Tim Garvin
Matthew Norton
MAT 340

Approximation Methods: Damped and Undamped Oscillators

There are several methods for solving differential equations numerically. The Euler method was the first method used to solve numerically differential equations. However, the Euler method is not very precise and is prone to errors. In fact, when it is used to model damped and undamped oscillators, the Euler method shows the oscillator's position, velocity, and acceleration increasing, therefore implying that it is gaining energy, which is clearly impossible due to the first law of thermodynamics. There have been numerous attempts by individuals to improve upon the Euler method since it was created, and they have gotten to a point where with just a few minor tweaks to the Euler algorithm, one can get an extremely precise approximation of a differential equation. Figure 1 shows an example of an undamped and damped oscillator. For the undamped oscillator, we used a time step of 0.002 in the code, and a time step of 0.25 for the damped oscillator.

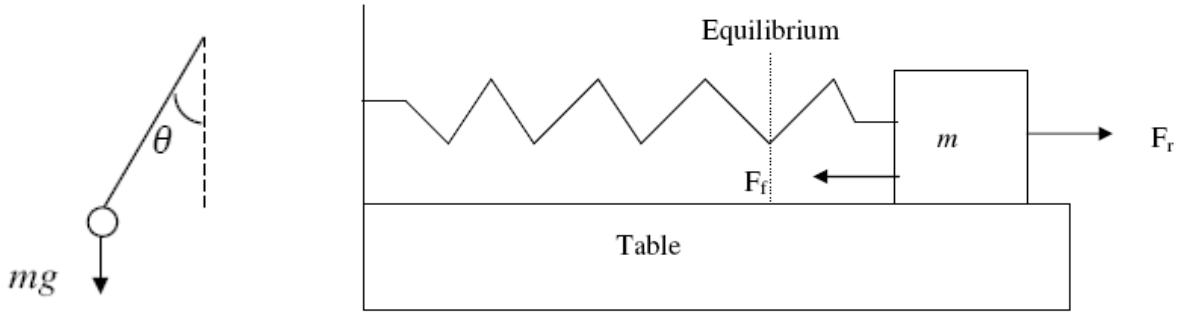

Fig. 1. An undamped oscillator (pendulum) and a damped oscillator (spring)

The equation for damped and undamped oscillators follows from Isaac Newton's second law of motion, F=ma.

$$\vec{F} = m\ddot{\vec{x}} = -c \cdot \dot{\vec{x}} - k \cdot \vec{x} \tag{1}$$

The Euler algorithm approximates x_{n+1} by adding the velocity multiplied by the time step to x_n,

$$x(t_0 + h) = x(t_0) + h \cdot v(t_0) \tag{2}$$

where *h* is the time step and *v(t)* the function for the velocity of the oscillator. The Euler algorithm is simple to program. Figure 2 shows the loop that one could use to implement the Euler method in a FORTRAN program.

```
DO I = 1, 250
A = -(C1 * Xold + C2 * Vold)
V = Vold + Dt * A
X = Xold + dt * Vold
TimeX = TimeX + Dt
WRITE (40, 1000) TimeX, X, V, A
Xold = X
Vold = V
END DO
```

Fig. 2. Sample FORTRAN code for using Euler method

Figure 3 depicts the position versus time for an initial angular displacement of 1º and an initial velocity of 0 rad/s that was obtained using the Euler method. Notice that extrema for the graph are not constant, they increase slightly as time increases. This is due to how the Euler algorithm is set up. The Euler algorithm uses an older value for velocity in calculating the position than what could be used. By not using the most recent value for velocity, the position was off and caused this noticeable error in the calculations of the Euler algorithm. Figure 4 shows the graph of position vs. time for the Euler method for various values for the time step. Notice how that as the time step gets smaller, the accuracy of the method increases. This increased the computational power that was required to find the solution. This increase is undesirable when there are other ways to find the solution to an undamped oscillator that do not require as much computational time to find and accurate solution.

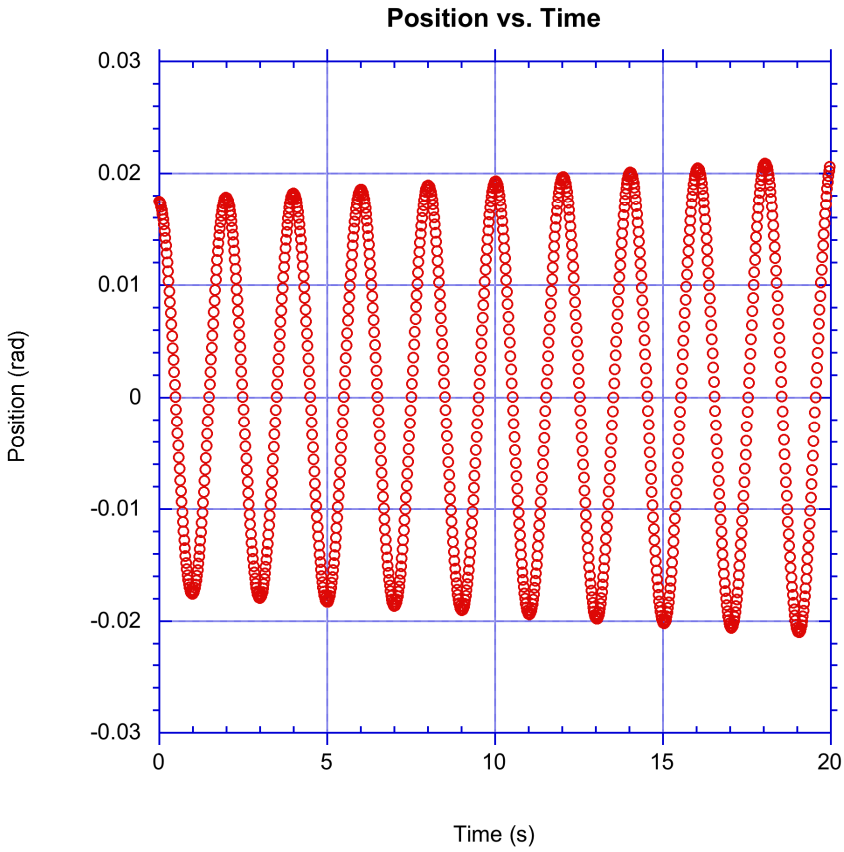

Fig. 3. Position vs. time for an undamped oscillator using Euler method

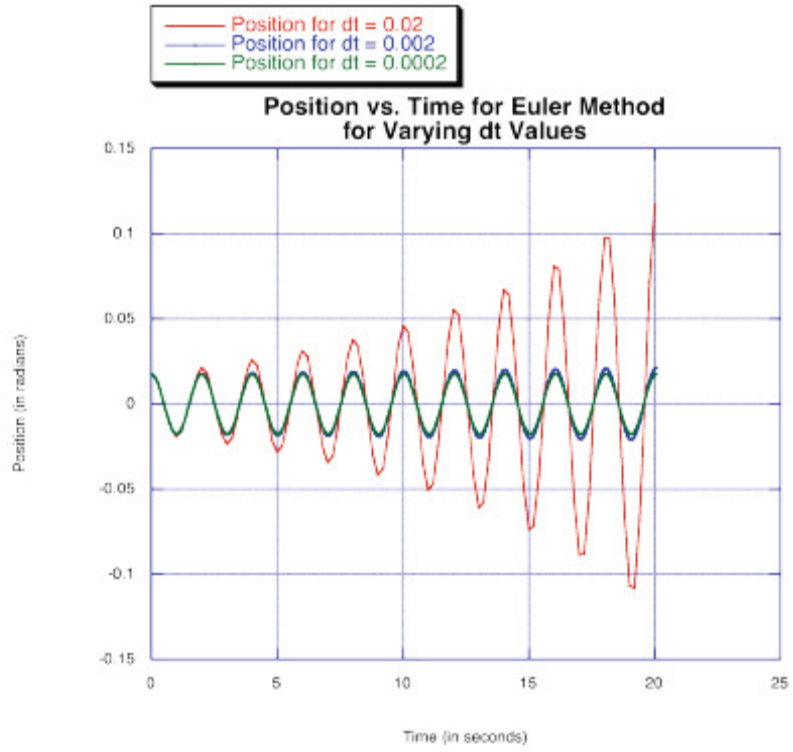

Fig. 4. Position vs. time for an undamped oscillator using Euler method with various time steps

When one considers a damped oscillator, the Euler method does even worse. Figure 5 depicts the position versus time for an initial displacement of 1m and an initial velocity of 0 m/s that was obtained using the Euler method. Notice that extrema for the graph are not constant, they increase greatly as time increases. In fact, this graph looks the opposite of how it should. Since the oscillator is damped, its amplitude should eventually decrease over time.

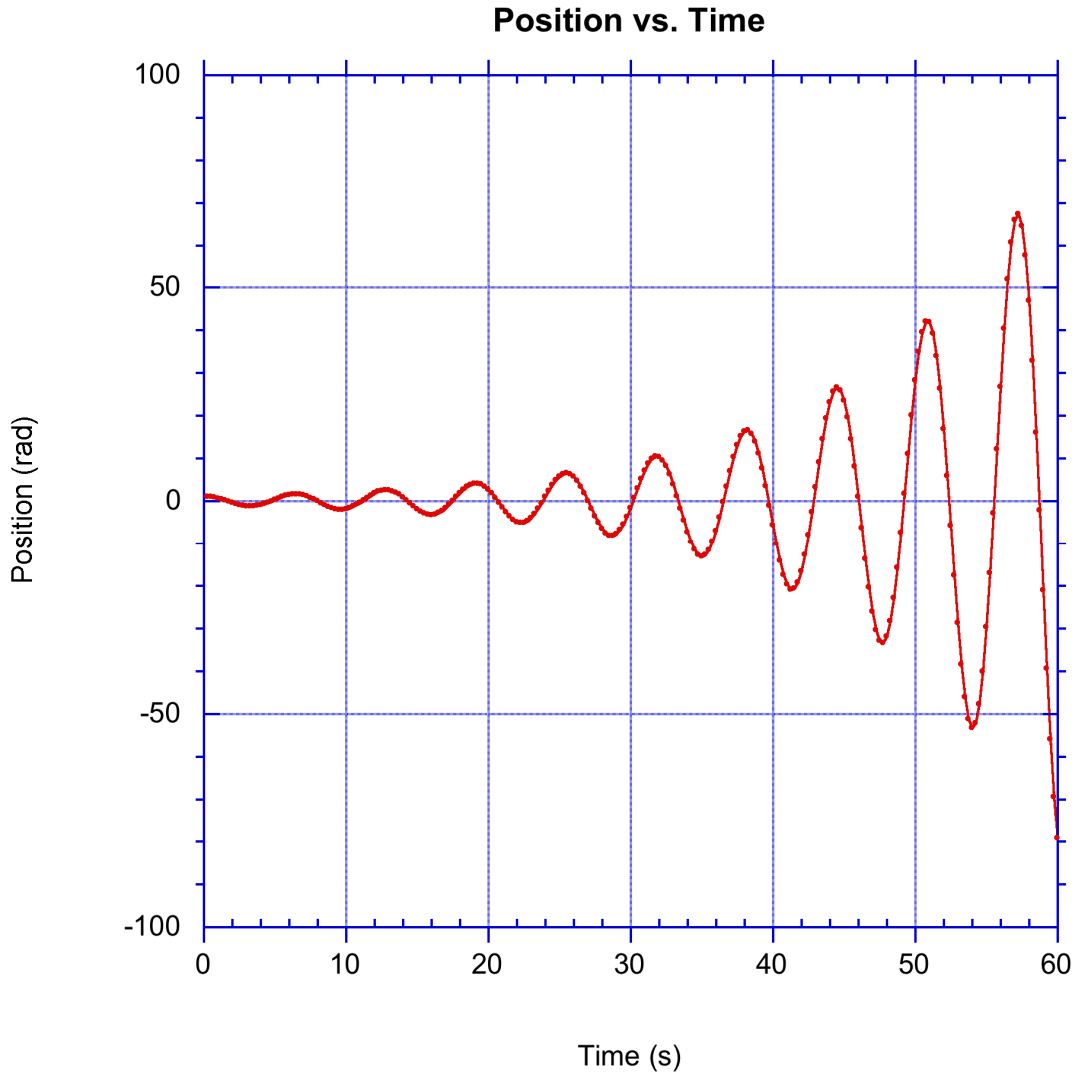

Fig. 5. Position vs. time for a damped oscillator using the Euler method

It can be clearly seen that there is a serious problem with the Euler method providing unstable results. The Euler-Cromer method for solving first order differential equations corrects this problem by always using the most recent value for the velocity when calculating the position. Therefore, the equation for the Euler-Cromer method is given by:

$$x(t_0 + h) = x(t_0) + h \cdot v(t_0 + h). \tag{3}$$

The Euler-Cromer algorithm is also simple to program. Figure 6 shows the loop that one could use to implement the Euler-Cromer method in a FORTRAN program.

```
DO I = 1, 250
A = -(C1 * Xold + C2 * Vold)
V = Vold + Dt * A
X = Xold + dt * V
TimeX = TimeX + Dt
WRITE (50, 1000) TimeX, X, V,
A
Xold = X
Vold = V
END DO
```

Fig. 6. Sample FORTRAN code for using Euler-Cromer method

Figure 7 depicts the position versus time for an initial angular displacement of 1º and an initial velocity of 0 rad/s that was obtained using the Euler-Cromer method. Notice how the graph is relatively stable and that the oscillator's amplitude does not increase as time increases. When one considers a damped oscillator, the Euler-Cromer method does much better than the Euler method. Figure 8 depicts the position versus time for an initial displacement of 1m and an initial velocity of 0 m/s that was obtained using the Euler-Cromer method. Notice that extrema for the graph decrease as time increases. This is how the graph should look because the damped oscillator's amplitude should eventually decrease over time.

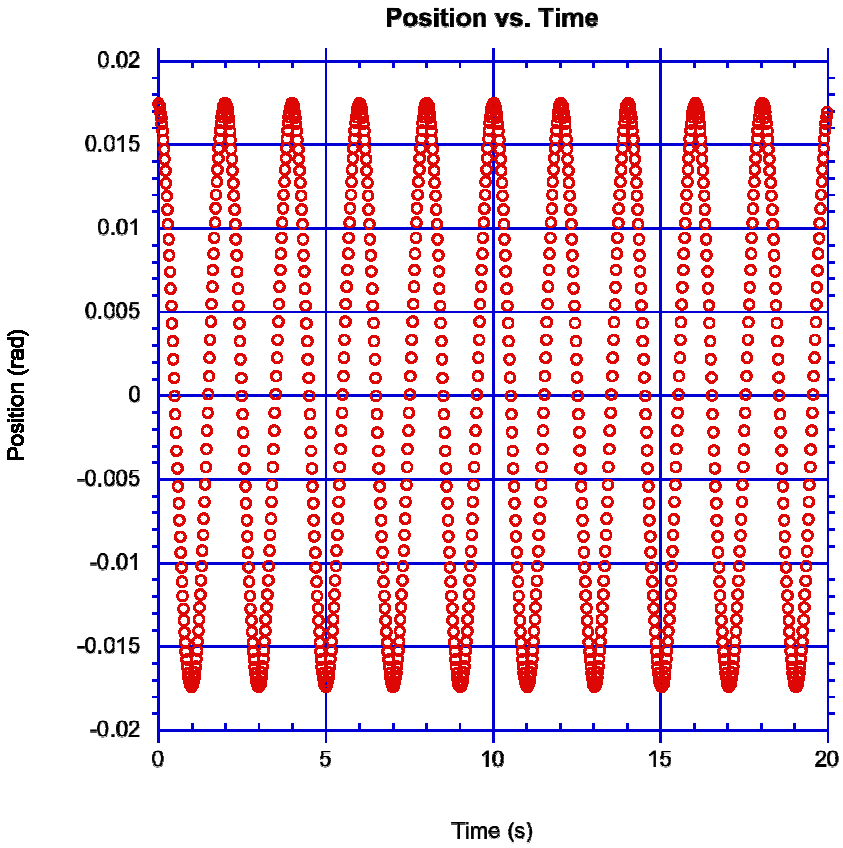

Fig. 7. Position vs. time for an undamped oscillator using Euler-Cromer method

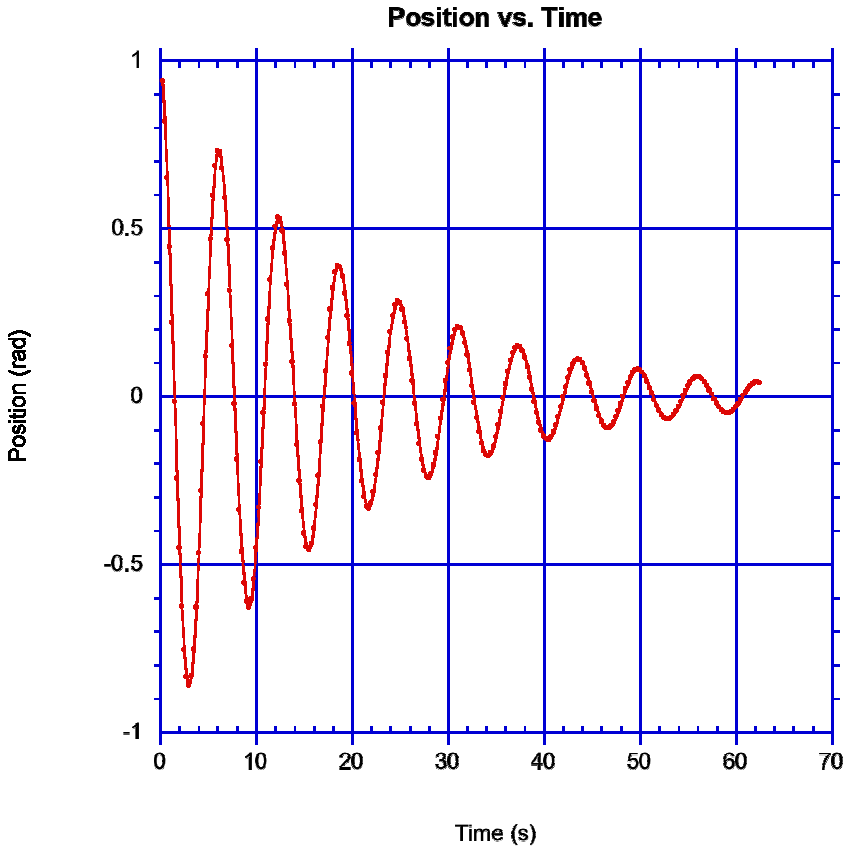

Fig. 8. Position vs. time for a damped oscillator using the Euler-Cromer method

There is another way to improve the Euler method. Feynman-Newton or half step method for solving first order differential equations corrects the problem with the by evaluating the function halfway through the interval, instead of at the endpoints. Therefore, the equation for the Feynman-Newton method is given by:

$$x(t_0 + h) = x(t_0) + h \cdot v(t_0 + \tfrac{h}{2}) \tag{4}$$

The Feynman-Newton algorithm is also simple to program. Figure 9 shows the loop that one could use to implement the Feynman-Newton method in a FORTRAN program.

```
Xold = Xint
Vold = Vint + Dt * A / 2
DO I = 1, 250
X = Xold + dt * Vold
A = -(C1 * X + C2 * Vold)
V = Vold + Dt * A
TimeX = TimeX + Dt
Timev = TimeX + Dt / 2.
WRITE (30, 1000) TimeX, X,
TimeV, V, A
Xold = X
Vold = V
```

Fig. 9. Sample FORTRAN code for using Feynman-Newton method

Figure 10 depicts the position versus time for an initial angular displacement of 1º and an initial velocity of 0 rad/s that was obtained using the Feynman-Newton method. Notice how the graph is relatively stable and that the oscillator's amplitude does not increase as time increases. When one considers a damped oscillator, the Feynman-Newton method also does much better than the Euler method. Figure 11 depicts the position versus time for an initial displacement of 1m and an initial velocity of 0 m/s that was obtained using the Feynman-Newton method. Notice that extrema for the graph decrease as time increases. This is how the graph should look because the damped oscillator's amplitude should eventually decrease over time.

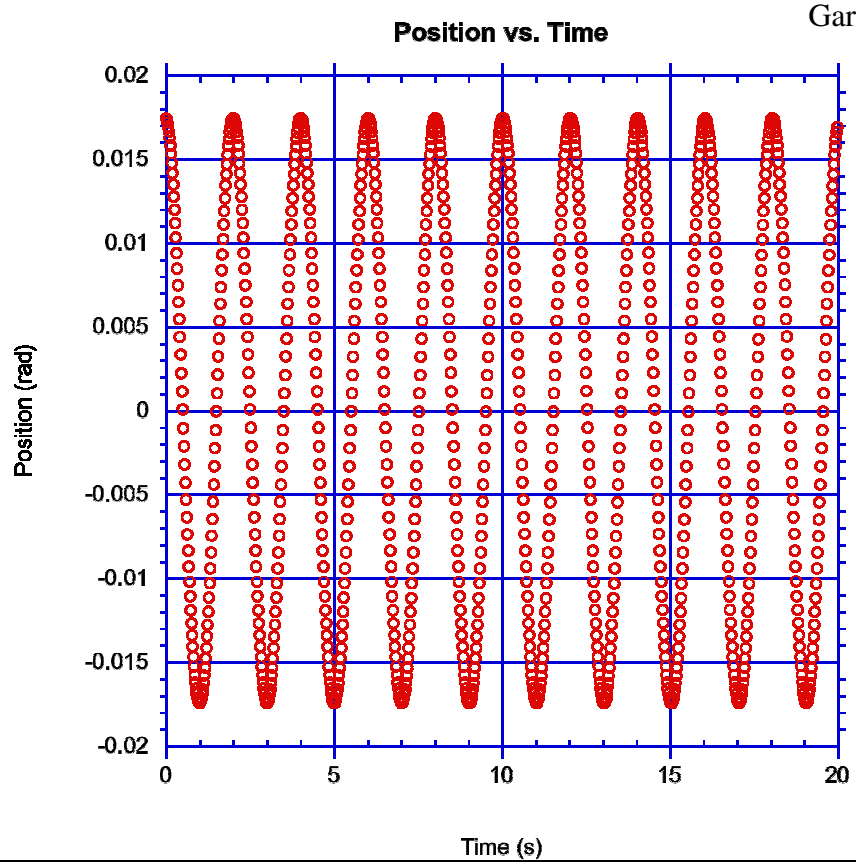

Fig. 10. Position vs. time for an undamped oscillator using Feynman-Newton method

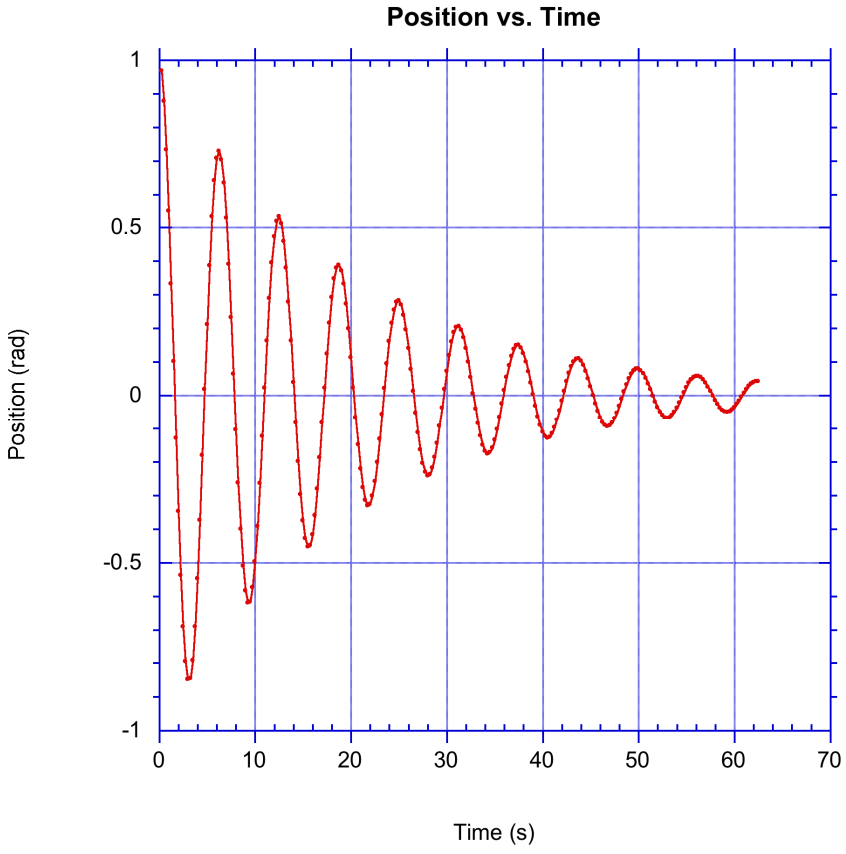

Fig. 11. Position vs. time for a damped oscillator using the Feynman-Newton method

We also found exact solutions for the damped and undamped oscillators. We used Maple to aid in our finding the exact solution. The undamped oscillator was easy because it is already known that an undamped oscillator can be modeled by the equation,

$$x(t) = A_0 \cdot \cos(\omega_0 \cdot t + \phi). \tag{5}$$

```
> m := 1;
                                  m := 1                                  (1)
> c := 0;
                                  c := 0                                  (2)
> k := 1;
                                  k := 1                                  (3)
> DE := m·diff(x(t), t$2) + c·diff(x(t), t) + k·x(t);
                                  DE := d^2/dt^2 x(t) + x(t)              (4)
> dsolve(DE = 0, x(t));
                          x(t) = _C1 sin(t) + _C2 cos(t)                  (5)
> solx := x(t) = _C1 sin(t) + _C2 cos(t);
                          solx := x(t) = _C1 sin(t) + _C2 cos(t)          (6)
> solv := diff(solx, t)
                          solv := d/dt x(t) = _C1 cos(t) - _C2 sin(t)     (7)
> solve({solx, solv}, [_C1, _C2]);
   [[_C1 = sin(t) x(t) + cos(t) (d/dt x(t)), _C2 = -((d/dt x(t)) sin(t) - x(t) cos(t)) / (cos(t)^2 + sin(t)^2)]]   (8)
> t := 0
                                  t := 0                                  (9)
> C2 := -(-x_0·cos(t)) / (cos(t)^2 + sin(t)^2);
                                  C2 := x_0                               (10)
> C1 := sin(t)·x_0;
                                  C1 := 0                                 (11)
> solution := C1·sin(b) + C2·cos(b);
                          solution := x_0 cos(b)                          (12)
```

Fig. 12. Steps to find the analytical solution to the undamped oscillator used in the program

Figure 12 shows the derivation of this equation for the specific example of the undamped oscillator with the initial conditions we used to make the graphs. In order to find the constants _C1 and _C2, we found the solution using the auxiliary equation. Then we took the derivative of the solution, because we need two equations to find two variables. Since we knew the initial position and velocity, we were able to solve the system of equations to find _C1 and _C2. Figure 13 shows the graph of the analytical solution, a simple cosine wave. Figure 13 shows the graph of the position vs. time for the analytical solution.

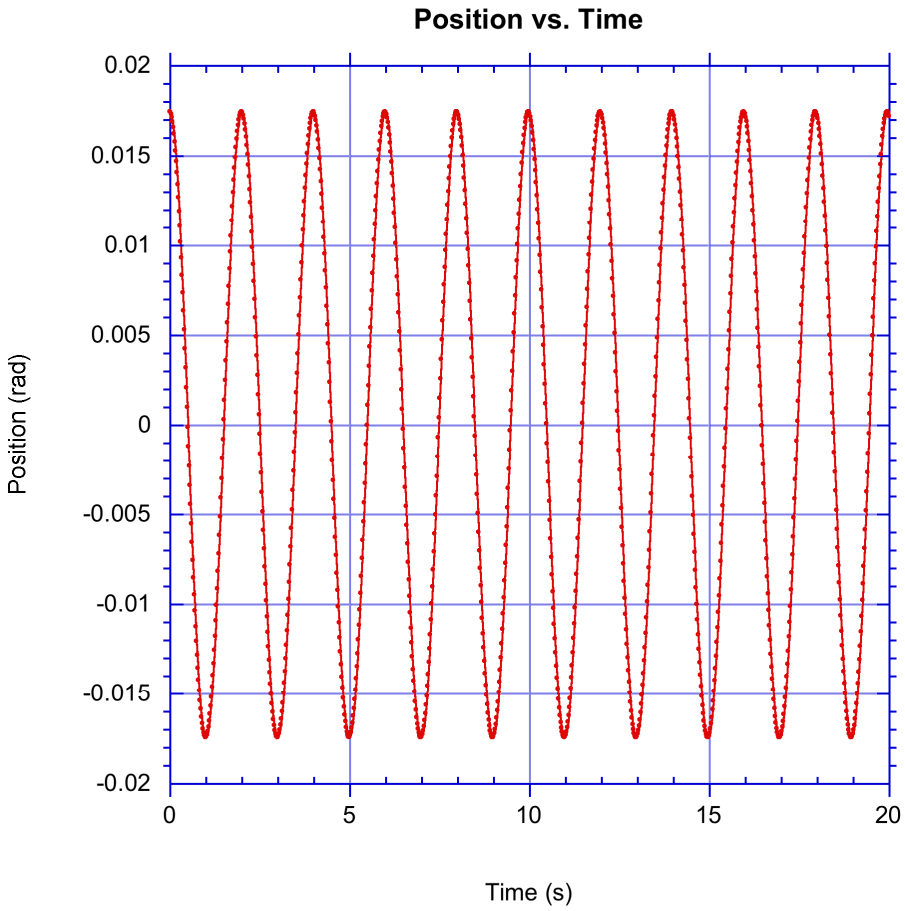

Fig. 13. Position vs. time for an undamped oscillator using the analytical solution

The analytical solution for the damped oscillator however is more complicated. We used Maple to aid in our finding the exact solution. We found that the exact solution of the damped oscillator we modeled was,

$$x(t) = \frac{1}{399} \cdot x_0 \cdot \sqrt{399} \cdot e^{-\frac{1}{20} \cdot t} \cdot \sin\left(\frac{1}{20} \cdot \sqrt{399} \cdot t\right) + x_0 \cdot e^{-\frac{1}{20} \cdot t} \cdot \cos\left(\frac{1}{20} \cdot \sqrt{399} \cdot t\right) \quad (6)$$

Figure 14 shows the graph of the analytical solution. Figure 15 shows the derivation of the analytical solution. We found _C1 and _C2 the same way we found them in for the undamped oscillator.

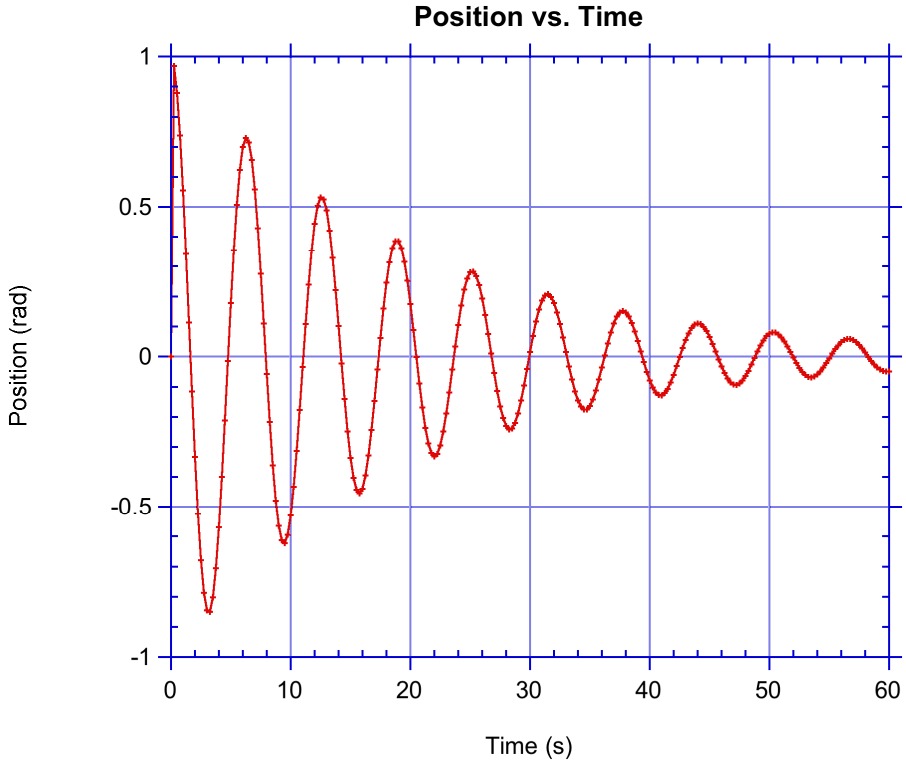

Fig. 14. Position vs. time for a damped oscillator using the analytical solution

```
> m := 1;
```

$$m := 1 \tag{1}$$

```
> c := 0.1;
```

$$c := 0.1 \tag{2}$$

```
> k := 1;
```

$$k := 1 \tag{3}$$

```
> DE := m·diff(x(t), t$2) + c·diff(x(t), t) + k·x(t);
```

$$DE := \frac{d^2}{dt^2} x(t) + 0.1 \left(\frac{d}{dt} x(t) \right) + x(t) \tag{4}$$

```
> dsolve(DE = 0, x(t));
```

$$x(t) = \_C1\, e^{-\frac{1}{20} t} \sin\left(\frac{1}{20} \sqrt{399}\, t \right) + \_C2\, e^{-\frac{1}{20} t} \cos\left(\frac{1}{20} \sqrt{399}\, t \right) \tag{5}$$

```
> solx := x(t) = _C1 e^(-1/20 t) sin(1/20 sqrt(399) t) + _C2 e^(-1/20 t) cos(1/20 sqrt(399) t);
```

$$solx := x(t) = \_C1\, e^{-\frac{1}{20} t} \sin\left(\frac{1}{20} \sqrt{399}\, t \right) + \_C2\, e^{-\frac{1}{20} t} \cos\left(\frac{1}{20} \sqrt{399}\, t \right) \tag{6}$$

```
> solv := diff(solx, t)
```

$$\begin{aligned} solv := \frac{d}{dt} x(t) = & -\frac{1}{20} \_C1\, e^{-\frac{1}{20} t} \sin\left(\frac{1}{20} \sqrt{399}\, t \right) \\ & + \frac{1}{20} \_C1\, e^{-\frac{1}{20} t} \cos\left(\frac{1}{20} \sqrt{399}\, t \right) \sqrt{399} - \frac{1}{20} \_C2\, e^{-\frac{1}{20} t} \cos\left(\frac{1}{20} \sqrt{399}\, t \right) \\ & - \frac{1}{20} \_C2\, e^{-\frac{1}{20} t} \sin\left(\frac{1}{20} \sqrt{399}\, t \right) \sqrt{399} \end{aligned} \tag{7}$$

```
> solve({solx, solv}, [_C1, _C2]);
```

$$\left[\left[\_C1 = \frac{1}{399} \right. \right. \tag{8}$$

$$\left. \frac{1}{e^{-\frac{1}{20} t}} \left(399\, x(t) \sin\left(\frac{1}{20} \sqrt{399}\, t \right) + 20 \cos\left(\frac{1}{20} \sqrt{399}\, t \right) \sqrt{399} \left(\frac{d}{dt} x(t) \right) \right. \right.$$

$$\left. \left. + x(t) \cos\left(\frac{1}{20} \sqrt{399}\, t \right) \sqrt{399} \right), \_C2 = -\frac{1}{399} \right.$$

$$\left. \frac{1}{e^{-\frac{1}{20} t} \left(\cos\left(\frac{1}{20} \sqrt{399}\, t \right)^2 + \sin\left(\frac{1}{20} \sqrt{399}\, t \right)^2 \right)} \left(\left(20 \left(\frac{d}{dt} x(t) \right) \right. \right. \right.$$

$$\sin\left(\frac{1}{20}\sqrt{399}\,t\right)+x(t)\sin\left(\frac{1}{20}\sqrt{399}\,t\right)-x(t)\cos\left(\frac{1}{20}\sqrt{399}\,t\right)\sqrt{399}\Bigg)\sqrt{399}\Bigg)$$

$$> \quad t := 0$$

$$t := 0 \tag{9}$$

$$> \quad C2 := -\frac{1}{399}\,\frac{\left(x_0\cdot\sin\left(\frac{1}{20}\sqrt{399}\,t\right)-x_0\cdot\cos\left(\frac{1}{20}\sqrt{399}\,t\right)\sqrt{399}\right)\sqrt{399}}{e^{-\frac{1}{20}t}\left(\cos\left(\frac{1}{20}\sqrt{399}\,t\right)^2+\sin\left(\frac{1}{20}\sqrt{399}\,t\right)^2\right)};$$

$$C2 := x_0 \tag{10}$$

$$> \quad C1 := \frac{1}{399}\,\frac{399\cdot x_0\cdot\sin\left(\frac{1}{20}\sqrt{399}\,t\right)+x_0\cdot\cos\left(\frac{1}{20}\sqrt{399}\,t\right)\sqrt{399}}{e^{-\frac{1}{20}t}};$$

$$C1 := \frac{1}{399}\,x_0\sqrt{399} \tag{11}$$

$$> \quad solution := C1\cdot e^{-\frac{1}{20}b}\sin\left(\frac{1}{20}\sqrt{399}\,b\right)+C2\cdot e^{-\frac{1}{20}b}\cos\left(\frac{1}{20}\sqrt{399}\,b\right)$$

$$solution := \frac{1}{399}\,x_0\sqrt{399}\,e^{-\frac{1}{20}b}\sin\left(\frac{1}{20}\sqrt{399}\,b\right)+x_0\,e^{-\frac{1}{20}b}\cos\left(\frac{1}{20}\sqrt{399}\,b\right) \tag{12}$$

Fig. 15. Steps to find the analytical solution to the damped oscillator used in the program

From the exact solution we derived for the damped oscillator, we compared each of the numerical methods to the exact graph, to see which one is more reliable. First, we tried the Euler method, which we predicted would not follow the exact curve, as shown in Figure 16.

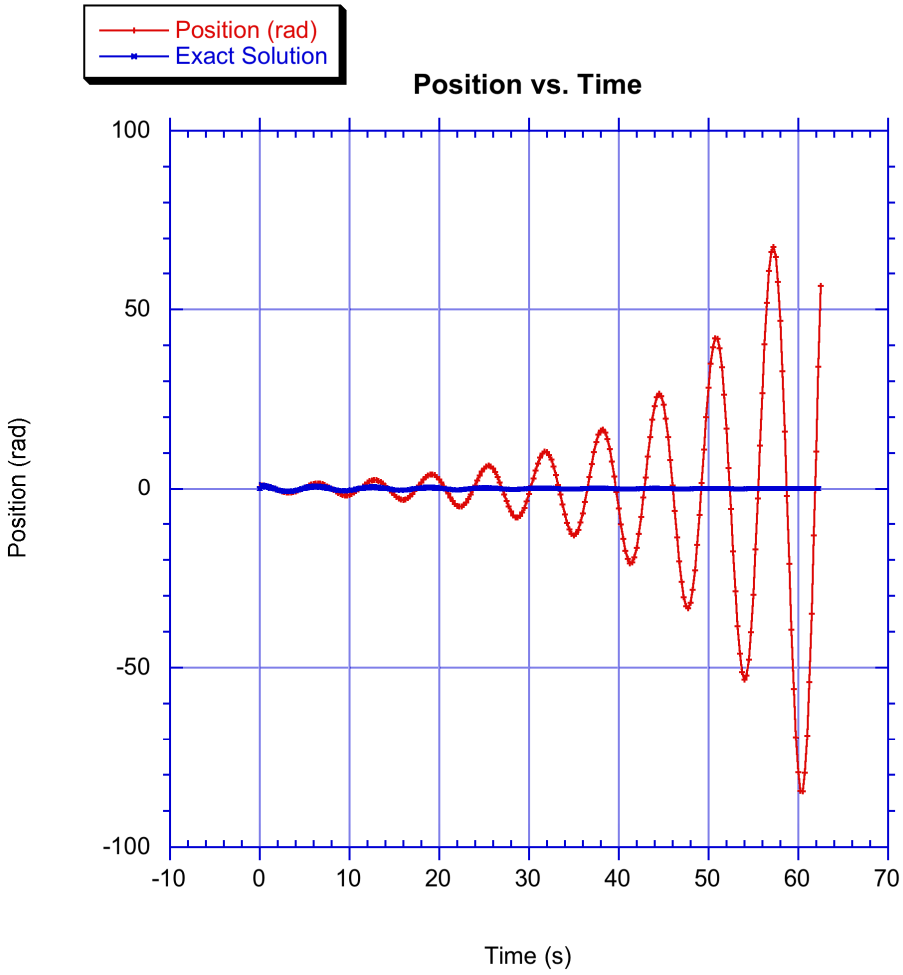

Fig. 16. Position vs. time for a damped oscillator using the analytical solution and Euler method

Notice how the red Euler curve completely diverges from the exact solution. Therefore, this numerical method is not reliable for approximating an oscillator. We then tried to compare the Euler-Cromer method to the exact solution. We saw a dramatic improvement from the Euler graph before, because it does not diverge. Compared to the exact graph however, it also comes very close. The Euler-Cromer graph compared to the exact solution is shown in Figure 17.

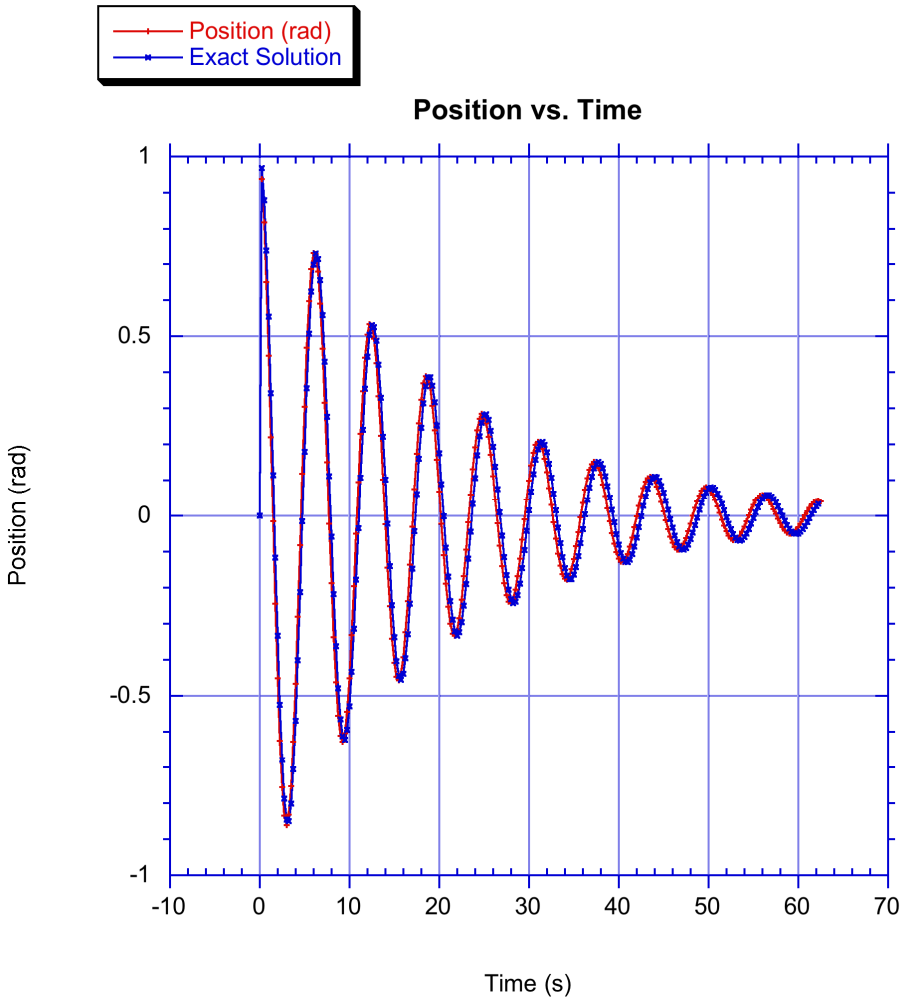

Fig. 17. Position vs. time for a damped oscillator using the analytical solution and Euler-Cromer method

Then we compared the data from the analytical solution to the data from the Feynman-Newton method. Compared to the Euler method it was also a very clear improvement. Yet next to the Euler-Cromer method, it looks very similar. The comparison of the Feynman-Newton method to the analytic solution is shown in Figure 18.

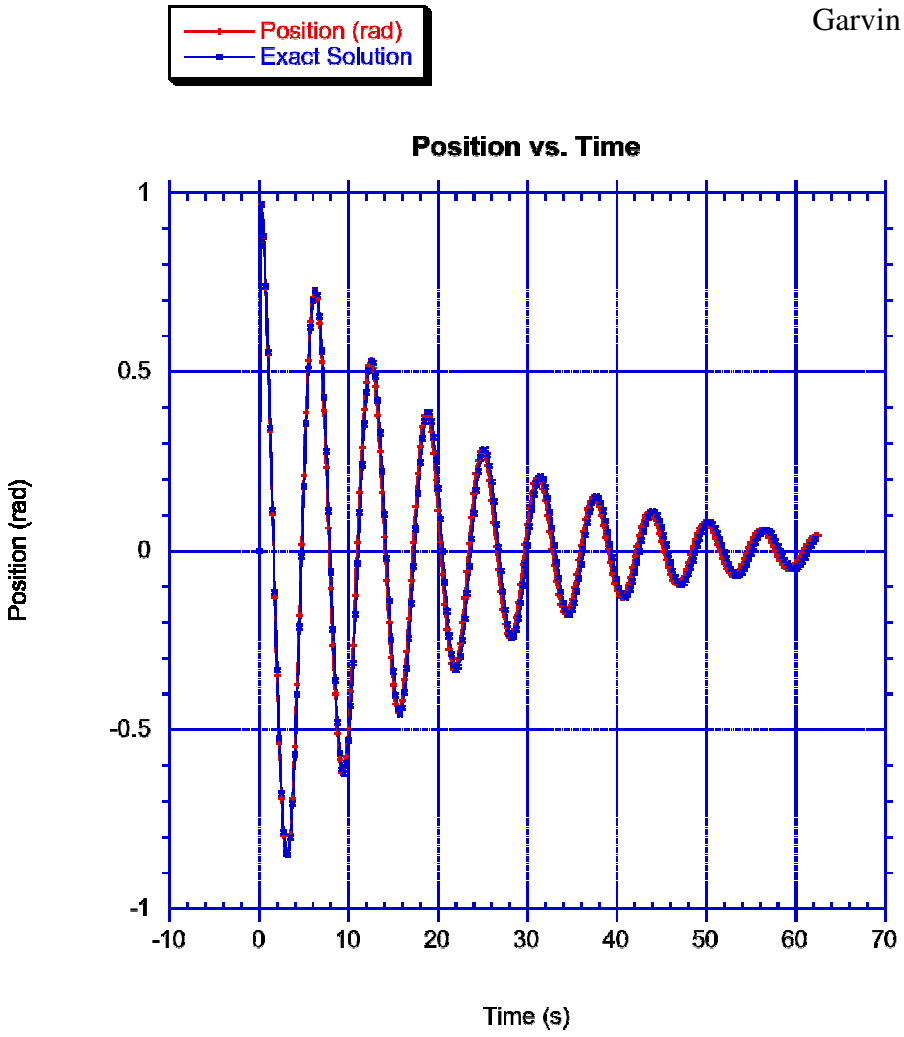

Fig. 18. Position vs. time for a damped oscillator using the analytical solution and Euler-Cromer method

Comparing the Euler-Cromer graph next to the Feynman-Newton graph, we noticed a little difference in the accuracy of each method, where Feynman-Newton came closer to the exact solution. To see this difference in a better light, we took the data of each graph and found the absolute value of the difference between the data from the numerical method. Then we plotted the data as a function of time, and we saw that the Euler-Cromer method was more inaccurate in all of its points, and the Feynman-Newton method diverged less from the exact solution. Figure 19 shows the graph of the differences.

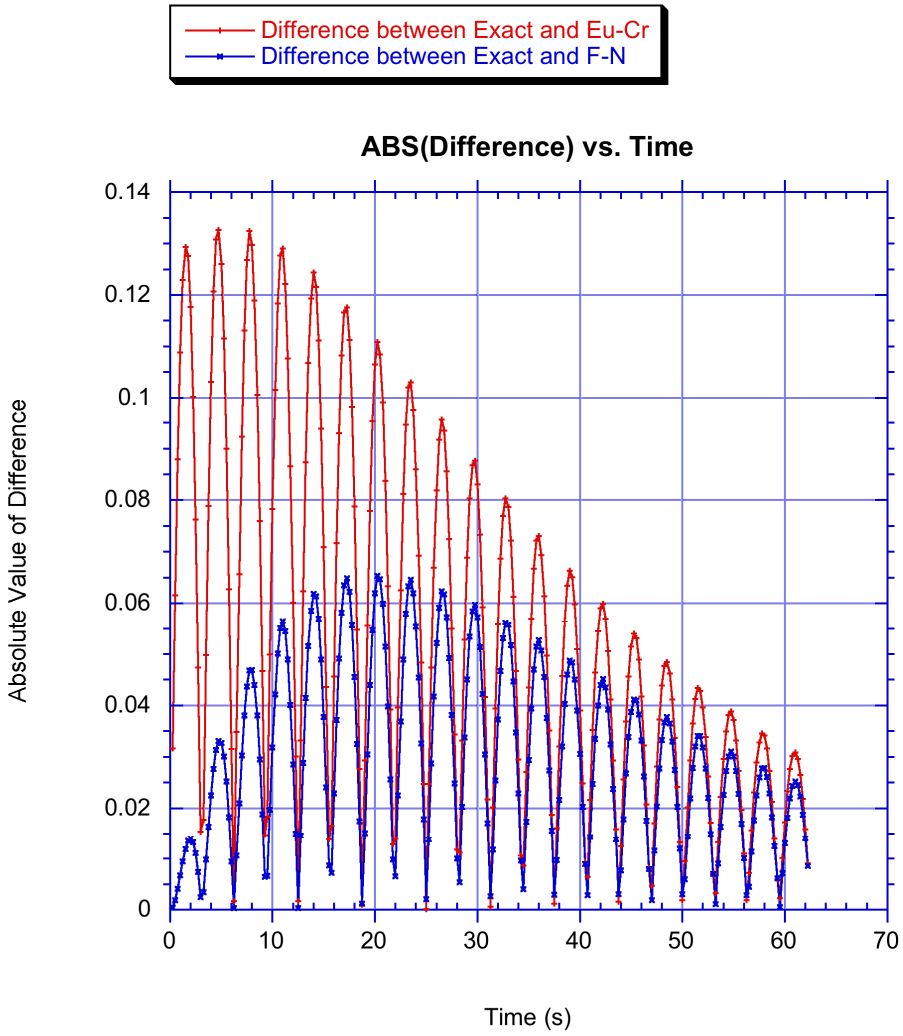

Fig. 19. Difference vs. time comparing the Euler-Cromer method and Feynman-Newton method to the analytic solution

Because of our findings, we conclude that out of the three numerical methods, Euler, Euler-Cromer, and Feynman-Newton, the Feynman-Newton method best approximates the exact solution to both damped and undamped oscillators. While a smaller time step for each method would help to better approximate the analytic solution, the Feynman-Newton method is the most efficient, because it does not need as small of a time step, and that takes less computing time.