12 will be made freely available at ⟨www.westga.edu/~jhasbun/osp/osp.htm⟩ in tandem with the textbook release.

[14] MATHEMATICA code is available at ⟨facultyweb.berry.edu/ttimberlake/comp_phys/⟩.

[15] See Ref. 12 and ⟨www.jbpub.com/catalog/0763746363/⟩.

[16] Stephen T. Thornton and Jerry B. Marion, *Classical Dynamics of Particles and Systems* (Thomson–Brooks/Cole, Belmont, CA, 2004), 5th ed.; Grant R. Fowles and George L. Cassiday, *Analytical Mechanics* (Thomson–Brooks/Cole, Belmont, CA, 2005), 7th ed.

[17] Rubin H. Landau and Manuel J. Paez, *Computational Physics: Problem Solving with Computers* (Wiley, New York, NY, 1997); Steven E. Koonin and Dawn C. Meredith, *Computational Physics: Fortran Version* (Addison–Wesley, Reading, MA, 1990); Samuel S. M. Wong, *Computational Methods in Physics & Engineering* (Prentice Hall, Englewood Cliffs, NJ, 1992); Alejandro L. Garcia, *Numerical Methods for Physics* (Prentice Hall, Upper Saddle River, NJ, 2000), 2nd ed.; Nicholas J. Giordano and Hisao Nakanishi, *Computational Physics* (Prentice Hall, Upper Saddle River, NJ, 2006), 2nd ed.; Paul L. DeVries, *A First Course in Computational Physics* (Wiley, New York, NY, 1994); David M. Cook, *Computation and Problem Solving in Undergraduate Physics* (Department of Physics, Lawrence University, Appleton, WI, 2003).