Paper ID #37483

A Path to Computational Thinking and Computer Programming through Physics Problems

Robert H Mason (Professor)

I graduated with a B.S. in physics (1995) and an M.S. in physics (1997) from Southern Illinois University, Edwardsville. I taught as an adjunct instructor at various schools around Edwardsville for the 1997-1998 academic year before accepting a full-time position at Moberly Area Community College in Moberly, MO. After a year in Missouri I took my current position at Olney Central College in Olney, IL (1999). Over the last 23 years I have taught a wide range of courses; mathematics, physics, physical science, and pre-engineering. I found the pre-engineering courses to be especially rewarding due to the diversity and the rigor of the material. In 2016 I completed an M.A. in math education at Eastern Illinois University because of a keen interest in math as well as the need for a program that was flexible enough to accommodate my teaching schedule. This was a good decision as the focus on pedagogy was invaluable in the classroom, even with my experience. around that same time I became aware of a group called Partnership for Integration of Computation into Undergraduate Physics (PICUP) and started attending workshops. My experiences motivated me to pursue my doctorate in applied physics at Southern Illinois University, Carbondale, beginning in 2020. It is my work at SIUC that has introduced me to the ASEE.

Hansika Sirikumara

Hansika Sirikumara, Ph.D., is an Assistant professor of Physics and Engineering at E. S. Witchger School of Engineering, Marian University Indianapolis. She completed her MS and PhD degrees from Southern Illinois University Carbondale. Her research expertise/interests are in engineering material properties for semiconductor device applications using computational methods.

A Path to Computational Thinking and Computer Programming through Disciplinary-Based Problems

Robert H Mason

Olney Central College, Olney, IL and Schools of Physics and Applied Physics, Southern Illinois University Carbondale, Carbondale, IL

Dr. Hansika I Sirikumara

E.S. Witchger School of Engineering, Marian University, Indianapolis, IN

Dr. Thushari Jayasekera

School of Physics and Applied Physics, Southern Illinois University Carbondale, Carbondale, IL

Abstract

We propose a novel approach to improve computational thinking (CT) and computer programming (CP) via a disciplinary-based problem solving approach. We hypothesize that the syntax needed for writing a computer code can be introduced through support programs in problem solving. This article demonstrates this approach through the problem of non-linear pendulum, a familiar problem solved in most engineering and physics classes. It is basically focused on implementing numerical techniques for solving ordinary differential equations (ODEs). Instead of using a syntax-based teaching approach, support programs can be used to identify the required coding elements. We envision that this approach can be transferable to many problems in STEM classes to motivate students towards computer programming.

1 Introduction

The proliferation of computer science into STEM fields has resulted in a high demand for STEM jobs in computing. Thus, it is crucial that students majoring in STEM subjects other than Computer Science (non-CS) should have easily accessible resources to develop CT and practice CP, which is a skill they should develop like reading, writing, and solving algebraic equations.[1,2,3,4]. We hypothesize that the use of support programs through the problem solving instead of traditional syntax-based approach for teaching computer programming for those who have no prior-knowledge of computer programming.

2 Explanation of the Problem and Computational Details

The dynamics of the simple pendulum consisting of a mass, m attached to the end of a massless string of length L is given by[5]:

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

Notice that we use t' as the time, which is measured in seconds.

By the conversion, $t \rightarrow \frac{t'}{\sqrt{l/g}}$, above equation changes to:

$$\frac{d^2\theta}{dt^2} = -\sin\theta \tag{1}$$

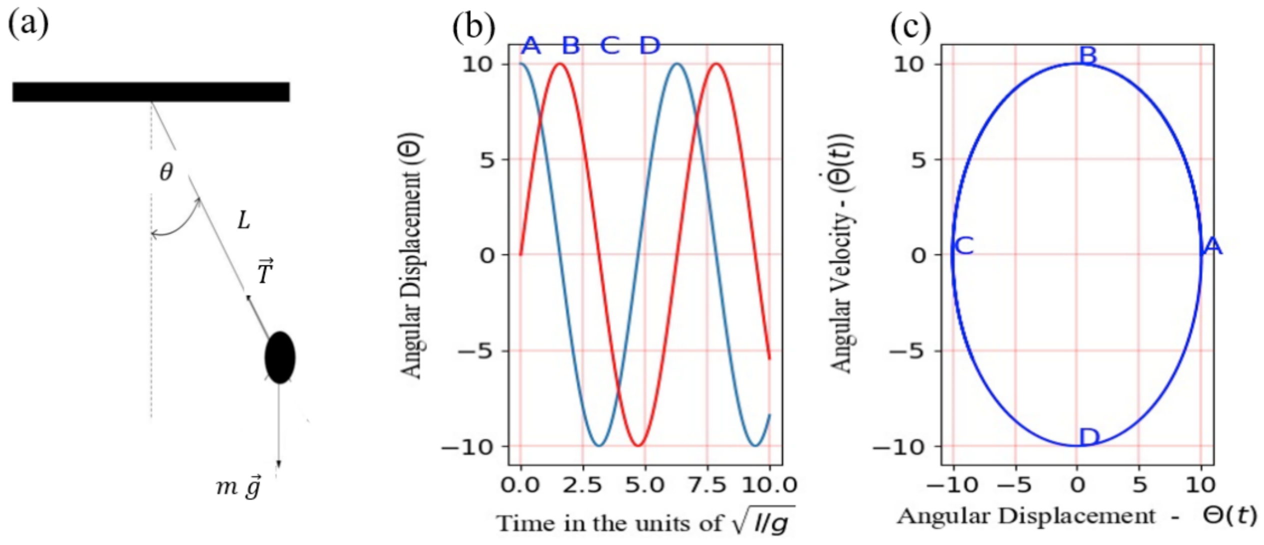

Figure 1: (a)Schematic diagram of a simple pendulum. (b) θ (blue) and $\dot{\theta}$ (red) as a function of time (c) the phase space diagram, i. e. angular velocity $\dot{\theta}$ vs θ.

Now t is measured in terms of $\sqrt{l/g}$,i.e. t in the above equation is dimensionless.

This is a non-linear second order ordinary differential equation, which is hard to be solved analytically. In the small angle approximation, for which, $\sin\theta \sim \theta$

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

, which has a solution of the form, $\theta = \theta_0 \sin(t + \phi)$, where θ_0 and ϕ are determined using the boundary conditions. If the pendulum bob is released at an angle θ_0 with a zero velocity, θ_0 is the initial angle and ϕ is zero.

Here, θ is a periodic function of t. We can clearly visualize the periodic nature in θ vs t graph. Additionally, we can plot the phase space diagram, which also shows the periodic nature of the motion as shown in Fig. 1.

The phase space diagram shown in the Fig. 1 shows that the pendulum bob passes the same position with same speed, which assures the periodic motion. At time $t = 0$, pendulum bob is released from rest at an angle $\theta_0 = 10^o$. This starting position is marked by A in the plots. Position B is the central position (where $\theta = 0$), when the pendulum bob passes with the highest speed and reach the maximum angle (i.e. θ_0) at position C. The speed of the pendulum bob reaches zero at position C and changes the direction of motion. It will reach the original position A via the central point D.

The dynamics of the simple pendulum for large angles described by eq. 1 is a non-linear second order ODE, which can be solved using numerical approaches. Here, students can be guided to numerically solve the first order ODEs and second order ODEs using support programs. Students will be guided to improve CT through applying the numerical techniques to solve non-linear pendulum.

3 Numerical Solutions to First Order ODEs

Support Program Solve the following equation numerically with the boundary condition at t=0, N=3000. Take $\tau = 5s$.

$$\frac{dN}{dt} = -\frac{N}{\tau} \tag{2}$$

Write a computer code to plot N as a function of t, using different numerical algorithms, and compare the solution with the analytic solution.

This equation has an analytic solution[7], $N(t) = N_0 e^{(-t/\tau)}$ where the constant, N_0 is determined using the boundary condition of the physical problem, and τ is a characteristic time constant of the system. Even though this equation has an analytic solution, students can be guided to learn numerical solutions to first order ODEs using this problem.

First the independent axis t will be descretized for the time axis $t_{initial}$ to t_{final} with a step size Δt.

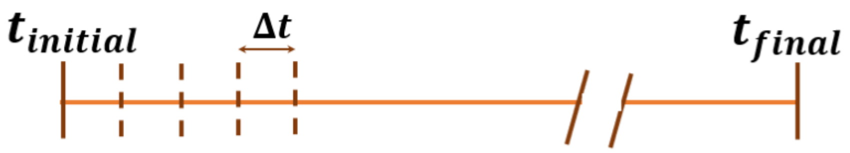

Figure 2: The axis of independent variable is discretized in steps of Δt.

The goal of a numerical algorithm is to evaluate $N(t + \Delta t)$ for all grid points. There are different algorithms to do this task. Here We will use two numerical algorithms without their mathematical derivation, which can be found elsewhere[6].

Euler Algorithm $N(t + \Delta t)$ relates to $N(t))$ as

$$N(t + \Delta t) = N(t) + \Delta t f(N(t), t),$$

Fourth Order Runge-Kutta (RK4) Method finds,

$$N(t + \Delta t) = N(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

where,

$$k_1 = \Delta t f(N, t)$$

$$k_2 = \Delta t f(N + \frac{1}{2}k_1, t + \frac{1}{2}\Delta t)$$

$$k_3 = \Delta t f(N + \frac{1}{2}k_2, t + \frac{1}{2}\Delta t)$$

$$k_4 = \Delta t f(N + k_3, t + \Delta t)$$

where $f(N, t) = -(N(t))/\tau$ for this problem defined by eq. 2.

Implementation of the algorithm The numerical algorithms can be implements in python as:

```
# Input Parameters
tau=1 # Known for the given problem
N=3000 # given - the N at time t=0
tinitial=0.0
tfinal=3.0 # Notice this t is now given in τ
deltat=0.05 # Defined by the programmer, which must assure the convergence of the results.
import numpy as np
tlist=np.arange(tinitial,tfinal,deltat)
# Numerical Solutions -Euler Algorithm
Nlist=[] # creating a list for storing the data
def fxn(N,t):
        return -N/tau
for t in tlist:
        Nlist.append(N)
        N+=deltat*exp(-t/tau)
# Plotting Data
import matplotlib.pyplot as plt
plt.plot(tlist,Nlist)
plt.show()
```

The results for eq. 2 , displayed in Fig. 3, show the increase in the accuracy for different algorithms.

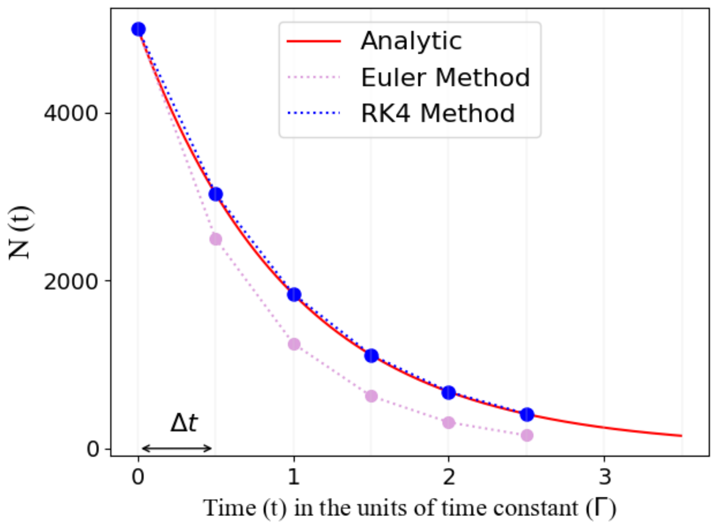

Figure 3: Analytical and Numerical solutions for the eq. 2

In a similar way, RK4 algorithm can be implemented in 4 steps as follows.

```
# RK4 Algorithm:
for t in tlist:
    Nlist.append(N)
    $k1 = \Delta t$ fxn(N,t)
    $k2 = \Delta t$ fxn(N+$\frac{1}{2}k1, t + \frac{1}{2}\Delta t)$
    $k3 = \Delta t$ fxn(N+$\frac{1}{2}k2, t + \frac{1}{2}\Delta t)$
    $k4 = \Delta t$ fxn(N+k3, t+$\Delta t)$
    $N+ = \frac{1}{6}(k1 + 2k2 + 2k3 + k4)$
```

4 Numerical Solutions to Coupled First Order ODEs

Support Program (Numerical solutions to coupled first order ODEs.)
Consider the following two coupled ODEs.

$$\frac{dx}{dt} = -0.3x + 0.8y \quad \text{and} \quad \frac{dy}{dt} = 0.5x - 0.25y \tag{3}$$

which can be written in the matrix form:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -0.3 & 0.8 \\ 0.5 & -0.25 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \quad \text{which can be written as} \quad \frac{d}{dt}\vec{X} = A\vec{X}$$

This set of equation has a solution:
x(t)=$\alpha\ a_{11}e^{\lambda_1 t} + \beta a_{21}\ e^{\lambda_2 t}$ and $y(t) = \alpha\ a_{12}e^{\lambda_1 t} + \beta a_{22}\ e^{\lambda_2 t}$
where λ_1 and λ_2 are the eigenvalues of the matrix A and the coefficients (a_{ij}) are its eigenvectors. The constants α and β are determined by the boundary conditions.
Write a program to numerically solve the above two ODEs. Plot the solutions along with the analytic solution for few selected initial conditions.

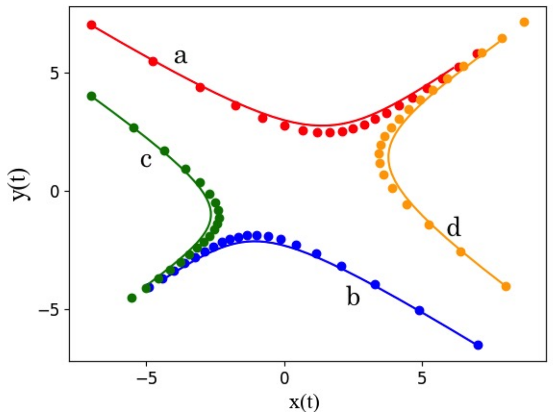

Figure 4: Solution to the coupled differential equation (eq.3) with 4 different sets of boundary conditions. The solid lines are the analytic solution and the dotted lines are the numerical solutions

Figure 4 shows the results of eq.3, which were numerically solved for different boundary conditions: (a) x=-7,y=7 (red), (b) x=7,y=-6.5 (blue), (c) x=-7,y=4 (green) and (d) x=8,y=-4 (orange). The implementation of the algorithm (Euler) is shown below.

```
import numpy as np
# Input Parameters:
x,y=-7,7
tinitial,tfinal=0,10.0
deltat=0.01
# Diescretization of time axis
tlist=np.arange(tinitial,tfinal,deltat)
def fxnxy(x,y):
        fxnx=-0.3x+0.8y
        fxny=0.50x-0.25y
        return fxnx,fxny
# Euler Algorithm
xlist=[]
ylist=[]
for t in tlist:
        xlist.append(x)
        ylist.append(y)
        x+=Δt * fxnxy[0]
        y+=Δt * fxnxy[1]
# Euler Algorithm
from matplotlib import pylab as pl
pl.plot(xlist,ylist)
pl.show()
```

5 Numerical Solutions to Non-Linear Simple Pendulum

Dynamics of the non-linear pendulum is described by a second order ODE given in eq. 1. A common approach to numerically solve a second order ODE is to consider it as two coupled first order ODEs. The eq. 1 can be considered as:

$$\frac{d\theta}{dt} = \omega \qquad \text{and} \qquad \frac{d\omega}{dt} = -\sin\theta \tag{4}$$

These two first order ODEs are coupled through time. We can solve coupled first order ODEs using any algorithm discussed in the previous section. The assignment of applying the numerical solutions to solve eq. 1 will improve student's Computational Thinking.

The angular displacement (θ) vs. time (t) and the phase space diagram for the results obtained for pendulum with initial angle $\theta = 10^0$ using $\Delta t = 0.01$ and $\Delta t = 0.001$ plotted with analytic solutions are shown in Fig. 5. It appears that (Fig.6) when we use a time step, $\Delta t = 0.01$, numerical results generated with RK4 algorithm agree with the analytic solution for small angles. However, Euler method does not seem to converge with $\Delta t = 0.01$.

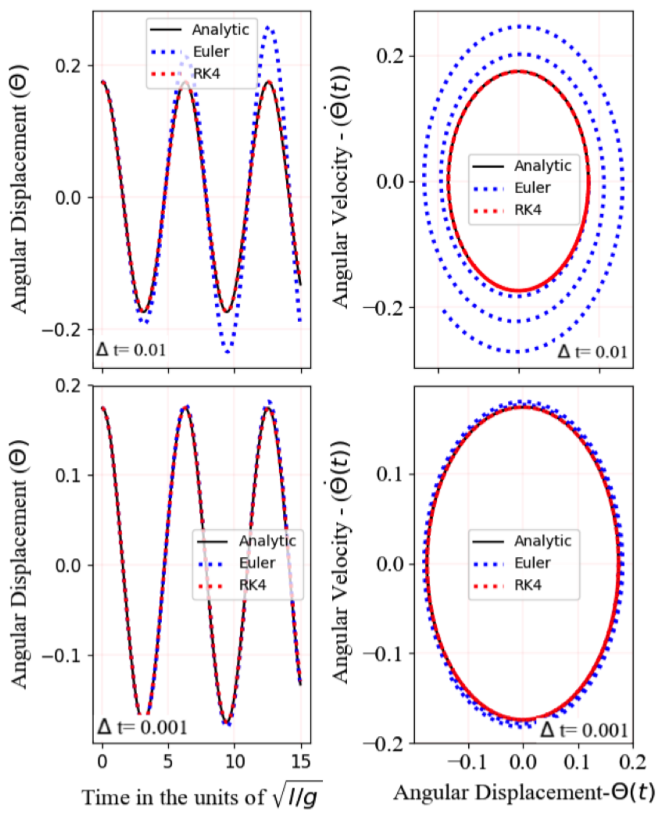

Figure 5: Numerical and Analytical solutions to nonlinear simple pendulum with initial angle $\theta_0 = 10^0$. Upper and lower panels show the results obtained with two time steps $\Delta t = 0.01$ and $\Delta t = 0.001$.

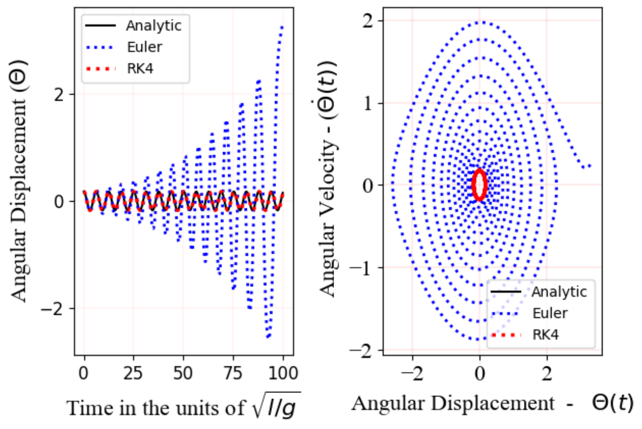

Figure 6: Numerical and Analytical solutions to non-linear simple pendulum with initial angle $\theta_0 = 10^0$ using $\Delta t = 0.01$

Numerical Instability: When we analyze the solution for the longer time period (Fig. 6), we note that the solutions with Euler method becomes highly unstable. This numerical instability is a result from accumulation of error through the computation, which has been discussed. We can conclude that the Euler method is not suitable for solving the nonlinear simple pendulum. Previous studies have used a improved Euler-Cromer algorithm which fixes that numerical instability. However, we did not include that in this discussion. This exercise clearly demonstrate the importance of checking the convergence of all calculation parameters before we produce realistic results.

Now we will move on with RK4 algorithm. The results of nonlinear simple pendulum for large angles is shown in Fig. 7. Unlike in the small angle approximation, the time period strongly depends on the maximum angle of oscillations. The variation of time period follows the data reported in the literature[8].

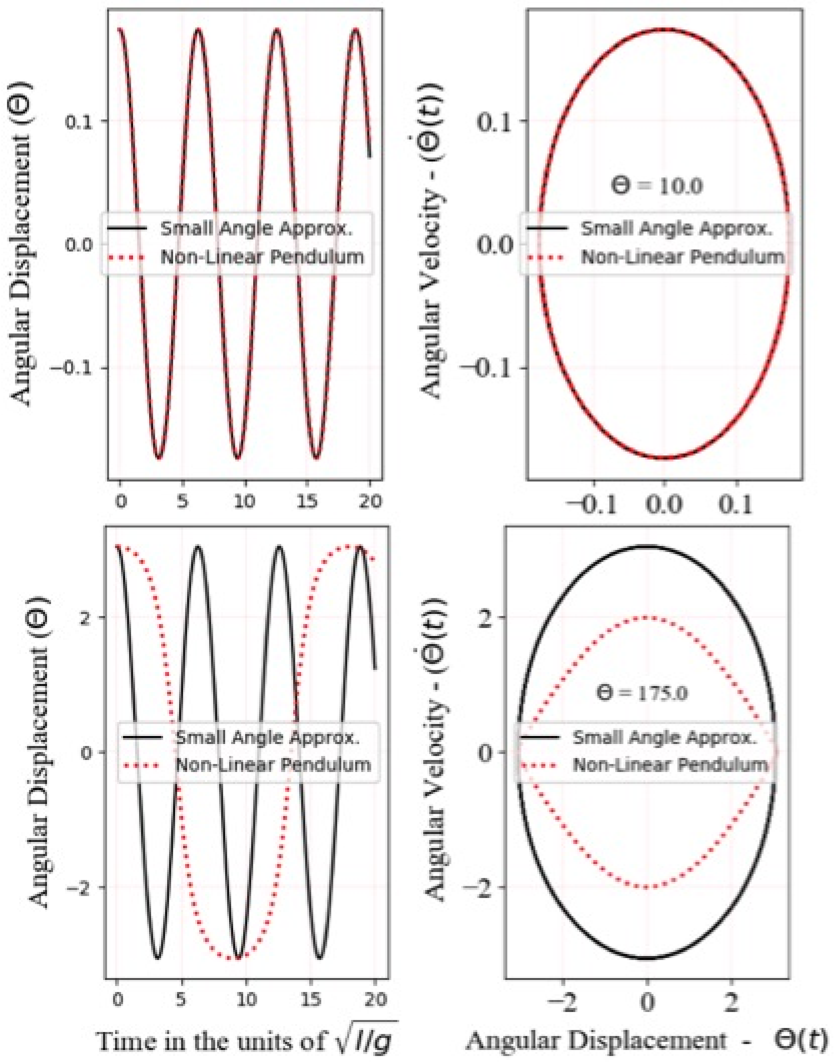

Figure 7: Angular displacement vs. time (left) and the angular velocity (right) as a function of time for nonlinear simple pendulum. Results with maximum angle $\theta_0 = 10^0$ and $\theta_0 = 175^0$ are shown in upper and lower panels.

6 Conclusion

We provided a novel approach with support programs to identify required coding elements to teach computer programming for non-CS STEM majors We demonstrated this approach with the problem of simple pendulum, a routinely taught problem in upper level engineering and physics classes. We envision that similar exercises can be developed to improve CT and CP among non-CS STEM major students in upper division undergraduate students or starting graduate students.

References

[1] Li, Y., Schoenfeld A. H., diSessa A. A., Graesser A. C., Benson L. C., English L. D., Duschl R. H., “ On Computational Thinking and STEM Education”, Journal of STEM Education Research, 3, 147, (2020).

[2] Lyon J. A., Magana, A. J., “Computational Thinking in Higher Education: A Review of the Literature”, Computer Applications in Engineering Education, **28**, 1174, (2020).

[3] Swaid, S.I, ”Bringing Computational Thinking to STEM Education”, Procedia Manufacturing, **3**, 3657, (2015).

[4] Wang J. M., ”Computational Thinking”,Commun. ACM., **49**, 33-35, (2014).

[5] Thornton, S. T., Marion J. B., ” Classical Dynamics of Paticles and Systems”, Thomson Learning (2003).

[6] Newman,M., “Computational Physics”, (2012).

[7] Giordano, N. J., Nakanishi H., “Computational Physics”, Pearson Education Inc. (2006).

[8] Kidd R. B., Fogg S. L., “A Simple Formula for the Large Angle Pendulum Period”, Physics Teacher, **40**, 81, (2002).