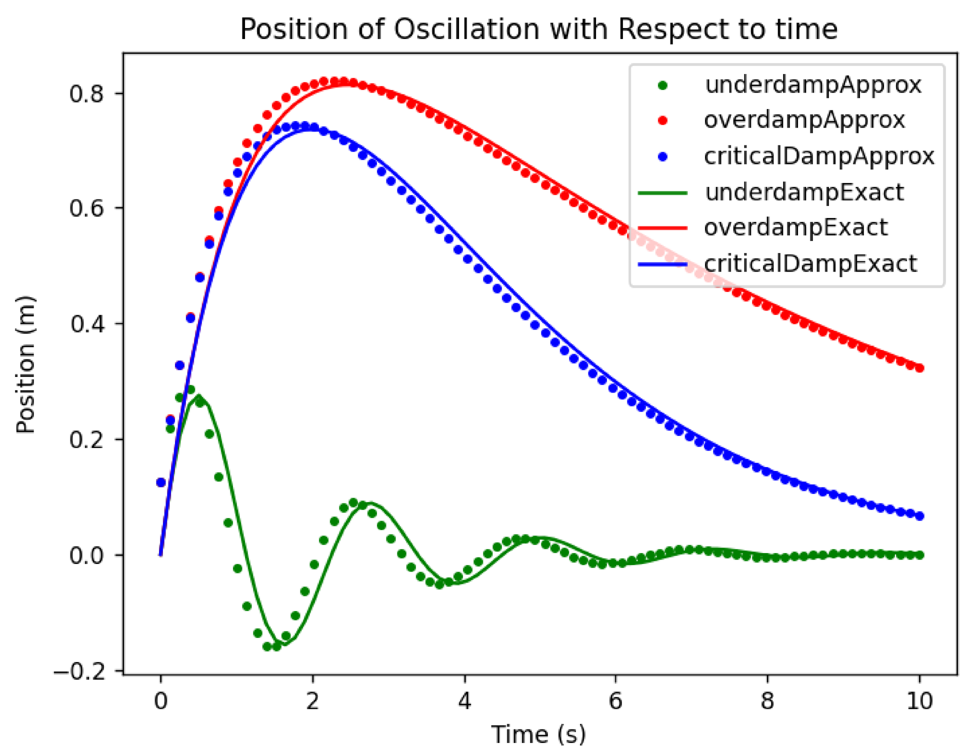
Math Challenge Problem

Ryan M, Hritanshu R, Claire H,
Porfi M, Tejas B



Problem

Assuming the mass is 1 kg, the motion of an object attached to a set of springs on a track given can be modeled by the differential equation:

$x''(t) + bx'(t) + kx(t) = 0$

where b is the dampening coefficient, k is the spring constant, and x(t) is the position of the object as a function of time.

The initial position and velocities were given to be

$x'(0) = 1$

$x(0) = 0$

We need to numerically model the function x(t), for various constants of b and k.

Physical Interpretation

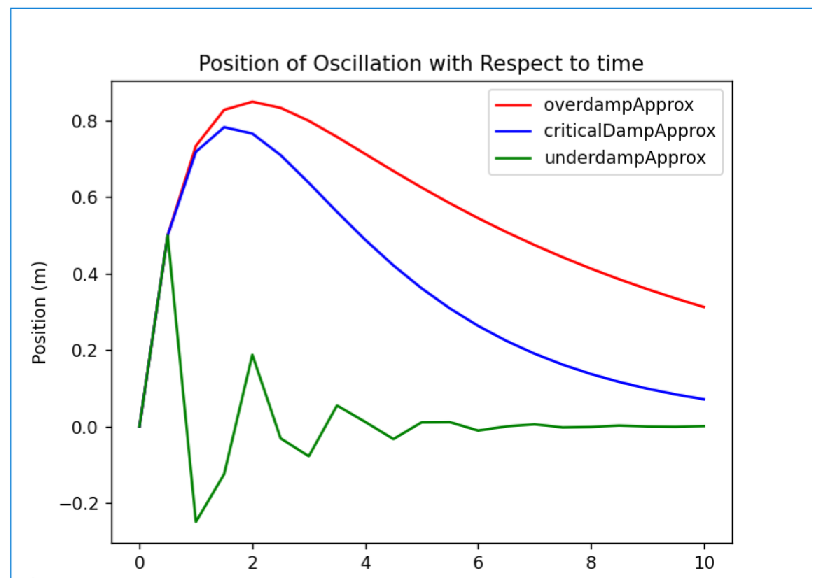
Overdamped motion occurs when $b^2 - 4k > 0$. The motion is slowed down so much that it can't oscillate, and approaches stable equilibrium slowly because of the high dampening force.

Critically damped motion occurs when $b^2 = 4k$ - It approaches equilibrium the fastest possible without oscillating.

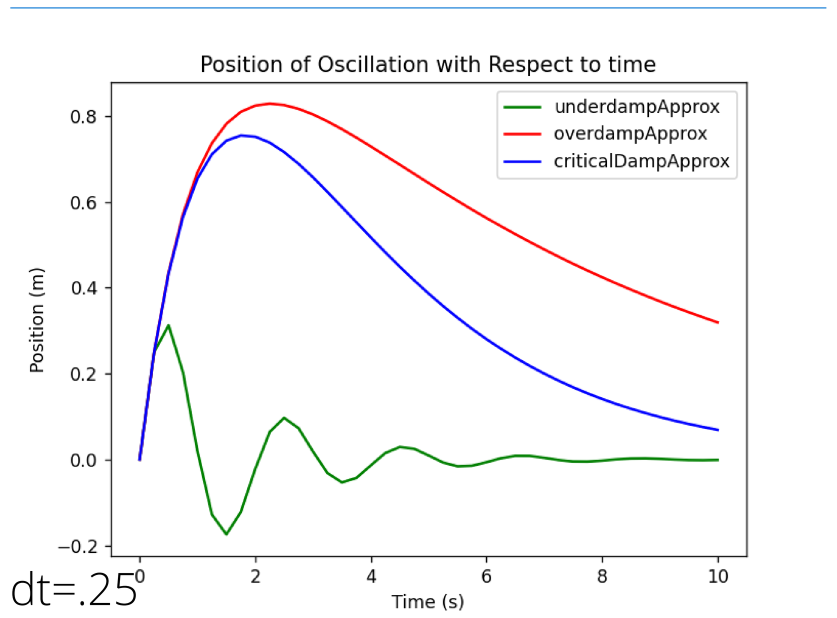
Underdamped: $b^2 - 4k < 0$ - The dampening force is low compared to the restorative force causing it to approach equilibrium so quickly that the object overshoots, starting the oscillatory motion

This ODE can be rewritten as $mx''(t) = -bx'(t) - kx(t)$ which says that the net force is equal to the sum of the dampening and restoring forces, respectively. The dampening force, bx'(t), is proportional to the velocity. The restoring force, kx(t), is proportional to the displacement from equilibrium.

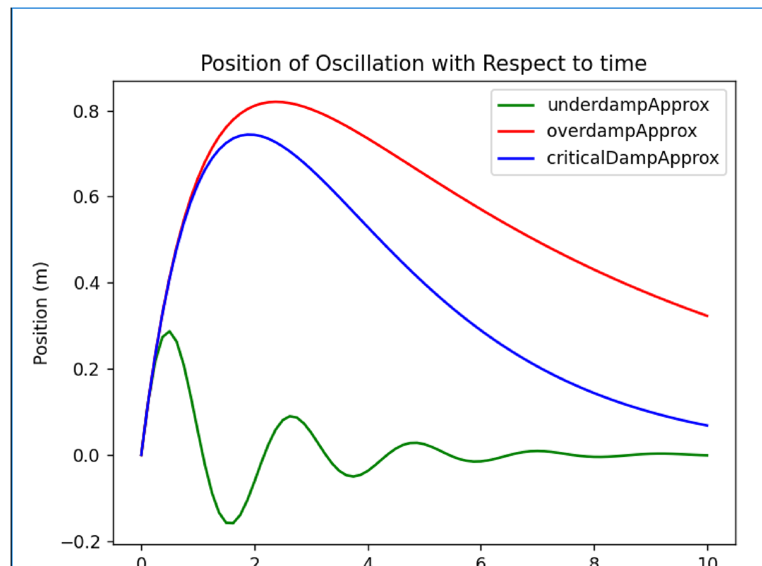
Data and Comparison



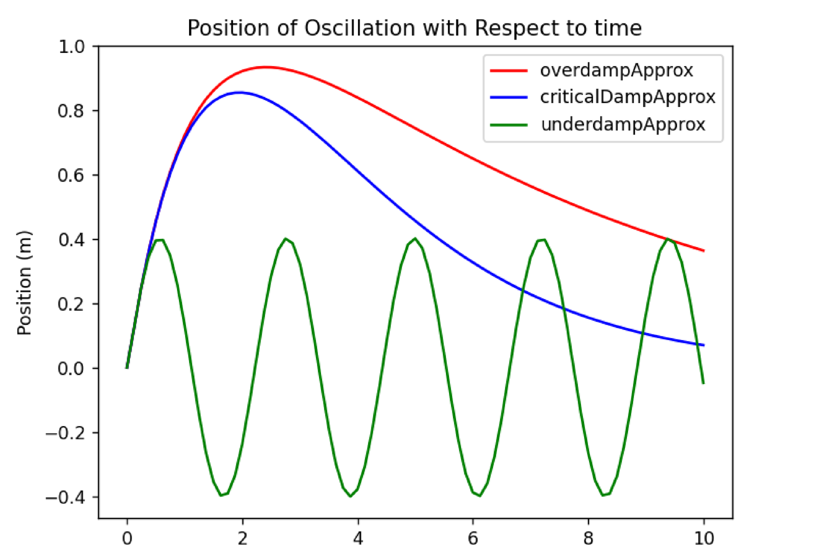
dt=.5



dt=.25

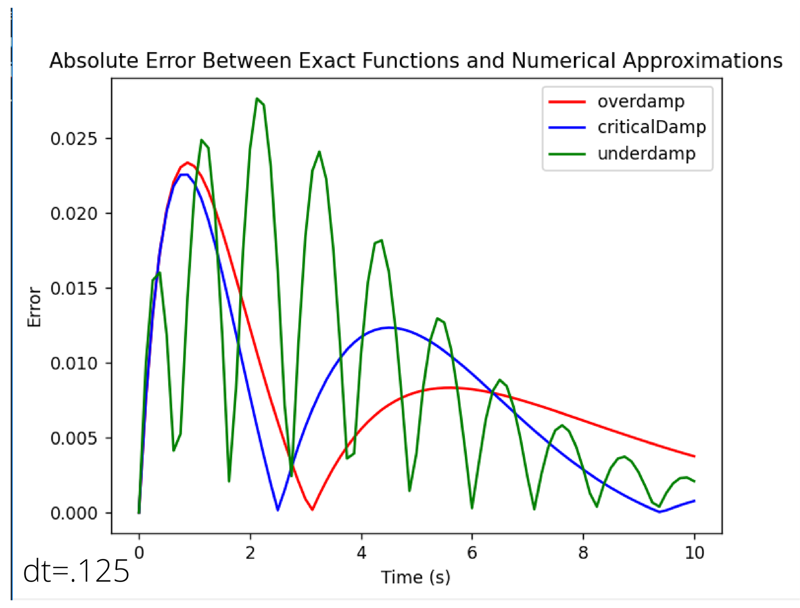


dt=.125

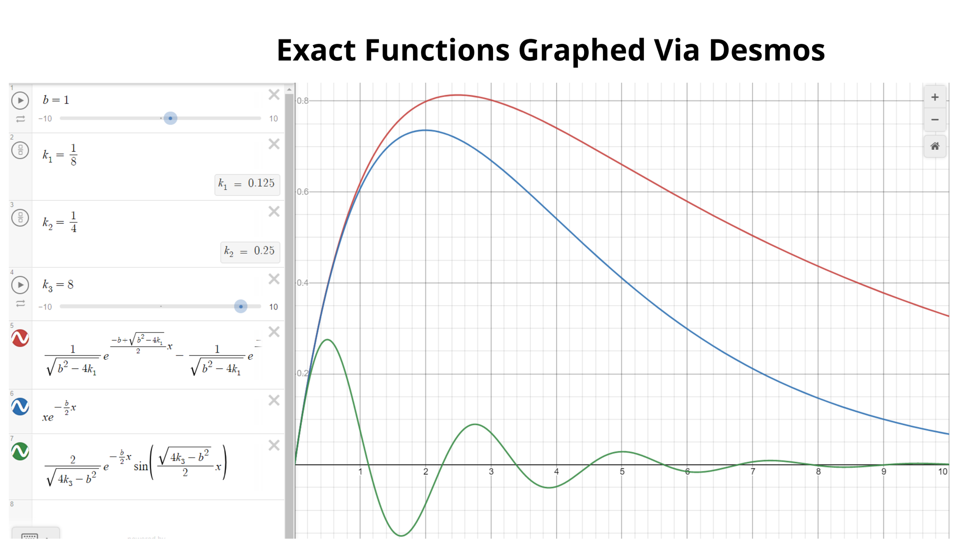


dt=.125

These two graphs illustrate the difference between calculating the acceleration before and after updating the velocity in each iteration of the loop.



dt=.125



Code

```python
import matplotlib.pyplot as plt
import numpy as np

def acc(x, velocity,drag, springconstant):
    return -drag*velocity-springconstant*x
def solve(x,v,b,k):
    t=0
    points = [x]
    while t<10:
        x += v*dt
        v += acc(x,v,b,k)* dt
        points.append(x)
        t+=dt
    return points
def overdamped(t,drag,k):
    r=np.sqrt(np.abs(drag**2-4*k))
    return 1/r * np.exp((-drag/2+r/2)*t)-1/r*np.exp((-drag/2-r/2)*t)
def criticaldamp(t,drag,k):
    return t*np.exp((-drag/2)*t)
def underdamped(t,drag,k):
    r=np.sqrt(np.abs(drag**2-4*k))
    return 2/r * np.exp((-drag/2)*t)*np.sin(r/2*t)

dt = .125
time = list(np.linspace(0,10,81, endpoint=True))

plt.plot(time,solve(0,1,1,1/8), 'r', label='overdampApprox')
plt.plot(time, solve(0,1,1,.25), 'b', label='criticalDampApprox')
plt.plot(time,solve(0,1,1,8), 'g', label='underdampApprox')

overdampeddata = [overdamped(t,1,1/8) for t in time]
criticaldampdata = [criticaldamp(t,1,.25) for t in time]
underdampeddata = [underdamped(t,1,8) for t in time]

plt.plot(time,underdampeddata, color='g', label='underdampExact')
plt.plot(time,overdampeddata, color='r', label='overdampExact')
plt.plot(time, criticaldampdata, color='b', label='criticalDampExact')

plt.plot(time,list(np.abs(np.subtract(solve(0,1,1,1/8),overdampeddata))), color='r', label='overdamp')
plt.plot(time, list(np.abs(np.subtract(solve(0,1,1,.25),criticaldampdata))), color='b', label='criticalDamp')
plt.plot(time, list(np.abs(np.subtract(solve(0,1,1,8),underdampeddata))), color='g', label='underdamp')

plt.legend()
plt.title("Position of Oscillation with Respect to time")
plt.xlabel("Time (s)")
plt.ylabel("Position (m)")
plt.show()
```

This code uses the velocity to change the x position over a small time, dt, and uses the acceleration based on the differential equation to change the velocity over the same small time, dt.

References

3Blue1Brown. (2019, March 31). Differential equations, a tourist's guide | DE1 [Video]. YouTube. https://www.youtube.com/watch?v=p_di4Zn4wz4

Desmos Graphing Calculator. Desmos. (n.d.). Retrieved March 31, 2022, from https://www.desmos.com/calculator

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.

Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.