Assignment 4

Brett Bonine
ASTR 5900: Numerical Methods

March 12, 2021

Problem 1

Numerically solve the harmonic oscillator problem

$$\frac{d^2x}{dt^2} = -\omega^2 x$$

a) Solve this as a system of Ordinary Differential Equations in ODEINT.

Answer: To use ODEINT for this problem, we preformed a variable substitution so we could express the second order differential equation as a system of two coupled first order differential equations:

$$\frac{dv}{dt} = -x; \quad \frac{dx}{dt} = v \tag{1}$$

We then read the above system into a 2d vector (our original input to ODEINT was a vector of 1d) and pass that through the same for loop we constructed in the previous assignment. This routine is carried out in the odeint_solver.cpp program. The resulting postiion and energy vs time graphs are shown below.

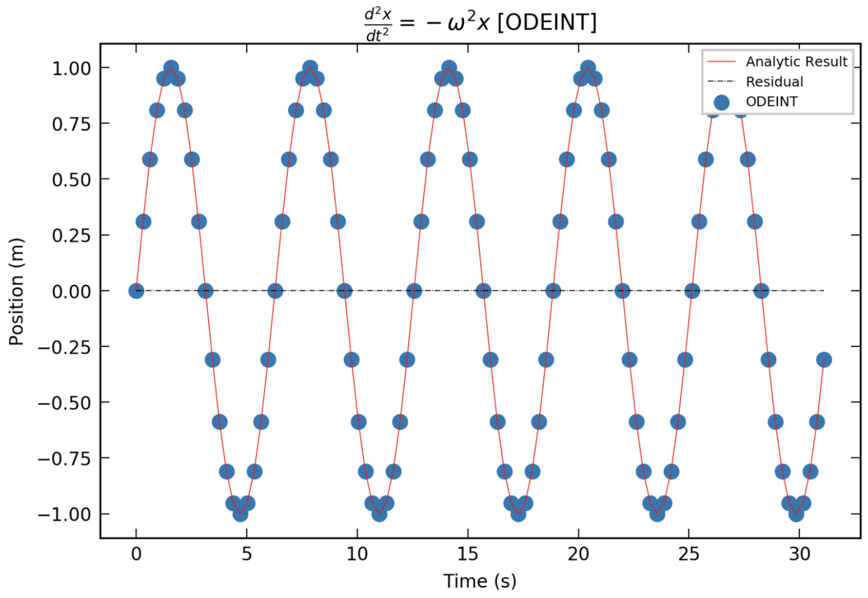


Figure 1: Position vs time graph for ODEINT program utilizing the Burlich-Stoer algorithm. This plot assumes an initial velocity of 1m/s, an initial position at the origin, and 100 steps.

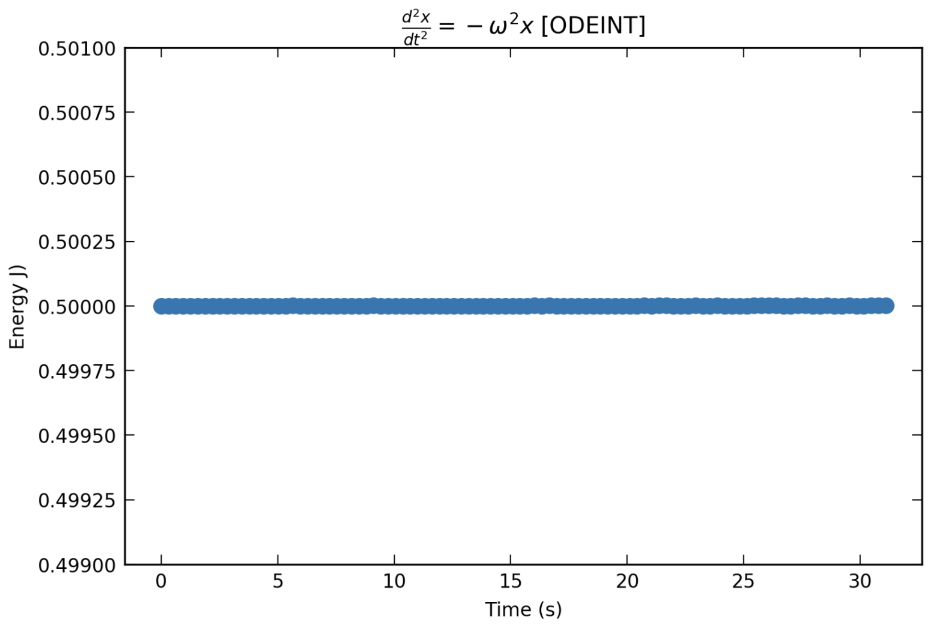


Figure 2: Energy vs time graph from the above ODEINT algorithm. As expected, the total energy of the system is constant in time.

b) Use the Leap-frog method. Generate the same plots as in part a.)

Answer: Recall that in the Leapfrog method, we have

$$x_{i+1} = x_i + v_i \triangle t + \frac{1}{2} a_i (\triangle t)^2$$

$$a_{i+1} = a(x_{i+1})$$

$$v_{i+1} = v_i + \frac{1}{2}(a_i + a_{i+1}) \triangle t$$

Our strategy will be as follows:

- Calculate the acceleration at the current point.
- Use the current acceleration and velocity to calculate the new position after one time-step.
- Find the acceleration at the new position.
- Calculate the velocity at the new point using current and previous acceleration.

Iterating through this process, we alternate calculating the position and velocity from the acceleration at every time step. A graph showing the resulting positions extracted from this procedure in leapfrog.cpp is shown below.

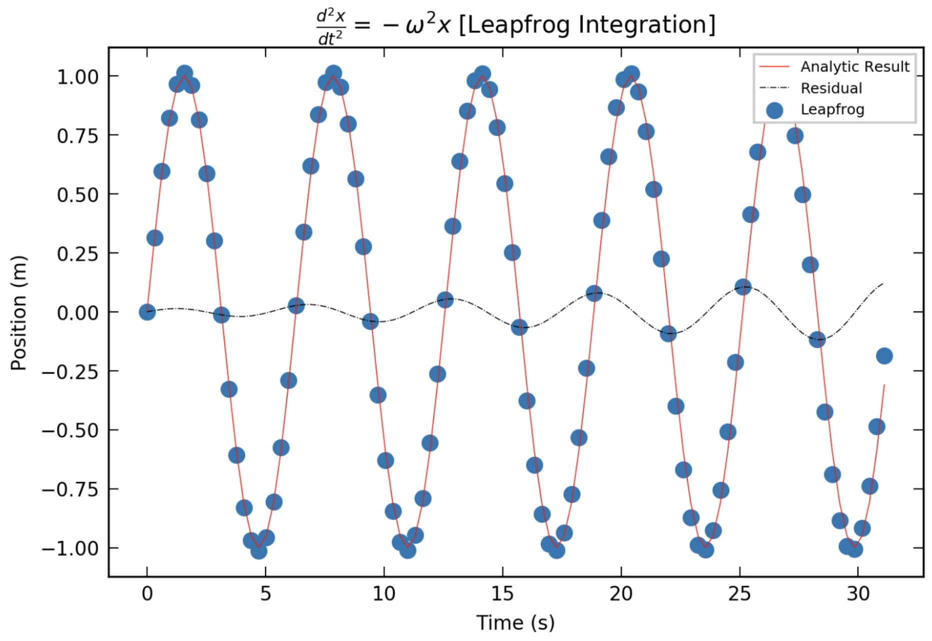


Figure 3: Position vs. Time graph for an initial velocity of 1 m/s and an initial displacement of zero and 100 steps. Note that the residual increases in magnitude with time.

We are also interested in investigating how the energy of the system changes with time. Since the spring force is a conservative force, the total energy should be constant at all times (Any energy lost from the potential is transferrd to the kinetic.) However, as shown in Figure 2, our algorithm fails to replicate the expected behavior. Instead of remaining constant, the total energy of the system oscillates around the analytic value.

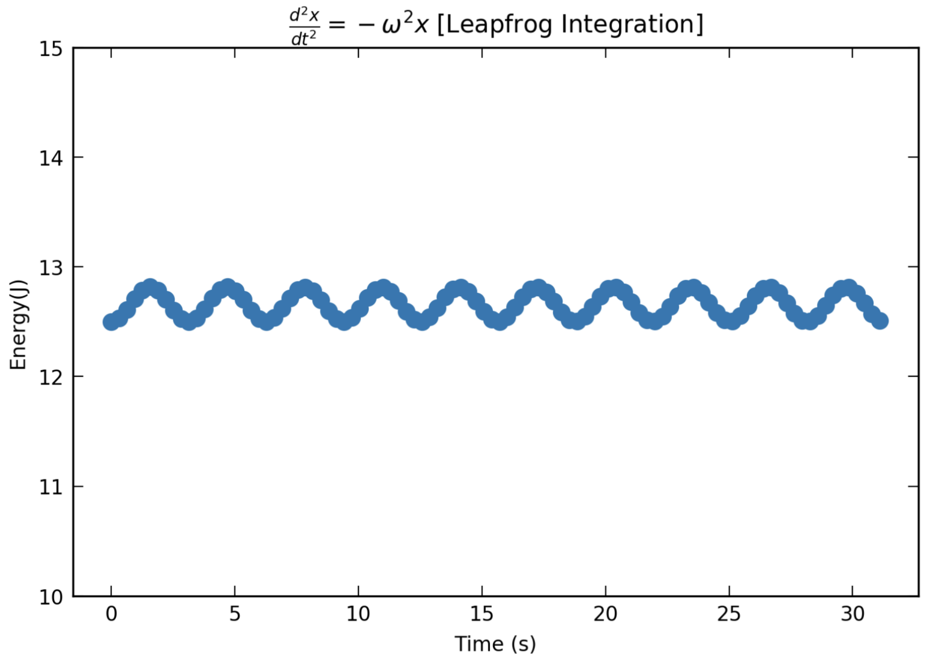


Figure 4: Energy vs. Time graph for the same situation. Note that the total energy oscillates around the mean. Since the analytic result is a constant, a residual is not shown.

c) Use the 2nd order Symplectic Method (aka, velocity-verlet method) described in class.

Answer: Recall that the velocity-verlet method calculates the generalized postion and momentum at each step as

$$q_{i+1]=q_i+\frac{(\triangle t}{m}}p_i + \frac{(\triangle t)^2}{2m}F(q_i)$$

$$p_{i+1} = p_i + \frac{(\triangle t)}{2}[F(q_i) + F(q_{i+1})]$$

Instead of calculating the acceleration twice in each iteration based off the current and updated position values, we instead evaluate the force and momentum.

Of course, since our system has assumed $k = m = w = 1$, we can see that these two algorithms are identical: When we express the force in terms of the acceleration according to Newton's 2nd law:

$$F = m\vec{a}$$

And the momentum p_i in terms of velocity and mass:

$$\vec{p} = m\vec{v}$$

We see that the two expressions in this algorithm are really just the leap-frog algorithm in disguise. As such, we expect to find the exact same results as we found in part a.

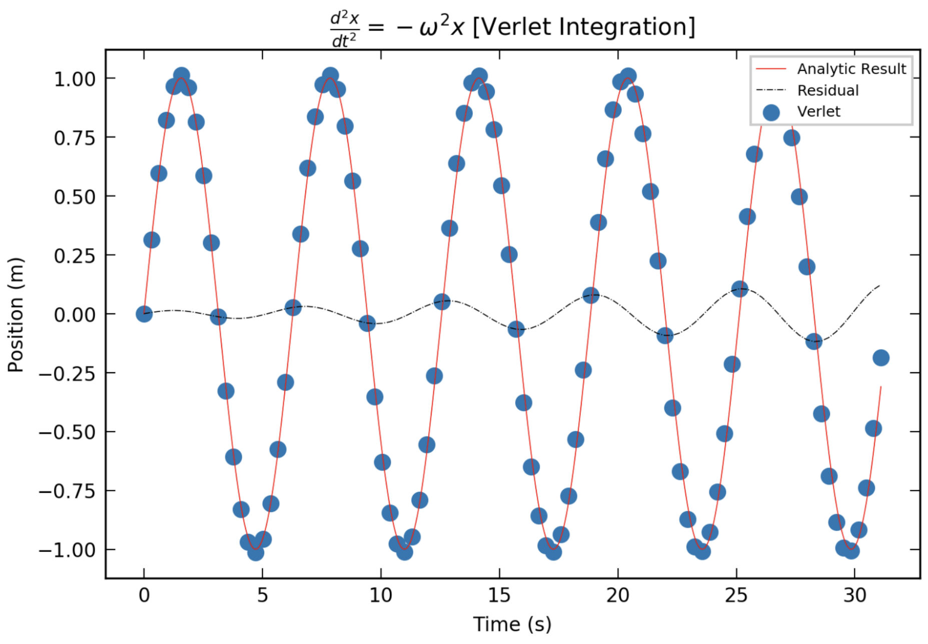


Figure 5: Position vs. Time graph for an initial velocity of 5 m/s and an initial displacement of zero. The expected sinusoidal behavior is obvious.

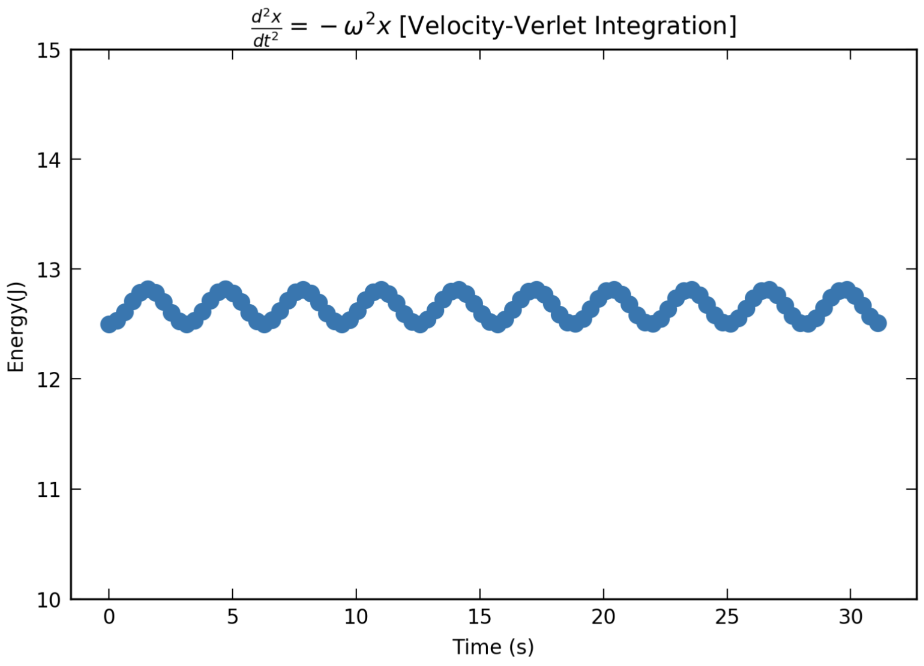


Figure 6: Energy vs. Time graph for the same situation. As with the Leap-frog case, the energy oscillates sinusoidal above the analytic (initial) value

d) Compare and contrast the Algorithms

Answer: As discussed in part b, the most straightforward algorithms to compare are the algorithms from part b and part c since they are virtually identical in terms of implementation and results. The only key difference between them is that the leapfrog algorithm explicitly calculates the acceleration at the intermediate leap-step and then uses that to find the new velocity. Meanwhile, the velocity-verlot algorithm computes the acceleration (or more specifically, in the process of computing the momentum.

The accuracy of these algorithms two algorithms is acceptable (as evident from the residuals in Figure 4 and Figure 2), but the magnitude of the residuals seems to increase as the simulation continues for more and more cycles. One way to maximize the accuracy of these algorithms is to increase the number of steps, shown below in Figure 6

Other than the gradually increasing deviation of the position from the analytic value, another issue with these two algorithms is that the energy fluctuates around the analytic value. The spring force that drives this system is conservative, i.e, the total energy of the system should be constant provided the oscillator is undamped and undriven. As we can see in Figure 2, the ODEINT implementation successfully reproduces this behavior. When combined with the improved behavior of the residuals for large t, the ability of ODEINT to conserve the energy of the system makes it the best of the three algorithms for our particular case.

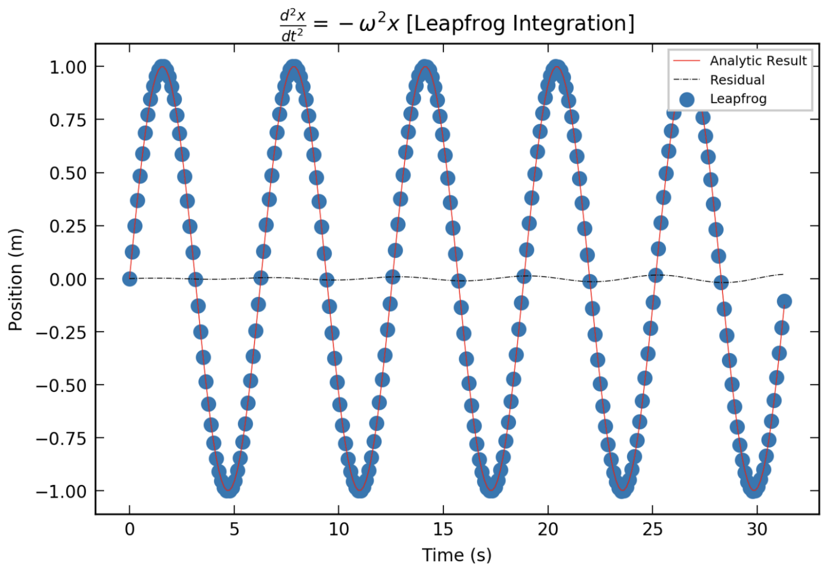
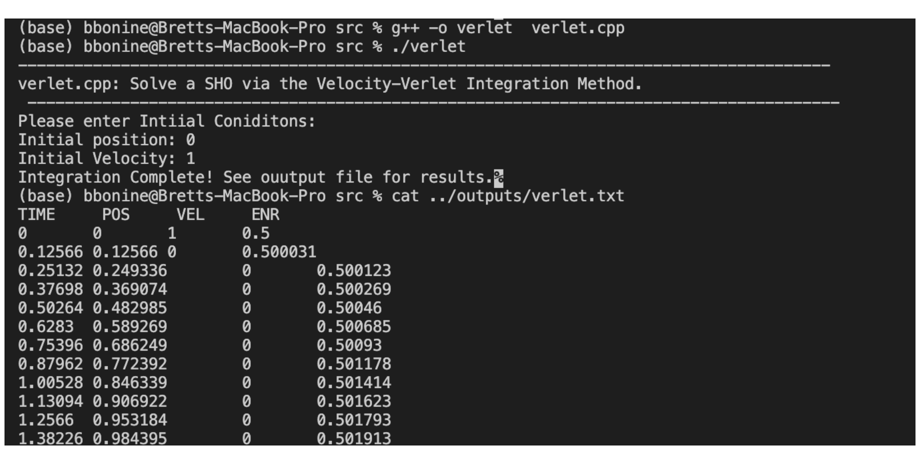


Figure 7: The same situation as Figure 2, but now with 250 steps instead of 100. The residuals are already noticeably smaller, but still don't quiet match the performance of ODEINT.

APPENDIX



```
(base) bbonine@Bretts-MacBook-Pro src % g++ -o verlet  verlet.cpp
(base) bbonine@Bretts-MacBook-Pro src % ./verlet
--------------------------------------------------------------------------------------
verlet.cpp: Solve a SHO via the Velocity-Verlet Integration Method.
 --------------------------------------------------------------------------------------
Please enter Intiial Coniditons:
Initial position: 0
Initial Velocity: 1
Integration Complete! See ouutput file for results.%
(base) bbonine@Bretts-MacBook-Pro src % cat ../outputs/verlet.txt
TIME     POS     VEL     ENR
0       0       1       0.5
0.12566 0.12566 0       0.500031
0.25132 0.249336        0       0.500123
0.37698 0.369074        0       0.500269
0.50264 0.482985        0       0.50046
0.6283  0.589269        0       0.500685
0.75396 0.686249        0       0.50093
0.87962 0.772392        0       0.501178
1.00528 0.846339        0       0.501414
1.13094 0.906922        0       0.501623
1.2566  0.953184        0       0.501793
1.38226 0.984395        0       0.501913
```

Figure 8: Terminal output showing the successful compilation and execution of one of the three executables, verlet.exe