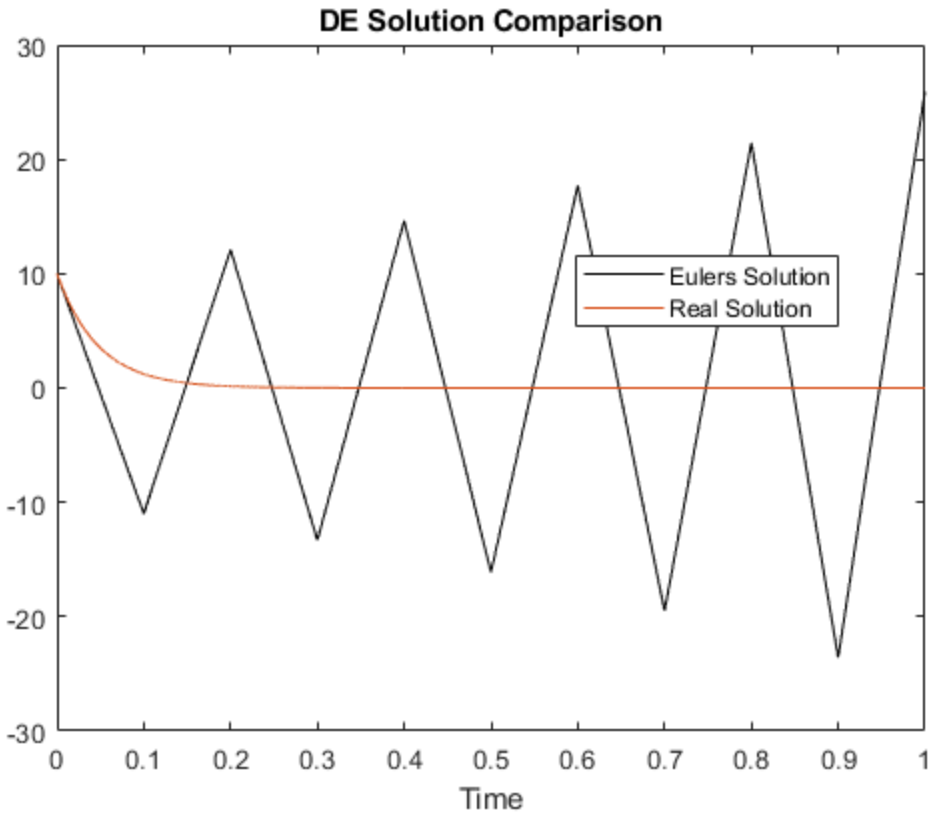
# Table of Contents

3 a) Solve Ivp using Eulers plot along with real solution ..... 1
3 b) Does graph from a approach exact solution? ..... 2
3 c) Find smallest time step to approach exact solution. ..... 2
3 d) Plot Rk4 solution using same timestep found in 3 c ..... 3
4 b) Solve the equations using initial conditions given ..... 4
4 c) Find time where spring no longer reaches distance of 1cm ..... 6

# 3 a) Solve Ivp using Eulers plot along with real solution

```
disp('Question 3a')

%create the de to solve
f = @(t, y) (-21 * y);
alpha = 10;
t0 = 0;
tf = 1;
N = 10;

%calculate solution using eulers with a timestep of .1
[eulerY, eulerT] = euler_timestep(f, t0, tf, alpha, N);

%this is the real solution obtained by solving the de by hand
solution = @(t) (10 * exp(-21 * t));

%obtain data points from solution using really small timestep to be
%accurate
goodTime = 0 : .0001 : 1;
realY = solution(goodTime);

%plot the real solution and eulers on the same graph
figure(1)
plot(eulerT, eulerY, 'black');
hold on;
plot(goodTime, realY);
legend('Eulers Solution', 'Real Solution', 'location', 'best');
title('DE Solution Comparison');
xlabel('Time')
```

```
Question 3a
```




# 3 b) Does graph from a approach exact solution?

```
disp('Question 3b:')
disp('Looking at the graph from part a, the approximation does
 not even come close to the exact solution. This is because of the
 horribly large timestep used on such a small interval.');
```

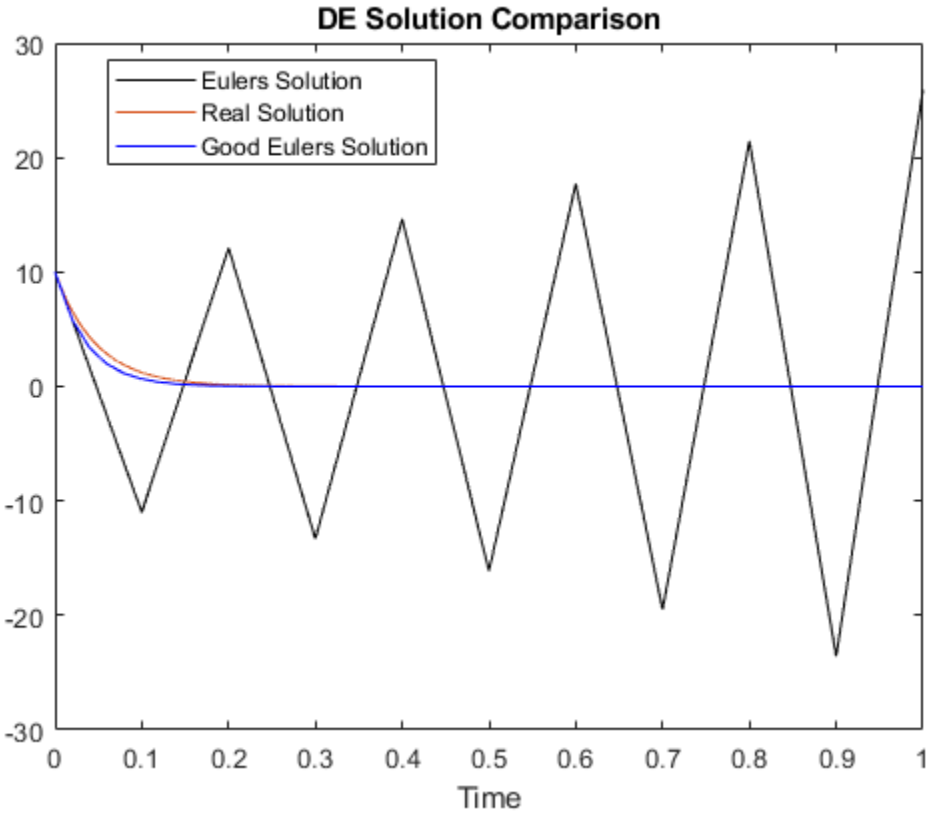
```
Question 3b:
Looking at the graph from part a, the approximation does not even come
 close to the exact solution. This is because of the horribly large
 timestep used on such a small interval.
```

# 3 c) Find smallest time step to approach exact solution.

```
%update n to be really big
N = 50;

%recalculate eulers
[goodEulerY, goodEulerT] = euler_timestep(f, t0, tf, alpha, N);

%plot the new solution on top of figure from part a
```

```
plot(goodEulerT, goodEulerY, 'blue');
legend('Eulers Solution', 'Real Solution', 'Good Eulers
 Solution', 'location', 'best');

disp('Question 3c:')
disp('The Eulers approximation begins to approach the solution and not
 blow up around 50 timesteps. At this point however the approximation
 still struggles to closely approximate the curve from t = 0.1 to 0.2.
 I found this number by slowly increasing the time steps and running
 the code again to look at the graph.')
```

```
Question 3c:
The Eulers approximation begins to approach the solution and not blow
 up around 50 timesteps. At this point however the approximation still
 struggles to closely approximate the curve from t = 0.1 to 0.2. I
 found this number by slowly increasing the time steps and running the
 code again to look at the graph.
```

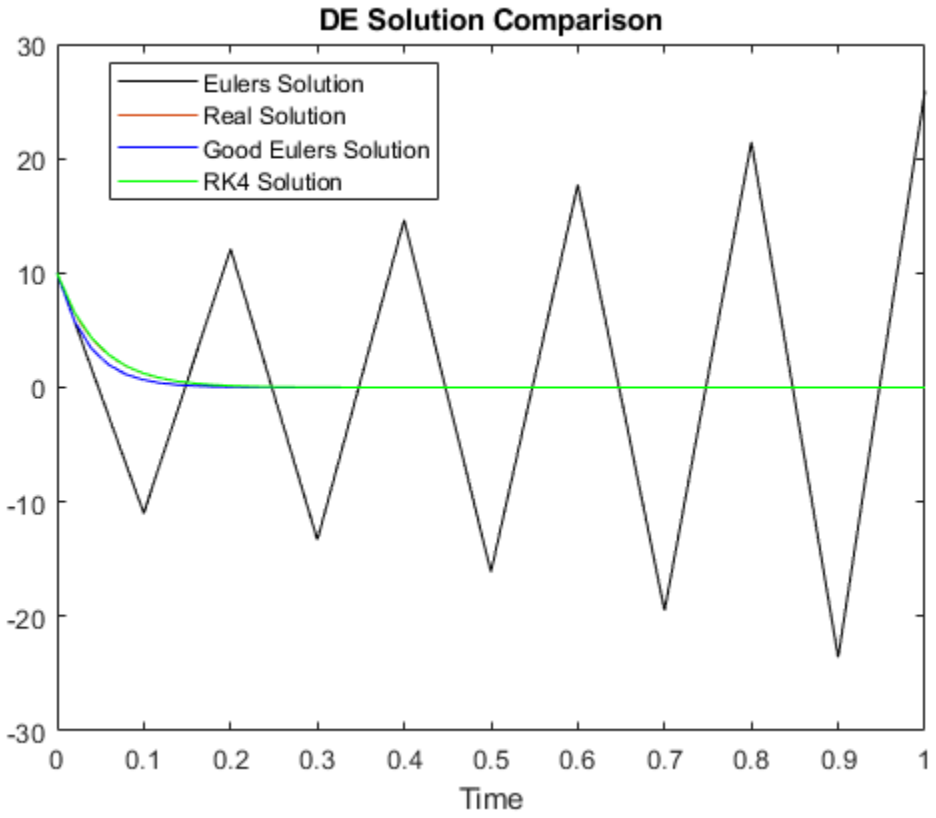



# 3 d) Plot Rk4 solution using same timestep found in 3 c

```
%find the approximation using rk4
[rk4Y, rk4T] = rk4(f, t0, tf, alpha, N);

%plot the approximation ontop of figure from ealrier parts
```

```
plot(rk4T, rk4Y, 'green');
legend('Eulers Solution', 'Real Solution', 'Good Eulers
 Solution', 'RK4 Solution', 'location', 'best');
hold off;

disp('Question 3d:');
disp('The RK4 approximation looks nearly identical to the true
 solution. By far better than Eulers at the same number of timesteps.
 This method performs much better because it is essentially the 4th
 order Taylor method with derivative approximations. Eulers Method is
 essentially a taylor method of order 1, by far worse than RK4.');
```

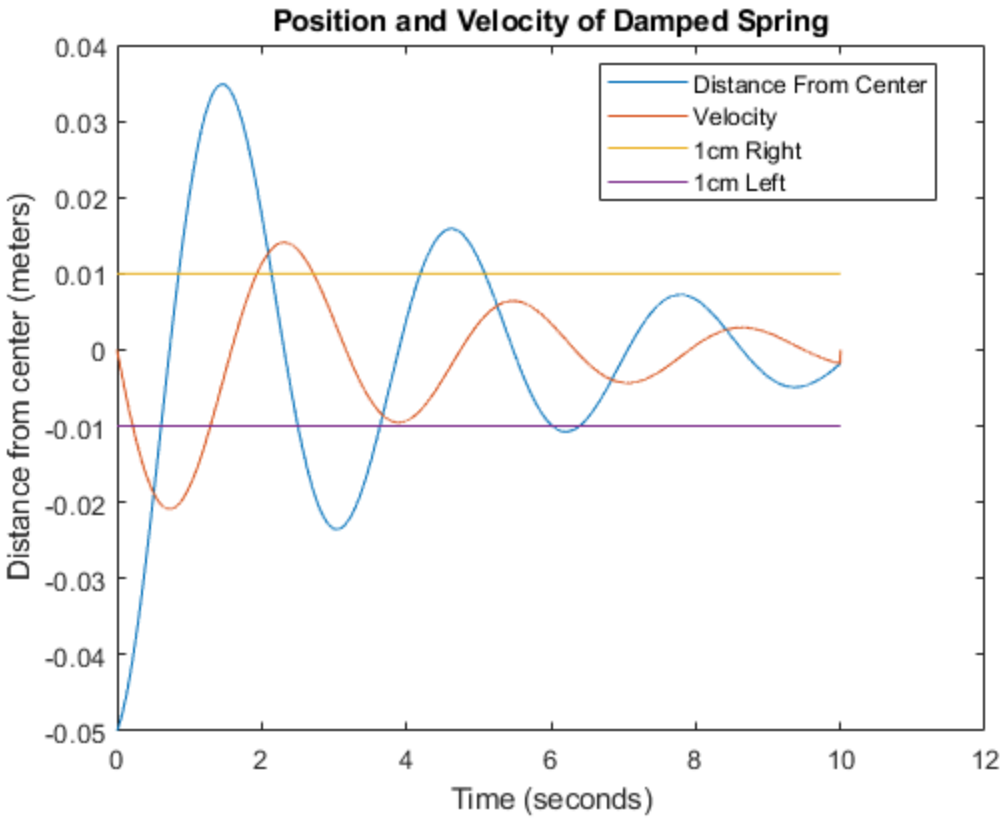
```
Question 3d:
The RK4 approximation looks nearly identical to the true solution. By
 far better than Eulers at the same number of timesteps. This method
 performs much better because it is essentially the 4th order Taylor
 method with derivative approximations. Eulers Method is essentially a
 taylor method of order 1, by far worse than RK4.
```




# 4 b) Solve the equations using initial conditions given

```
%create the differential equations to solve
f=@(t,u,v) (v);
g=@(t,u,v) ((-0.5*v - 4*u) / 1);
```

```
alphas=[0,-.05]; %the first de solves velocity, so it starts at 0,
 from
t0 = 0;          %rest, the second de solves positon, starts at -0.05
 m
tf = 10;
N = 10000;

% solve using the Euler system developed in question 2
[y,t] = euler_system(f,g,t0,tf,alphas,N);

%extract solutions into individual vectors
%the first de solves for velocity, x prime
xprime = y(:,1);
%the second de solves for position from center x
x = y(:,2);

%plot results of x(t)to visualize them
figure(2)
plot(t, x);

%on same figure print a line at 1cm and - 1cm mark
line = zeros(size(t));
line(:) = .01;
negline = zeros(size(t));
negline(:) = -.01;

%plot the 1 cm lines and velocity xprime
hold on;
plot(t, xprime);
plot(t, line);
plot(t, negline);
xlabel('Time (seconds)');
ylabel('Distance from center (meters)');
legend('Distance From Center','Velocity','1cm Right' ,'1cm Left'
 , 'location', 'best')
title('Position and Velocity of Damped Spring');
hold off;
```




# 4 c) Find time where spring no longer reaches distance of 1cm

```
disp('Question 4c:');

disp('From the graph we can approximate that the spring no longer
 reaches 1cm slightly after 6 seconds');

disp('Now lets code to find a more exact solution!');

%because we care about 1cm in EITHER direction, take the absolute
 value of
%x
absDistance = abs(x);

%to find when the spring no longer reaches 1cm then we will iterate
 over
%abs distance, and everytime we detect a change from being less
 than .01 to
%greater than .01 or vice versa then we will save the index, after
 that we
%can just add 1 and find the time at that index

%start with a negative index, as we know this cant exist
```

```
lastChangeIndex = -1;

%iterate over absDistance excluding first and last elements
for i = 2 : length(absDistance) - 1

    %grab values around the current i value
    left = absDistance(i - 1);
    right = absDistance(i + 1);

    %if there is a change in equivalence to .01
    %then we pass over .01 between left and right
    %so save i as the last index to reach .01
    if left < .01 && right > .01 || left > .01 && right < .01
        lastChangeIndex = i;
    end

end

%print out the time at last index + 1 to determine the time
time = t(lastChangeIndex + 1);

fprintf('The time at which the spring no longer reaches a distance
 greater than 1cm from the center is %f seconds.\n', time);
```

```
Question 4c:
From the graph we can approximate that the spring no longer reaches
 1cm slightly after 6 seconds
Now lets code to find a more exact solution!
The time at which the spring no longer reaches a distance greater than
 1cm from the center is 6.395000 seconds.
```

*Published with MATLAB® R2020a*