**Anastashia Jebraeilli**
Department of Physics and Astronomy
PHYS8602 - Dr. David Landau
University of Georgia, Athens, Georgia, 30605
(Dated: February 5th, 2021)

In this paper, the motion of a simple harmonic oscillator whose hamiltonian is well known is simulated using Euler's method. Using this method phase space plots are determined. The same simulation is run with varying time steps, $\Delta = 0.001,\ 0.01,\ and\ 0.1$ for a total of 10 seconds starting at a time of 0 seconds. The variable time steps are used to illustrate that if too late of a time step is chosen for an Euler method simulation it is possible to generate a state in which the system is unrecoverable.

## I. INTRODUCTION

In our course Computer Simulations of Materials, PHYS8602, many physical problems will be solved by using numerical computational methods. In order to complete this simulation, we need to get acquainted with C code syntax as well as the algorithms needed to compute the dynamics of a physical system.

The very first problem that will be solved is the thoroughly studied and well-understood simple harmonic oscillator. The hamiltonian that describes simple harmonic motion is

$$\mathcal{H} = \frac{p^2}{2m} + \frac{kx^2}{2} \qquad (1.01)$$

$p$ is the momentum, $x$ is the position, $m$ is the mass, and $k$is the spring constant. To determine the dynamics of such a system we will be using Euler's method of time-integration of the equations of motion over different time intervals of $\Delta = 0.001,\ 0.01,\ and\ 0.1$ for a total of 10 seconds starting at a time of 0 seconds.

## II. METHODS

Euler's method is known to be the oldest and easiest to use method of solving a differential equation numerically. Euler's method is not the most efficient method in most cases, but it is relatively easily implemented and good for simple scenario applications. Euler's method is broadly defined in mathematics by taking

$$y_{n+1} = y_n + hf(t_n, y_n), \quad 0 \le n \le N - 1 \qquad (2.01)$$

to be exact, where $y(t)$is the approximate solution at node points is

$$y(t_n) = y_h(t_n) = y_n,\ n = 0,\ 1,\ \dots,\ N. \qquad (2.02)$$

For this assignment, obtaining approximate values of $x_i \simeq x(t_i)\ and\ v_i \simeq v(t_i)\ for\ t = t_0 + ih\ with\ i = 1, \dots, N$ are needed to numerically solve the one-dimensional simple harmonic oscillator. Choosing the initial conditions $x_0,\ t_0,\ and\ v_0$for simplicity to be 1.0, 0.0, and 5.0. The old position and velocity are used to calculate the new position and velocity. The pseudo code illustrates the C code algorithm that was used very well and is shown below (*Figure 1)*:

```
// Psuedo Code

// 1. start
// 2. define the total run time of integration (TF, in seconds)
// 3. define the function f(x,y)
// 4. define the initial condititions (x0, y0), number of steps (n), and the point of calculation (xn)
// 5. create the file for the data to be written to
// 6. compute the step size (h) [ h =( xn - x0 ) / n ]
// 7. i = 1
// 8. loop
//      yn = y0 + h * f( x0 + i*h, y0 )
//      y0 = yn
//      i += 1
//    while i < n
// 9. return the result, yn
// 10. stop

/////////////////////////////////////////
```

*Figure 1.* Simple pseudocode that describes how the simulation will be implemented.

Additionally, it is seen that there are a few constant values that must be assumed, specifically, $k = 1\ and\ m = 1$. From these assumptions, as well as setting the initial conditions, initial momentum and energy values are found.

## III. RESULTS AND ERROR ANALYSIS

50 total simulations were ran on the UGA teaching cluster for each time step ($\Delta = 0.001,\ 0.01,\ and\ 0.1$). The data files from each run contained 5 columns containing the time, position, velocity, momentum, and total energy.

An example of a portion of one of these data files is shown below for
$\Delta = 0.001$(*Figure 2*):

| | | | | |
|---|---|---|---|---|
| 0.000000e+00 | 6.417773e+00 | -5.894470e-03 | -5.894470e-03 | 2.059392e+01 |
| 1.000000e-03 | 6.417767e+00 | -1.231224e-02 | -1.231224e-02 | 2.059395e+01 |
| 2.000000e-03 | 6.417755e+00 | -1.873001e-02 | -1.873001e-02 | 2.059397e+01 |
| 3.000000e-03 | 6.417736e+00 | -2.514777e-02 | -2.514777e-02 | 2.059399e+01 |
| 4.000000e-03 | 6.417711e+00 | -3.156550e-02 | -3.156550e-02 | 2.059401e+01 |
| 5.000000e-03 | 6.417680e+00 | -3.798321e-02 | -3.798321e-02 | 2.059403e+01 |
| 6.000000e-03 | 6.417642e+00 | -4.440089e-02 | -4.440089e-02 | 2.059405e+01 |
| 7.000000e-03 | 6.417597e+00 | -5.081854e-02 | -5.081854e-02 | 2.059407e+01 |
| 8.000000e-03 | 6.417547e+00 | -5.723613e-02 | -5.723613e-02 | 2.059409e+01 |
| 9.000000e-03 | 6.417489e+00 | -6.365368e-02 | -6.365368e-02 | 2.059411e+01 |
| 1.000000e-02 | 6.417426e+00 | -7.007117e-02 | -7.007117e-02 | 2.059413e+01 |
| 1.100000e-02 | 6.417356e+00 | -7.648859e-02 | -7.648859e-02 | 2.059415e+01 |
| 1.200000e-02 | 6.417279e+00 | -8.290595e-02 | -8.290595e-02 | 2.059417e+01 |
| 1.300000e-02 | 6.417196e+00 | -8.932323e-02 | -8.932323e-02 | 2.059419e+01 |
| 1.400000e-02 | 6.417107e+00 | -9.574042e-02 | -9.574042e-02 | 2.059421e+01 |
| 1.500000e-02 | 6.417011e+00 | -1.021575e-01 | -1.021575e-01 | 2.059423e+01 |
| 1.600000e-02 | 6.416909e+00 | -1.085745e-01 | -1.085745e-01 | 2.059425e+01 |
| 1.700000e-02 | 6.416800e+00 | -1.149915e-01 | -1.149915e-01 | 2.059428e+01 |
| 1.800000e-02 | 6.416685e+00 | -1.214083e-01 | -1.214083e-01 | 2.059430e+01 |
| 1.900000e-02 | 6.416564e+00 | -1.278249e-01 | -1.278249e-01 | 2.059432e+01 |
| 2.000000e-02 | 6.416436e+00 | -1.342415e-01 | -1.342415e-01 | 2.059434e+01 |
| 2.100000e-02 | 6.416302e+00 | -1.406579e-01 | -1.406579e-01 | 2.059436e+01 |
| 2.200000e-02 | 6.416161e+00 | -1.470742e-01 | -1.470742e-01 | 2.059438e+01 |
| 2.300000e-02 | 6.416014e+00 | -1.534904e-01 | -1.534904e-01 | 2.059440e+01 |
| 2.400000e-02 | 6.415861e+00 | -1.599064e-01 | -1.599064e-01 | 2.059442e+01 |
| 2.500000e-02 | 6.415701e+00 | -1.663223e-01 | -1.663223e-01 | 2.059444e+01 |
| 2.600000e-02 | 6.415534e+00 | -1.727380e-01 | -1.727380e-01 | 2.059446e+01 |

*Figure 2.* A sample of the data output from the simulation.
Each of these data files was analyzed using a separate python code in order to create phase plots for each $\Delta$.These phase plots are shown below *(Figures 3a-3c*):

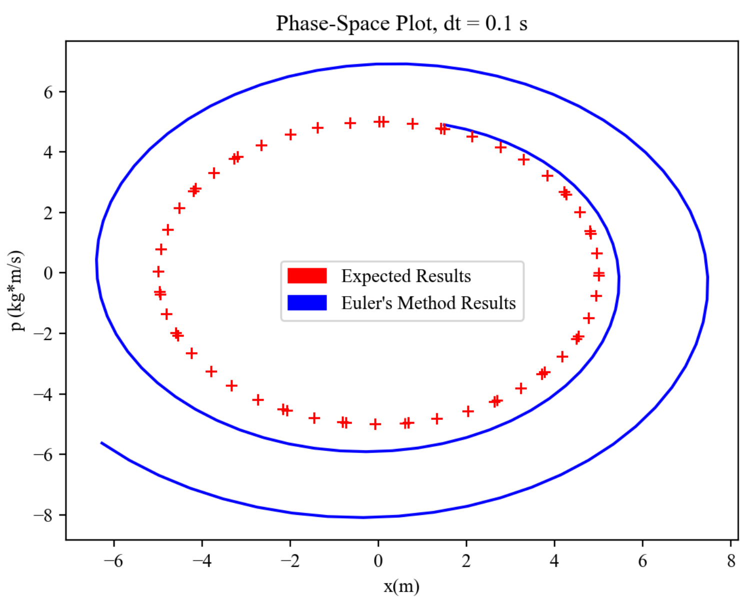


*Figure 3a.* This figure is the Phase-Space plot for the simple harmonic system when a time step of $\Delta = 0.1s$is used, running from an initial time of t=0 to t=10s. The plot illustrates the results from the simulation using Euler's method as well as what the expected results for the system should be. It is very apparent that the results from Euler's method start off matching the expected results well but quickly spiral away.

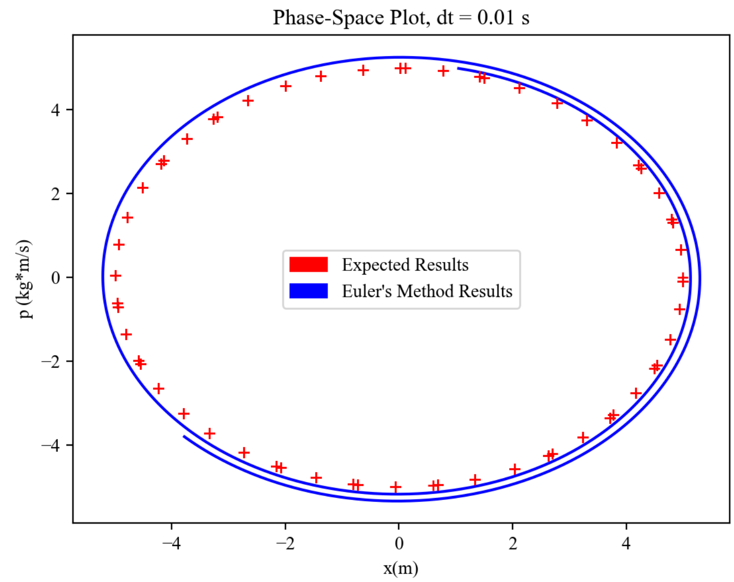


*Figure 3b.* This figure is again a phase-space plot of the Euler's method and expected results of the simple harmonic oscillator in question. However, in this figure, the data pictured comes from the results of the simulation being run with a time-step of $\Delta = 0.01s$, from an initial time of t=0 to t=10s. The Euler's method results using this smaller time step map to the expected results much better than the $\Delta = 0.1s$results did. However, as the simulation moved through time the Euler's method results did visibly diverge from the expected results in a spiral pattern similar to $\Delta = 0.1s$, even though it was less dramatic.

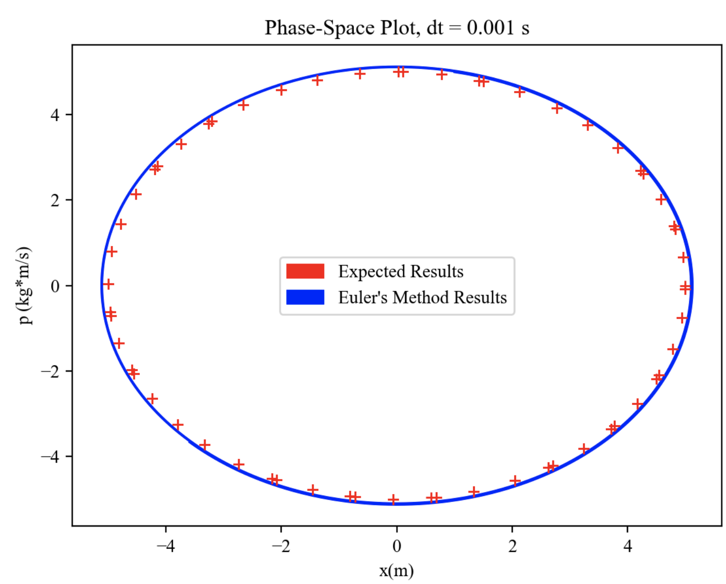


*Figure 3c.* This phase-space plot depicts Euler's method results when a time step of $\Delta = 0.001s$ is used for a simulation running from an initial time of t=0 to a final time of

t=10s. These Euler's method results seem to match the expected results perfectly. There is not a visible diverging spiral pattern in this phase-space plot and Euler's method results instead create a perfect ellipse matching what is expected of a simple harmonic oscillator.

Additionally, a plot was created of both the errors in E and the errors in x as functions of $\Delta$.Specifically, these errors were found by computing the standard error, defined as

$$SE = \frac{\sigma}{\sqrt{n}} \qquad (3.01)$$

where σ is the standard deviation and n is the number of runs (n=50), in E and x throughout all 50 simulations run per $\Delta$. $\Delta$ was plotted along the x-axis as $log(\Delta)$so that the graphs overall trends would be more obviously shown. These plots are shown below (*Figure 4a-b)*:

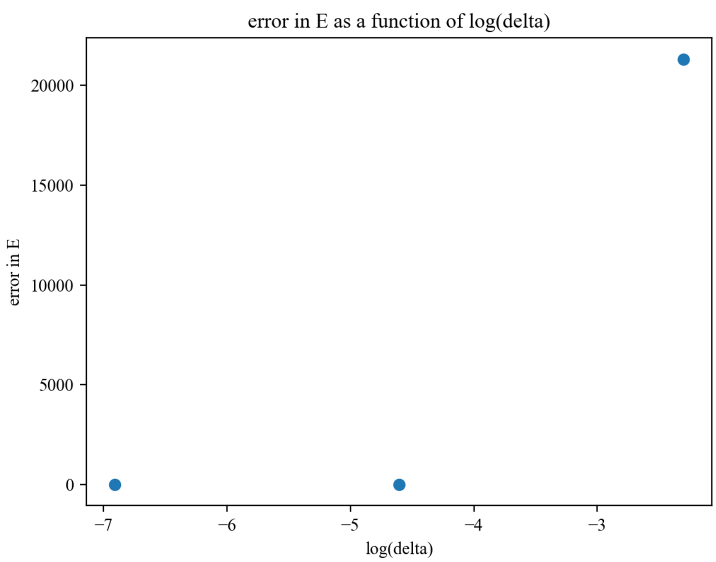


*Figure 4a.* The error in E (energy) as a function of the log(Δ)

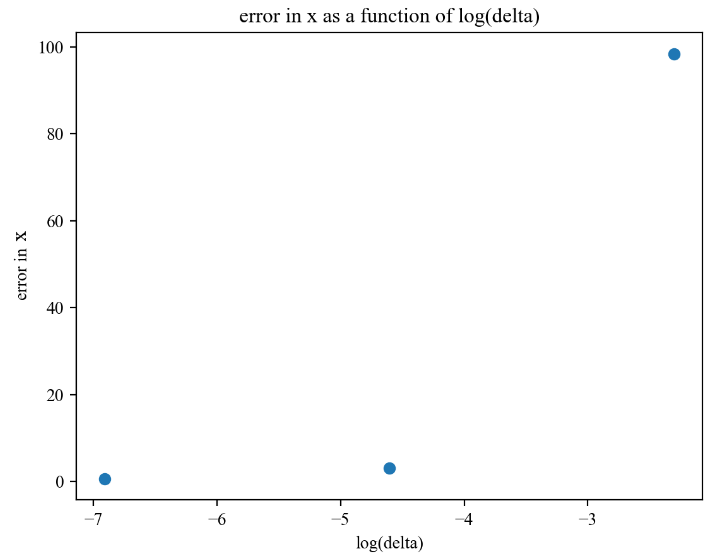


*Figure 4b.* The error in x (position) as a function of the log(Δ)

These plots of the results from the simulations are very illuminating. The standard error and $log(\Delta)$clearly shows that as the time steps became larger, the standard error became larger in both E and x, almost exponentially. From the phase space plots, it is seen that as the time step $\Delta$became larger there a huge amount of error as the cycle of oscillation moved through time. A conclusion can be drawn from these plots that Euler's method requires a very short time step in order to decrease the amount of error found in the result.

## IV. CONCLUSION

From these results, it can be concluded that while Euler's method is good for simple scenarios because it is generally easy to implement even for a mildly complicated system, the time step has to be incredibly small as well as integrated out for quite a large time. If one wants to decrease computation time when using the Euler method, there is also a substantial sacrifice of accuracy. Furthermore, from looking at the phase space plot of

$\Delta = 0.1$, a conclusion can be drawn that it is quite possible for the Euler method to generate a state in which the system is unrecoverable. If the step size is chosen to be too big the simulation could become "out of control," or a non physical scenario. In conclusion, the Euler method is easy but not that good because of how short the times steps have to be and how long the total integration time would need to be to generate results with low error.