Georgia Journal of Science

Volume 76 No.2 Scholarly Contributions from the Membership and Others

Article 9

2018

A Nonlinear Approximate Solution to the Damped Pendulum Derived Using the Method of Successive Approximations

Justin A. Hill
University of West Georgia, Carrollton, GA 30118

Javier E. Hasbun 8394092
University of West Georgia, jhasbun@westga.edu

A Nonlinear Approximate Solution to the Damped Pendulum Derived Using the Method of Successive Approximations

Acknowledgements
This work was supported by a grant from SEEP (STEM Education Improvement Plan)

A NONLINEAR APPROXIMATE SOLUTION TO THE DAMPED PENDULUM DERIVED USING THE METHOD OF SUCCESSIVE APPROXIMATIONS

Justin A. Hill and Javier E. Hasbun*
Department of Physics
University of West Georgia
Carrollton, Georgia, 30118
(Dated: July 9, 2018)
*Corresponding author
E-mail: jhasbun@westga.edu

ABSTRACT

An approximate analytic solution to the damped pendulum is derived using the method of successive approximations to obtain a nonlinear approximation for the system. We take the approximate solution to the undamped pendulum using the method of successive approximations and compare it to the damped pendulum solution when a linear approximation is used. By looking at these two solutions, we can make an educated guess about the form of the general, approximate solution to the nonlinear damped pendulum. By adjusting the initial guesses and the initial conditions, we derive approximate solutions in three ways. Using MATLAB, the approximate solutions are compared to the full numerical solution through the Euler-Cromer method. To determine how accurate the approximations are, the errors of the approximations are calculated relative to the full numerical Euler-Cromer solution. Each new approximation came with a significant decrease in error, with the final error being 0.0099. This resulted in an improvement to the method of successive approximations. Finally, our best approximation is compared to an available and previously published work.

Keywords: pendulum, nonlinearity, successive approximation, damped pendulum, analytic solution, numerical solution, matlab, octave

INTRODUCTION

The undamped pendulum system differential equation involves a nonlinear term ($\sin(\theta)$), as discussed further below, and when $\sin(\theta)$ is approximately θ, there is a simple analytic solution. However, when the $\sin(\theta)$ term is kept, there is not a closed form solution, albeit there is a well-known infinite series solution (Boas 2006). We are interested in pursuing a damped version of the pendulum whose general solution is unknown, yet an approximate solution can be found through the method of successive approximations (Chow 1995). The method of successive approximations is very useful for obtaining accurate solutions to oscillatory systems. In this paper, we apply the method of successive approximations to the damped pendulum. By doing so, a very accurate analytic approximate solution for the system can be derived from Newton's second law. Throughout this process, three different but similar solutions are found by adjusting the initial forms of the solutions and the initial conditions. The mathematics for both initial guesses are identical until the initial conditions are to be considered. For this reason, the following derivation will be done for the more accurate solution until the initial conditions, at which point, they will all be dealt with separately. We start by drawing a force diagram to visualize the system. Figure 1 shows the forces acting on a mass at the end of a pendulum. Here, T is the tension on the string, m is the mass, g is the acceleration due to gravity, and cv is the damping force component that is proportional to the velocity in the x direction with c being the coefficient of friction. The string, of length L, is assumed not to stretch in the work that follows.

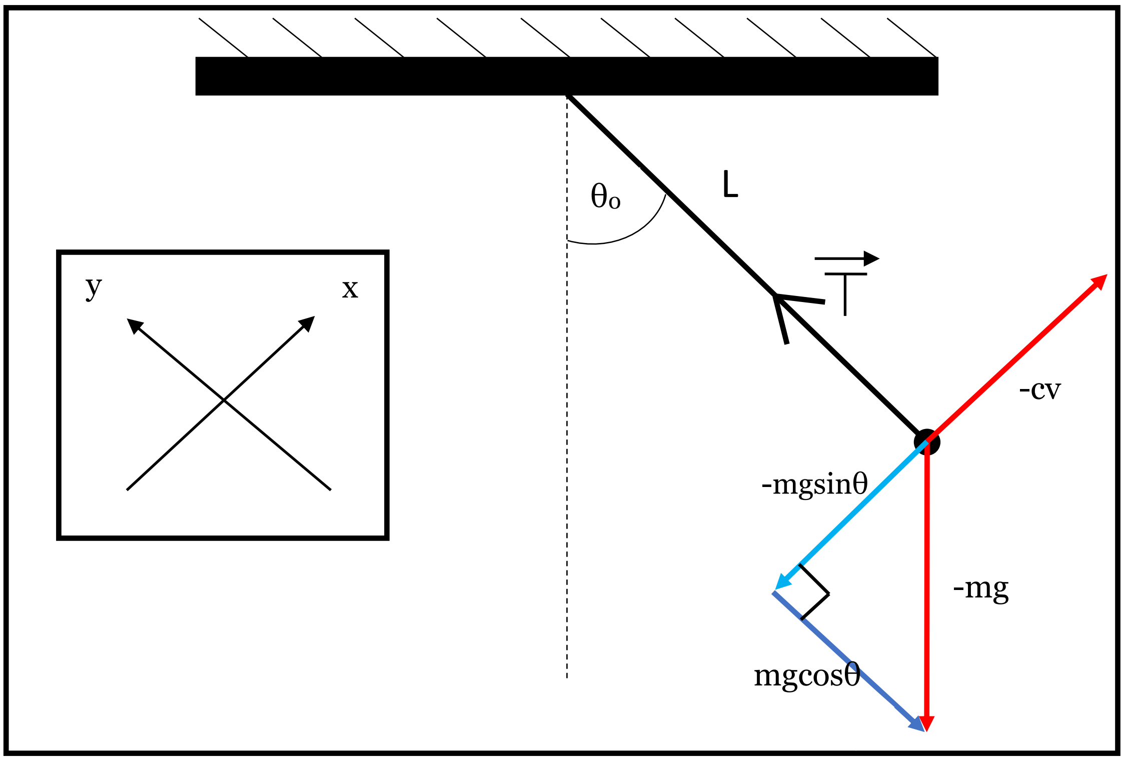

Figure 1. The different forces acting on a mass at the end of a pendulum. The orientation of the coordinate grid used to describe the forces is also shown.

From Figure 1, Newton's second law can be written as a second order differential equation.

$$\vec{F}_{Net} = \sum \vec{F}_{ext} = m\vec{a} \tag{1}$$

Since there is no stretching, the y-component of the gravitational force will cancel the tension force, thus there will be no motion in the y-direction, and Equation (1) becomes

$$m\frac{d^2x}{dt^2} = -c\frac{dx}{dt} - mg\sin(\theta), \tag{2}$$

or

$$m\frac{d^2x}{dt^2} + c\,\frac{dx}{dt} + mg\sin(\theta) = 0. \tag{3}$$

To write this equation in terms of the angle θ, let

$$x = L\theta \tag{4}$$

and substitute it into Equation (3) to get

$$\frac{d^2\theta}{dt^2} + \frac{c}{m}\frac{d\theta}{dt} + \omega_0^2\sin(\theta) = 0, \tag{5}$$

where $\omega_0 = \sqrt{\frac{g}{L}}$. Equation (5) is the general form of the differential equation for a damped pendulum. There is no exact analytic solution that we are aware of. However, when c = 0, Boas (2006) shows an approximate solution for its period. As mentioned above, in this work we obtain an approximate solution to this equation. Notice this equation includes a sin(θ) term, and a Taylor-series expansion to third order gives

$$\sin(\theta) \approx \theta - \frac{\theta^3}{6}, \tag{6}$$

which when substituted into Equation 5 gives the approximation (Hasbun 2009),

$$\frac{d^2\theta}{dt^2} + \frac{c}{m}\frac{d\theta}{dt} + \omega_0^2\theta \approx \omega_0^2\frac{\theta^3}{6}. \tag{7}$$

Before moving ahead, it is important to discuss briefly what is known about Equation (7). If, for example, we ignore the third order term on the right of Equation (7), we get

$$\frac{d^2\theta}{dt^2} + \frac{c}{m}\frac{d\theta}{dt} + \omega_0^2\theta = 0. \tag{8}$$

This equation is that of the damped simple harmonic oscillator for small oscillations and a possible analytic solution is of the form

$$\theta_{dSHO} = e^{-\gamma t}\theta_0 cos(\omega t), \tag{9}$$

where $\omega = \sqrt{\omega_0^2 - \gamma^2}$, $\gamma = \frac{c}{2m}$, and $\omega_0 = \sqrt{\frac{g}{L}}$ (Marion, 1988). In this solution, we have assumed that at t = 0, $\theta_{dSHO} = \theta_0$ and by default this assumes that $\dot{\theta}_{dSHO} = -\gamma\theta_0$. However, we are interested in going beyond Equation (8) and obtaining an approximate solution to Equation (7), which, of course, is an approximate solution to Equation (5). With this in mind, we use Equation (9) as a guide and assume two possible initial guesses,

$$\theta(t) = e^{-Bt}(A_1\cos(\omega_1 t) + A_2\cos(\omega_2 t)), \tag{10}$$

and

$$\theta(t) = e^{-Bt}(A_1\cos(\omega_1 t + \delta) + A_2\cos(\omega_2 t + \delta)). \tag{11}$$

Equation (11) contains six unknowns that will need to be solved for: B, A_1, ω_1, A_2, ω_2, and δ. Later, there will be two more unknowns included in this list. To determine these unknowns, the initial guess must be substituted into the differential equation. Presently, we derive the solution involving Equation 11 because it is the more accurate solution due to including the δ term, where the δ represents the phase shift of the system and allows us to have more freedom as regards the initial boundary conditions for $\theta(0)$ and $\dot{\theta}(0)$. The first and second derivative are found to be the following

$$\frac{d\theta}{dt} = -\mathrm{B}e^{-Bt}\mathrm{A}_1\cos(\omega_1 t+\delta) - \mathrm{A}_1\omega_1 e^{-Bt}\sin(\omega_1 t+\delta) - \mathrm{B}e^{-Bt}\mathrm{A}_2\cos(\omega_2 t+\delta)$$

$$-\mathrm{A}_2\omega_2 e^{-Bt}\sin(\omega_2 t+\delta), \tag{12}$$

$$\frac{d^2\theta}{dt^2} = e^{-Bt}\cos(\omega_1 t+\delta)[\mathrm{A}_1\mathrm{B}^2 - \mathrm{A}_1\omega_1^2] + e^{-Bt}\sin(\omega_1 t+\delta)[2\mathrm{A}_1\mathrm{B}\omega_1] +$$

$$e^{-Bt}\cos(\omega_2 t+\delta)\left[\mathrm{A}_2\mathrm{B}^2 - \mathrm{A}_2\omega_2^2\right] + e^{-Bt}\sin(\omega_2 t+\delta)[2\mathrm{A}_2\mathrm{B}\omega_2]. \tag{13}$$

It is also necessary to cube Equation 11 so that it can be substituted into the right-hand side of Equation (7). When doing this, it is important to neglect all cross terms. We get the following

$$(\theta(t))^3 \approx e^{-3Bt}(\mathrm{A}_1^3\cos^3(\omega_1 t+\delta) + \mathrm{A}_2^3\cos^3(\omega_2 t+\delta)). \tag{14}$$

Now Equations (12), (13), and (14) can be substituted into Equation (7). Once that is done and all the same terms containing the time variable have been grouped together, we get

$$\cos(\omega_1 t+\delta)\left[\mathrm{A}_1\mathrm{B}^2 - \mathrm{A}_1\omega_1^2 - \mathrm{A}_1\frac{cB}{m} - \mathrm{A}_1\omega_0^2\right] + \sin(\omega_1 t+\delta)\left[2\mathrm{A}_1\mathrm{B}\omega_1 - \mathrm{A}_1\omega_1\frac{c}{m}\right] +$$

$$\cos(\omega_2 t+\delta)\left[\mathrm{A}_2\mathrm{B}^2 - \mathrm{A}_2\omega_2^2 - \mathrm{A}_2\frac{cB}{m} - \mathrm{A}_2\omega_0^2\right] + \sin(\omega_2 t+\delta)\left[2\mathrm{A}_2\mathrm{B}\omega_2 - \mathrm{A}_2\omega_2\frac{c}{m}\right] \approx$$

$$\mathrm{A}_1^3\omega_0^2\frac{1}{6}e^{-2Bt}\cos^3(\omega_1 t+\delta) + \mathrm{A}_2^3\omega_0^2\frac{1}{6}e^{-2Bt}\cos^3(\omega 2t + \delta). \tag{15}$$

By assuming small damping or small B, the remaining exponential term can be approximated as

$$e^{-2Bt} \approx 1 - 2\mathrm{B}t. \tag{16}$$

To simplify the cubed cosine terms the following identity is used

$$\cos^3(\omega_1 t+\delta) = \frac{1}{4}(3\cos(\omega_1 t+\delta) + \cos(3(\omega_1 t+\delta))). \tag{17}$$

Substituting Equation (17) into the right-hand side of Equation (15) gives

$$\mathrm{A}_1^3\omega_0^2\frac{1}{6}e^{-2Bt}\left(\frac{1}{4}(3\cos(\omega_1 t+\delta) + \cos(3(\omega_1 t+\delta)))\right) + \mathrm{A}_2^3\omega_0^2\frac{1}{6}e^{-2Bt}\left(\frac{1}{4}(3\cos(\omega_2 t+\delta) + \cos(3(\omega_2 t+\delta)))\right)$$

$$. \tag{18}$$

The idea here is that it is convenient to disregard harmonics greater than the first. Thus, not forgetting Equation (16), Equation (18) can be simplified to become

$$A_1^3\omega_0^2\frac{1}{8}\cos(\omega_1 t+\delta)-A_1^3\omega_0^2\frac{B}{4}t\cos(\omega_1 t+\delta)+A_2^3\omega_0^2\frac{1}{8}\cos(\omega_2 t+\delta)-A_2^3\omega_0^2\frac{B}{4}t\cos(\omega_2 t+\delta) \tag{19}$$

which can be substituted back into the right-hand side of Equation (15) to get

$$\begin{aligned}&\cos(\omega_1 t+\delta)\left[A_1B^2-A_1\omega_1^2-A_1\frac{cB}{m}-A_1\omega_0^2\right]+\sin(\omega_1 t+\delta)\left[2A_1B\omega_1-A_1\omega_1\frac{c}{m}\right]+\\&\cos(\omega_2 t+\delta)\left[A_2B^2-A_2\omega_2^2-A_2\frac{cB}{m}-A_2\omega_0^2\right]+\sin(\omega_2 t+\delta)\left[2A_2B\omega_2-A_2\omega_2\frac{c}{m}\right]\approx\\&A_1^3\omega_0^2\frac{1}{8}\cos(\omega_1 t+\delta)-A_1^3\omega_0^2\frac{B}{4}t\cos(\omega_1 t+\delta)+A_2^3\omega_0^2\frac{1}{8}\cos(\omega_2 t+\delta)-A_2^3\omega_0^2\frac{B}{4}t\cos(\omega_2 t+\delta).\end{aligned} \tag{20}$$

Regrouping the terms, we get

$$\begin{aligned}&\cos(\omega_1 t+\delta)\left[A_1B^2-A_1\omega_1^2-A_1\frac{cB}{m}-A_1\omega_0^2-\frac{1}{8}A_1^3\omega_0^2\right]+\sin(\omega_1 t+\delta)\left[2A_1B\omega_1-A_1\omega_1\frac{c}{m}\right]+\\&\cos(\omega_2 t+\delta)\left[A_2B^2-A_2\omega_2^2-A_2\frac{cB}{m}-A_2\omega_0^2-\frac{1}{8}A_2^3\omega_0^2\right]+\sin(\omega_2 t+\delta)\left[2A_2B\omega_2-A_2\omega_2\frac{c}{m}\right]+\\&t\cos(\omega_1 t+\delta)\left[A_1^3\omega_0^2\frac{B}{4}\right]+t\cos(\omega_1 t+\delta)\left[A_2^3\omega_0^2\frac{B}{4}\right]=0.\end{aligned} \tag{21}$$

For this equation to hold, the coefficients of the terms containing the variable for time must vanish, to obtain

$$A_1B^2-A_1\omega_1^2-A_1\frac{cB}{m}-A_1\omega_0^2-\frac{1}{8}A_1^3\omega_0^2=0, \tag{22}$$

$$2A_1B\omega_1-A_1\omega_1\frac{c}{m}=0, \tag{23}$$

$$A_2B^2-A_2\omega_2^2-A_2\frac{cB}{m}-A_2\omega_0^2-\frac{1}{8}A_2^3\omega_0^2=0. \tag{24}$$

Some of the coefficients are trivially zero or are repeated expressions and are ignored. From Equation (23), we can determine B,

$$B=\frac{c}{2m}. \tag{25}$$

From Equation (22), ω_1 can be solved for in terms of ω_0, A_1, and B. Using Equation (24) and following the same algebra, ω_2 can be solved for as well to obtain

$$\omega_1=\sqrt{\omega_0^2\left(1-\frac{A_1^2}{8}\right)-B^2}, \tag{26}$$

$$\omega_2=\sqrt{\omega_0^2\left(1-\frac{A_2^2}{8}\right)-B^2}. \tag{27}$$

It turns out that all the above steps are similar if δ had been neglected, but this is the point where the δ, as well as the initial conditions, will make a difference. Each of the initial guesses will be treated separately to obtain the final solutions.

INITIAL CONDITIONS

The Method of Successive Approximations (MSA)

Recalling the solution without the delta, Equation (10),

$$\theta(t) = e^{-Bt}\left(A_1\cos(\omega_1 t) + A_2\cos(\omega_2 t)\right), \tag{28}$$

where Equations (25), (26), and (27) still hold, and A_1 and A_2 have still yet to be defined, we let

$$A_1 = A_2 = \frac{\theta_0}{2}, \tag{29}$$

so that

$$A_1 + A_2 = \theta_0. \tag{30}$$

This, along with Equations (25–27), for B, ω_1, and ω_2, allow us to obtain the first approximate solution, which we call the MSA.

The Modified Method of Successive Approximations (MMSA)

As can be seen in Equations (29) and (30), A_1 and A_2 are fixed at a set value. In our next approximation, we modify the definitions of A_1 and A_2. Rather than locking in their values, we introduced a new variable, x. The general form of the solution is the same as for the MSA, Equation (28), but we let

$$A_1 = x_1\theta_0, \tag{31}$$

and

$$A_2 = x_2\theta_0, \tag{32}$$

such that

$$x_1 + x_2 = 1. \tag{33}$$

Here the values of A_1 and A_2 can be varied based on the chosen values of x_1 and x_2. It is done this way in order to find the values of A_1 and A_2 that give the most accurate approximation, based on the error formula defined later below. We refer to this approximation as the MMSA.

The Improved Modified Method of Successive Approximations (IMMSA)

Introduced earlier in Equation (11), the final solution involves the additional parameter, δ. As mentioned before, the δ represents the phase shift of the system, but it also has another important use. When using the method of successive approximations, there is a restriction imposed on the initial value of the velocity. To improve on that restriction, the phase shift is included in the initial guess as well as redefining A_1 and A_2 once more. Recall the form of the solution for this method, Equation (11),

$$\theta(t) = e^{-Bt}(A_1\cos(\omega_1 t + \delta) + A_2\cos(\omega_2 t + \delta)). \tag{34}$$

In this approximation, we let

$$A_1 = yx\theta_0, \tag{35}$$

$$A_2 = y(1 - x)\theta_0, \tag{36}$$

such that

$$A_1 + A_2 = y\theta_0. \tag{37}$$

Notice that now, the unknown, y, must also be solved for in addition to the parameter δ. Applying the initial conditions,

$$\theta(0) = \theta_0 \text{ and } \dot{\theta}(0) = \dot{\theta}_0,$$

which, with the help of Equation (34), gives

$$\theta_0 = (A_1 + A_2)\cos(\delta), \tag{38}$$

or,

$$\cos(\delta) = \frac{\theta_0}{(A_1 + A_2)}. \tag{39}$$

The second initial condition gives

$$\dot{\theta}_0 = -B(A_1 + A_2)\cos(\delta) - (\omega_1 A_1 + \omega_2 A_2)\sin(\delta). \tag{40}$$

Substituting Equation (39) into the above equation gives

$$\dot{\theta}_0 = -B\theta_0 - (\omega_1 A_1 + \omega_2 A_2)\sin(\delta), \tag{41}$$

from which we find that,

$$\sin(\delta) = -\frac{\dot{\theta}_0 + B\theta_0}{(\omega_1 A_1 + \omega_2 A_2)}. \tag{42}$$

Equations (39) and (42) can be combined to obtain

$$\delta = \tan^{-1}\left\{-\frac{(A_1 + A_2)(\theta_0' + B\theta_0)}{\theta_0(\omega_1 A_1 + \omega_2 A_2)}\right\}. \tag{43}$$

Now that we have obtained δ, the last unknown that needs to be solved for is y. This is done by using the trigonometric identity,

$$\cos^2(\delta) + \sin^2(\delta) = 1, \tag{44}$$

so that using Equations (39) and (42) with the above equation gives,

$$1 = \left(\frac{\theta_0}{A_1 + A_2}\right)^2 + \left(-\frac{\theta_0' + B\theta_0}{\omega_1 A_1 + \omega_2 A_2}\right)^2, \tag{45}$$

which with the values of A_1, Equation (35), and A_2, Equation (36), give the following,

$$1 = (\frac{\theta_0}{y\theta_0})^2 + (-\frac{\theta_0' + B\theta_0}{yx\theta_0\omega_1 + y(1-x)\theta_0\omega_2})^2 \quad (46)$$

which can be simplified to solve for y to obtain

$$y = \sqrt{1 + \left(\frac{\theta_0' + B\theta_0}{\theta_0(\omega_1 x + \omega_2(1-x))}\right)^2}. \quad (47)$$

This is the final unknown that needs to be solved for, with the help of Equations (25), (26), and (27). Because this equation involves ω_1 and ω_2, which, in turn, involve A_1 and A_2, which depend on y, then it must be solved numerically in a self-consistent way depending on the value of x chosen that makes the approximations most accurate. This approximation is referred to as the IMMSA. At this point, all three of the theoretical approximate solutions are ready to be computed. One of last steps in this process is to come up with the numerical solution to Equation (5), against which the above three approximate solutions (MSA, MMSA, and IMMSA) will be compared.

The Euler-Cromer Method

To determine the accuracy of the above approximate solutions, we need a full numerical solution to compare them to. This numerical solution is the Euler-Cromer solution (Cromer 1981). The Euler-Cromer method is a simple way to numerically obtain the motion of a system for all time, given the initial values of the force, position, velocity, and acceleration. This method turns the acceleration into two first order derivatives,

$$a = \frac{d\theta}{dt} \text{ and } \dot{\theta} = \frac{d\theta}{dt}. \quad (48)$$

However, the two derivatives are expressed as forward difference equations, that is,

$$a \approx \frac{(\theta_{i+1} - \theta_i)}{\Delta t} \text{ and } \dot{\theta} \approx \frac{(\theta_{i+1} - \theta_i)}{\Delta t}.$$

Here, the values with "i+1" are the values one time-step, Δt, after the values with "i". The above equations allow us to solve for those "i+1" values, or

$$\dot{\theta}_{i+1} = \dot{\theta}_i + a_i \Delta t, \quad (49)$$

and

$$\theta_{i+1} = \theta_i + \dot{\theta}_{i+1} \Delta t, \quad (50)$$

where $\dot{\theta}_{i+1}$ is used according to the Euler-Cromer rule (Cromer 1981). We also know that the force is given by

$$\mathbf{F} = -cL\frac{d\theta}{dt} - mg\sin(\theta), \quad (51)$$

which can be rewritten, using the above relationships, as

$$F_{i+1} = -cL\dot{\theta}_{i+1} - mg\sin(\theta_{i+1}). \quad (52)$$

Newton's second law tells us

$$a = \frac{F}{m}, \tag{53}$$

which is rewritten as

$$a_{i+1} = \frac{F_{i+1}}{m}. \tag{54}$$

The time variable starts from its initial value and increases by Δt at each step as

$$t_{i+1} = t_i + \Delta t. \tag{55}$$

Equations (48–55) are incorporated into a program that can determine the system's motion numerically, given the initial conditions. This full numerical solution to the damped pendulum will be used to determine how accurate the approximate solutions (MSA, MMSA, and IMMSA) are. A measure of accuracy in each approximation is determined according to

$$\text{Error} = \sqrt{\frac{1}{N}\sum(y_{th\,i} - y_{e\,i})^2}, \tag{56}$$

where y_{thi} refers to an approximate solution and y_{ei} is the numerical Euler-Cromer solution at time t_i.

SIMULATION AND RESULTS

Program 1 – Successive_Approx_MSA_MMSA.m

Here, we use MATLAB (Mathworks) to perform the computations. In the first program, the two simplest methods, the MSA and the MMSA, are solved and plotted along with the numerical Euler-Cromer solution and MATLAB's ODE solver. Figure 2 shows the four different solutions plotted together. The values of the constants and the ones we chose for the initial conditions can be found in the program's code in appendix A.

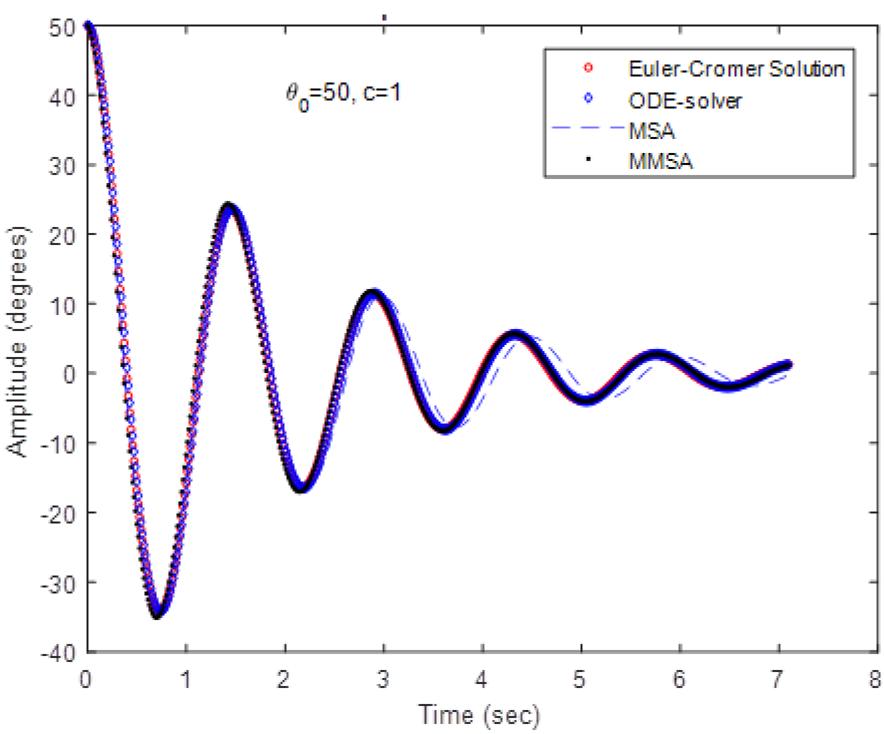

Figure 2. Four different solutions to Equation (5), two numerical (Euler-Cromer and ODE solver) and two approximations (MSA and MMSA). The initial displacement of the pendulum is θ = 50° and the damping coefficient, c = 1.

In this program, after choosing the initial conditions, we needed to determine the best value of x_1, between zero and one, that produced the lowest error. This was done by trial and error and it was found that the value $x_1 = 0.6$ gave the lowest error. The errors, Equation (56), computed by the program came out to be Error1 (MSA) = 0.0408 and Error2 (MMSA) = 0.0313. This tells us that the method of successive approximations was less accurate than the modified method of successive approximations. Note also that the reason we included MATLAB's ODE solver here is that we needed to show that the Euler-Cromer method is accurate enough in the present problem that we do not have to perform numerical computations with more sophisticated approaches.

Program 1 – Successive_Approx_MMSA_IMMSA.m

The second program no longer includes MATLAB's ODE solver or the simplest MSA solution. Like the first program, this one again plots the numerical Euler-Cromer solution and the MMSA, but now also includes the improved modified method of successive approximations (IMMSA). Looking back at the final solution to the IMMSA, notice there is a cycle of dependencies between the variables. Equations (35) and (36), for A_1 and A_2, are dependent on the value of y. But Equation (47), for y, depends on the value of ω_1 and ω_2. The problem is that the values of ω_1 and ω_2 are dependent on the values of A_1 and A_2. This creates a loop that must be taken care of computationally. For this reason, a function is used in the code that determines the zero of the y-function (LHS of [47] – RHS of [47]) based on the different values of ω_1 and ω_2 and all the possible values of y. The results are shown in Figure 3.

The IMMSA and the MMSA depend on the value of $x = x_1$ (with $x_2 = 1 - x_1$) that gives the lowest error and the value found is $x_1 = 0.6$. It can be seen in the graph that the solutions are extremely close, but the actual error values obtained are the following, Error1 (MMSA) = 0.0313 and Error2 (IMMSA) = 0.0099. This shows how much closer the IMMSA is to the full numerical Euler-Cromer solution to Equation (5) than is the MMSA.

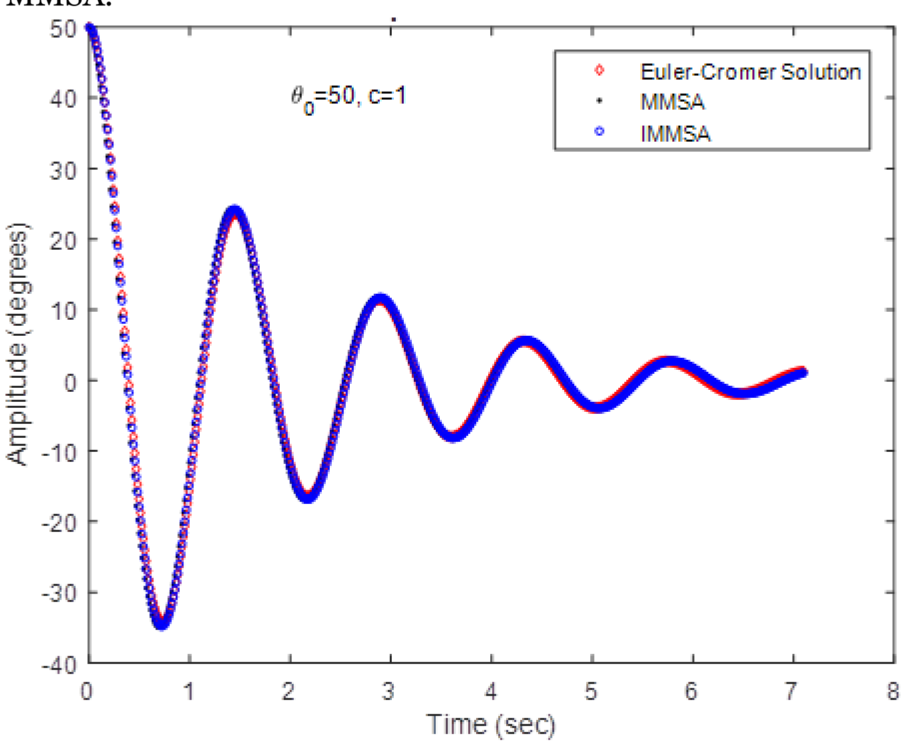

Figure 3. Results produced by the second program. The three solutions (numerical: Euler-Cromer, approximate: MMSA and IMMSA) are shown, as well as the initial displacement of the pendulum, θ = 50°, and the value of the damping coefficient, c = 1.

The IMMSA Compared to Johannessen's Approximation

In a previous work (Johannessen 2014) the author developed an approximation for the nonlinear damped pendulum given in the form

$$\psi(\mathrm{u}) = 2\arctan\left(\frac{\sqrt{m(u)}\,sn(\xi(u),m(u))}{dn(\xi(u),m(u))}\right), \tag{57}$$

where $\theta(t) = \psi(\mathrm{u})$ with $\mathrm{u} = \omega_0 t$, $m(u) = m_0 exp(-2\gamma u)$, $m_0 = \sin\left(\frac{\psi_{max}}{2}\right)^2$; including initial conditions: $\psi(0) = 0$, $\psi'(0) = 2\sin\left(\frac{\psi_{max}}{2}\right)$, and where sn, dn are Jacobi elliptic functions. In Equation (57) the function $\xi(u)$ is given by

$$\xi(u) = \left(1 + \frac{1}{4}m(u) + \frac{9}{64}m(u)^2\right)u + \frac{1}{8\gamma}\left(m(u) - m_0\right) + \frac{1}{8\gamma}\left(m(u)^2 - {m_0}^2\right) \tag{58}$$

with ψ_{max} being the condition of maximum amplitude. We can compare the above result of Equation (57) with our best approximation, the IMMSA of Equation (34) along with Equations (25–27, 35, 36, 43, and 47), if we let $\theta(0) = 0$, $\theta'(0) = 2\omega_0 \sin\left(\frac{\psi_{max}}{2}\right)$ with $\theta_{max} = \psi_{max} = 2\sin^{-1}\left(\frac{\theta\prime(0)}{2\omega_0}\right)$ in our equations as shown in Figure 4. We also rewrite Equation (34) as

$$\theta(t) = e^{-B|t-t_0|}\left(A_1\cos(\omega_1(t-t_0)+\delta) + A_2\cos(\omega_2(t-t_0)+\delta)\right). \tag{59}$$

To effect the comparison, we solve for t_0, such that $A_1\cos(\omega_1(t-t_0)+\delta) + A_2\cos(\omega_2(t-t_0)+\delta) = 0$ at t = 0 as Equation (57) also requires (Johannessen 2014). This comparison is carried out in Figure 4 with the parameters previously employed (Johannessen 2014). We also perform an error comparison of each approximation against the MATLAB solver, as in Equation (56), rather than the Euler-Cromer method. For our IMMSA, the error is 0.2617 and for Equation (57) the error is 0.5397, indicating that our approximation does very well in this regime. Appendix C contains the script used to carry out the comparison shown in Figure 4.

DISCUSSION

The purpose behind this paper is to find a theoretical approximate solution to the damped pendulum using the method of successive approximations. While this has been achieved using the simplest solution, we wanted to find solutions that are more accurate than our first try. By including the phase shift and redefining the amplitudes A_1 and A_2 in the method of successive approximations, the error produced is much smaller. The most accurate approximation is that of the improved modified method of successive approximations, (IMMSA), which had an error of 0.0099. This is a much better error than the MSA, which had an error of 0.0408, and the MMSA, which had an error of 0.0313. Finally, the IMMSA has been compared to the Johannessen's approximation (Johannessen 2014) with very good results. We think that the IMMSA is a significantly important result because it leads to an advancement in the method of successive approximations as well as providing insight into the ways that different approximations are used and how they can affect the outcome.

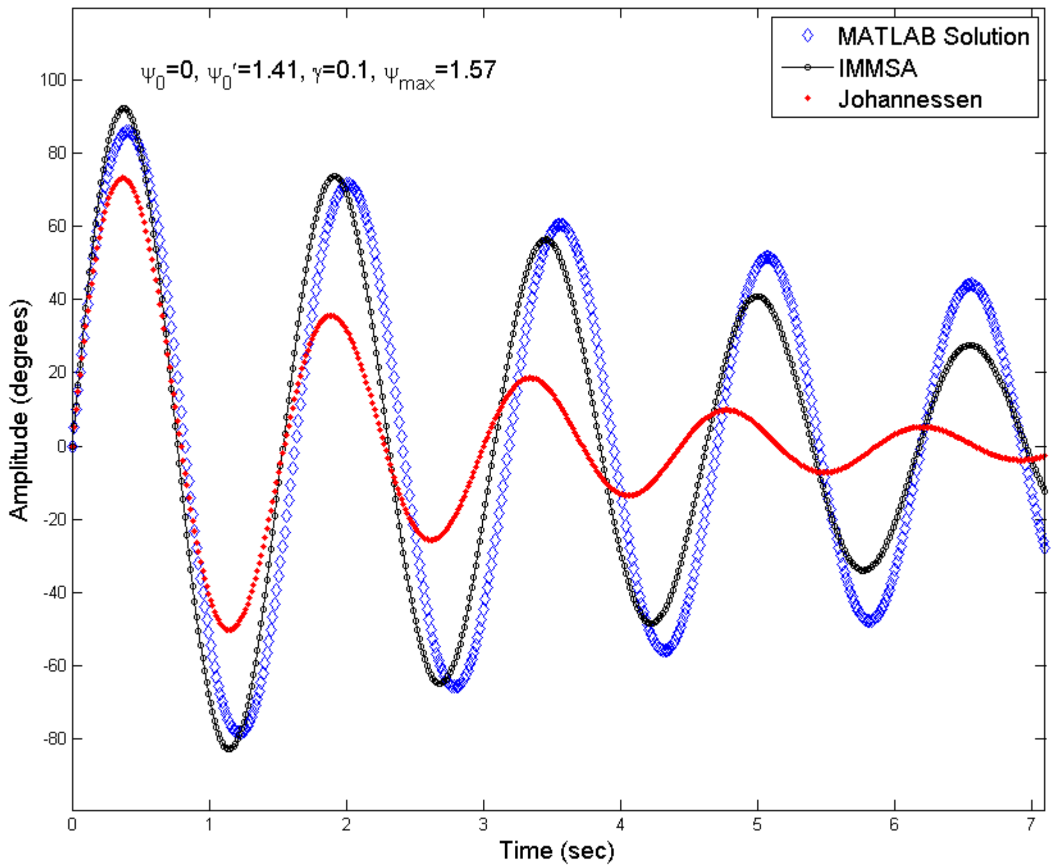

Figure 4. Results of the script of Appendic C. Data were generated by the IMMSA, Equations (25–27, 35, 36, 43, and 47), Equations (57–58) of Johannessen's approximation, and MATLAB's numerical solution. The parameters used here are as follows: m = 1, γ = 0.1, c = 2mγ, $\psi'(0) = 2\sin(\psi_{max}/2)$, $\psi_{max} = \pi/2$, $\psi(0) = 0$ (Johannessen 2014).

ACKNOWLEDGEMENTS

This work was supported by a grant from SEEP (STEM Education Improvement Plan)

REFERENCES

Boas, M.L. 2006. Mathematical Methods in the Physical Sciences. John Wiley & Sons, p. 557.

Chow, T.L. 1995. Classical Mechanics. New York: John Wiley, pp. 314–315

Cromer, A. 1981. Stable solutions using the Euler approximation. American Journal of Physics, 49, 455.

Hasbun, J.E. 2009. Classical Mechanics with MATLAB Applications. Jones & Bartlett, p. 104.

Johannessen, K. 2014. An analytical solution to the equation of motion of the damped nonlinear pendulum. Eur. J. Phys., 35, 035014.

MathWorks (www.mathworks.com), owner of MATLAB. The paper code is also compatible with the open source Octave (https://www.gnu.org/software/octave/).

Marion, J.B. and S.T. Thornton. 1988. Classical Dynamics of Particles and Systems, 3rd ed. Harcourt Brace Jovanovich.

APPENDIX A

This is the code from the first program that was used to plot the two approximate theoretical solutions, the MSA and the MMSA, and the two full numerical solutions, the Euler-Cromer and MATLAB's ODE solver. The values of the constants and the values we chose for the initial conditions are as follows: m = 1.0; c = 1.0; t_0 = 0; g = 9.8; L = 0.5; θ_0 = 50°; θ_0' = 0; x_1 = 0.6.

---------------- Script Listing ---------------

```
% Successive_Approx_MSA_MMSA.m (7/2018) By Justin Hill and J. Hasbun
% This solves the full pendulum with damping numerically by The Euler method
% and the ODE solver as well as solving the approximate form through the
% method of successive approximations (MSA). The method of successive
% approximation is further improved to become the modified method of     %
successive approximations (MMSA).


function Successive_Approx_MSA_MMSA
global w0 m c
c=1;
m=1.0;
t0=0.0;
g=9.8;
L=.5;
B=c/(2*m);

cf=2*pi/360;          %conversion factor from degrees to radians
w0=sqrt(g/L);
tau0=2*pi/w0;         %period for the SHO
tmax=5*tau0;          %maximum time
th=50;                %initial amplitude in degrees
thd=0;                %initial speed in degrees
thr=th*cf;            %initial angle in radians
dtheta0=thd*cf;       %initial speed in radians
NPTS=500;
dt=tmax/(NPTS-1);
t=[0:dt:tmax];

          %The Method of Successive Approximations

omA=sqrt((w0^2)*(1-(thr^2)/8)-(B^2));
A1=thr;
A3=((w0^2)*(A1^3))/(24*(w0^2)-24*(B^2)-216*(omA^2));
thMSA=exp(-B.*t).*(thr*cos(omA.*t)+A3*cos(3*omA.*t));

        %The Modified Method of Successive Approximations

x1=0.6;
x2=1-x1;
A1=x1*thr;
A2=x2*thr;
om1=sqrt((w0^2)*(1-A1^2/8)-(B^2));
om2=sqrt((w0^2)*(1-A2^2/8)-(B^2));
thMMSA=exp(-B.*t).*(A1*cos(om1.*t)+A2*cos(om2.*t));

           %The Numerical Solution (Euler Method)
```

```
F0=-c*L*dtheta0-m*g*sin(thr);  %initial force
a0=F0/m/L;                     %initial acceleration
theta(1)=thr;
dtheta(1)=dtheta0;
t(1)=t0;
F(1)=F0;
a(1)=a0;

for i=1:NPTS-1

  dtheta(i+1)=dtheta(i)+a(i)*dt;                 %new theta
  theta(i+1)=theta(i)+dtheta(i+1)*dt;            %new theta dot
  F(i+1)=-c*L*dtheta(i+1)-m*g*sin(theta(i+1)); %new force
  a(i+1)=F(i+1)/m/L;                             %new acceleration

end;

Error1=sqrt(sum((thMSA(:)-theta(:)).^2)/NPTS)
Error2=sqrt(sum((thMMSA(:)-theta(:)).^2)/NPTS)



          %The Numerical Solution (MATLAB SOLVER)
ic1=[thr;dtheta0];
[tm,th2m]=ode45(@fderivs,[t0:dt:tmax],ic1);% matlab numerical solution

plot(t,theta/cf,'rd');
hold on
plot(tm,th2m(:,1)/cf,'bd'); %The MATLAB solver solution
plot(t,thMSA/cf,'b--');
plot(t,thMMSA/cf,'k.');
legend('Euler-Cromer Solution','ODE-solver','MSA','MMSA');
str=cat(2,'\theta_0=',num2str(th,3),', c=',num2str(c,3));
text(2,th*(1-0.2),str);
xlabel('Time (sec)');
ylabel('Amplitude (degrees)');
title('Comparison of Solutions');

function derivs = fderivs(t,z)
global w0 m c
% pend2_der: returns the derivatives for the pendulum's full solution
% The function pen2_der describes the equations of motion for a
% pendulum. The parameter w0, is part of the input
% Entries in the vector of dependent variables are:
% x(1)-position, x(2)-angular velocity
derivs = [z(2); -w0^2*sin(z(1))-c*z(2)/m]; %the damping case is included now
```

APPENDIX B

This is the code from the second program that plotted the numerical Euler-Cromer solution as well as the modified method of successive approximations and the improved modified method of successive approximations. Both were compared using the error formula written into the program. You can see below that a zero function was used to calculate the zero of the y-function. The constants and chosen initial values are as follows: m = 1.0; c = 1.0; t_0 = 0; g = 9.8; L = 0.5; $\theta_0 = 50°$; $\theta_0' = 0$; x_1 = 0.6.

---------------- Script Listing ---------------

```
% Successive_Approx_MMSA_IMMSA.m (7/2018) By Justin Hill and J. Hasbun
% This solves the full pendulum with damping numerically by The Euler method
% as well as solving the approximate form through the
% The modified method of successive approximations (MMSA) and further by
% the improved modified method of successive approximation (IMMSA).

function Successive_Approx_MMSA_IMMSA
global w0 m c
c=1;
m=1.0;
t0=0.0;
g=9.8;
L=.5;
B=c/(2*m);

cf=2*pi/360;             %conversion factor from degrees to radians
w0=sqrt(g/L);
tau0=2*pi/w0;            %period for the SHO
tmax=5*tau0;             %maximum time
th=50;                   %initial amplitude in degrees
thd=0;                   %initial theta_dot in degrees/sec
thr=th*cf;               %initial angle in radians
dtheta0=thd*cf;          %initial theta_dot in radians
NPTS=500;
dt=tmax/(NPTS-1);
t=[0:dt:tmax];

       %The Modified Method of Successive Approximations

x1=0.6;
x2=1-x1;
A1=x1*thr;
A2=x2*thr;
om1=sqrt((w0^2)*(1-A1^2/8)-(B^2));
om2=sqrt((w0^2)*(1-A2^2/8)-(B^2));
thMMSA=exp(-B.*t).*(A1*cos(om1.*t)+A2*cos(om2.*t));

     %The Improved Modified Method of Successive Approximations

y = fzero(@(y) y_iter(y,x1,thr,dtheta0),0.5)
A12=y*x1*thr;
A22=y*(1-x1)*thr;
om12=sqrt(w0^2*(1-A12^2/8)-B^2);
om22=sqrt(w0^2*(1-A22^2/8)-B^2);
del=atan(-(dtheta0+B*thr)*(A12+A22)/(thr*(om12*A12+om22*A22)));
thIMMSA=exp(-B*t).*(A12*cos(om12*t+del)+A22*cos(om22*t+del));

         %The Numerical Solution (Euler Method)

F0=-c*L*dtheta0-m*g*sin(thr);  %initial force
a0=F0/m/L;                     %initial acceleration
theta(1)=thr;
dtheta(1)=dtheta0;
t(1)=t0;
F(1)=F0;
a(1)=a0;
```

```
for i=1:NPTS-1

  dtheta(i+1)=dtheta(i)+a(i)*dt;                %new theta
  theta(i+1)=theta(i)+dtheta(i+1)*dt;           %new theta dot
  F(i+1)=-c*L*dtheta(i+1)-m*g*sin(theta(i+1)); %new force
  a(i+1)=F(i+1)/m/L;                            %new acceleration

end;

Error1=sqrt(sum((thMMSA(:)-theta(:)).^2)/NPTS)
Error2=sqrt(sum((thIMMSA(:)-theta(:)).^2)/NPTS)
plot(t,theta/cf,'rd','MarkerSize',3);
hold on
plot(t,thMMSA/cf,'k.');
plot(t,thIMMSA/cf,'bo','MarkerSize',3);
legend('Euler-Cromer Solution','MMSA','IMMSA');
str=cat(2,'\theta_0=',num2str(th,3),', c=',num2str(c,3));
text(2,th*(1-0.2),str);
xlabel('Time (sec)');
ylabel('Amplitude (degrees)');
title('Comparison of Solutions');

function fyzero=y_iter(y,x,thr,thrd)
global w0 m c
A1=y*x*thr;
A2=y*(1-x)*thr;
B=c/2/m;
om1=sqrt(w0^2*(1-A1^2/8)-B^2);
om2=sqrt(w0^2*(1-A2^2/8)-B^2);
fyzero=y-sqrt(1+((thrd+B*thr)/(om1*x+om2*(1-x))/thr)^2);
```

APPENDIX C

This is the code from the first program that was used to plot the two approximate theoretical solutions, the IMMSA and the Johannessen's (our Equations [57–58]). Both solutions are compared to the result of the MATLAB's ODE solver in Figure 4. The parameters used are as follows: m = 1.0, gamma = 0.1, c = 2*m*gamma, g = 9.8, L = 0.5, psi0 = 0.0, psi0' = 2*sin(psi_max/2), psi_max = pi/2, and x = 0.6.

---------------- Script Listing ---------------

```
% IMMSA_and_Johannessen.m by J. E. Hasbun (9/2018)
% This compares the IMMSA, the MATLAB solver, and Johannessen's
% solutions.
% This solves the full pendulum with damping numerically using a MATLAB
% solver as well as solving the approximate form through the
% by the improved modified method of successive approximation (IMMSA) which
% is compared to the work of Johannessen (Eur. J. Phys, V38, 035014
% (2014)).

function IMMSA_and_Johannessen
clear
global w0 m c

m=1.0;
```

```
t0=0.0;
g=9.8;
L=.5;
gam=0.1;              %as used by Johannessen
c=2*m*gam;
B=c/(2*m);
cf=2*pi/360;          %conversion factor from degrees to radians
w0=sqrt(g/L);
tau0=2*pi/w0;         %period for the SHO
tmax=5*tau0;          %maximum time

%Here are the conditions when psi_max and psi0 are provided
psi_max=pi/2;               %maximum angle needed - radians
psi0=0.0;                   %initial angle psi=thr - radians
psi0_p=2*sin(psi_max/2);    %initial psi prime=dtheta0/w0 - radians/sec
thr=psi0;                   %radians
dtheta0=w0*psi0_p;          %rad/sec
th=thr/cf;                  %theta_0 in degrees
NPTS=500;
dt=tmax/(NPTS-1);
t=[0:dt:tmax];

           %The IMMSA solution
x=0.6; %as used here

thr0=-1.4; %For the amplitude of the IMMSA, for this comparison
y = fzero(@(y) y_iter(y,x,thr0,dtheta0),1.0); %solve for y
A12=y*x*thr0;
A22=y*(1-x)*thr0;
om12=sqrt(w0^2*(1-A12^2/8)-B^2);
om22=sqrt(w0^2*(1-A22^2/8)-B^2);
del=atan(-(dtheta0+B*thr0)*(A12+A22)/(thr0*(om12*A12+om22*A22)));

%solve for t00 so that theta passes through zero at t=0 in this comparison
ff=@(tt) A12*cos(-om12*tt+del)+A22*cos(-om22*tt+del);
t00=fzero(@(tt) ff(tt),-1.5);
fprintf('thr0=%4.5f, y=%4.5f, t00=%4.5f\n',thr0,y,t00)
thIMMSA=exp(-B*abs(t-t00)).*(A12*cos(om12*(t-t00)+del)+A22*cos(om22*(t-
t00)+del));

           %The Numerical Solution (MATLAB SOLVER)
ic1=[thr;dtheta0];
[tm,th2m]=ode45(@fderivs,[t0:dt:tmax],ic1);% matlab numerical solution
Error_thIMMSA=sqrt(sum((thIMMSA(:)-th2m(:,1)).^2)/NPTS);

           %Johannessen's solution
sc1=w0/sqrt(1+gam^2);
u=sc1*t;
mu0=(sin(psi_max/2))^2;
mu=mu0*exp(-2*gam*u);
xi=(1+mu/4+9*mu.^2/64).*u+(mu-mu0)/gam/8+9*(mu.^2-mu0^2)/gam/256;
[sn,cn,dn]=ellipj(xi,mu);
thJohann=2*atan(sqrt(mu).*sn./dn);
Error_thJohann=sqrt(sum((thJohann(:)-th2m(:,1)).^2)/NPTS);

fprintf('Error_thIMMSA=%4.5f,
Error_thJohann=%4.5f\n',Error_thIMMSA,Error_thJohann)
```

```
plot(tm,th2m(:,1)/cf,'bd'); %The MATLAB solver solution
hold on
plot(t,thIMMSA/cf,'ko-','MarkerSize',3);
plot(t,thJohann/cf,'r.')

legend('MATLAB Solution','IMMSA','Johannessen');
str=cat(2,'\psi_0=',num2str(psi0,3),', \psi_0\prime=',num2str(psi0_p,3),...
  ', \gamma=',num2str(gam,3),', \psi_{max}=',num2str(psi_max,3));
text(0.5,max(thIMMSA/cf)*(1+0.1),str);
axis([0 tmax min(thIMMSA/cf)*(1+0.2) max(thIMMSA/cf)*(1+0.3)])
xlabel('Time (sec)');
ylabel('Amplitude (degrees)');
title('Comparison of Solutions');

function fyzero=y_iter(y,x,thr,thrd)
global w0 m c
A1=y*x*thr;
A2=y*(1-x)*thr;
B=c/2/m;
om1=sqrt(w0^2*(1-A1^2/8)-B^2);
om2=sqrt(w0^2*(1-A2^2/8)-B^2);
fyzero=y-sqrt(1+((thrd+B*thr)/(om1*x+om2*(1-x))/thr)^2);

function derivs = fderivs(t,z)
global w0 m c
% pend2_der: returns the derivatives for the pendulum's full solution
% The function pen2_der describes the equations of motion for a
% pendulum. The parameter w0, is part of the input
% Entries in the vector of dependent variables are:
% x(1)-position, x(2)-angular velocity
derivs = [z(2); -w0^2*sin(z(1))-c*z(2)/m]; %the damping case is included now
```