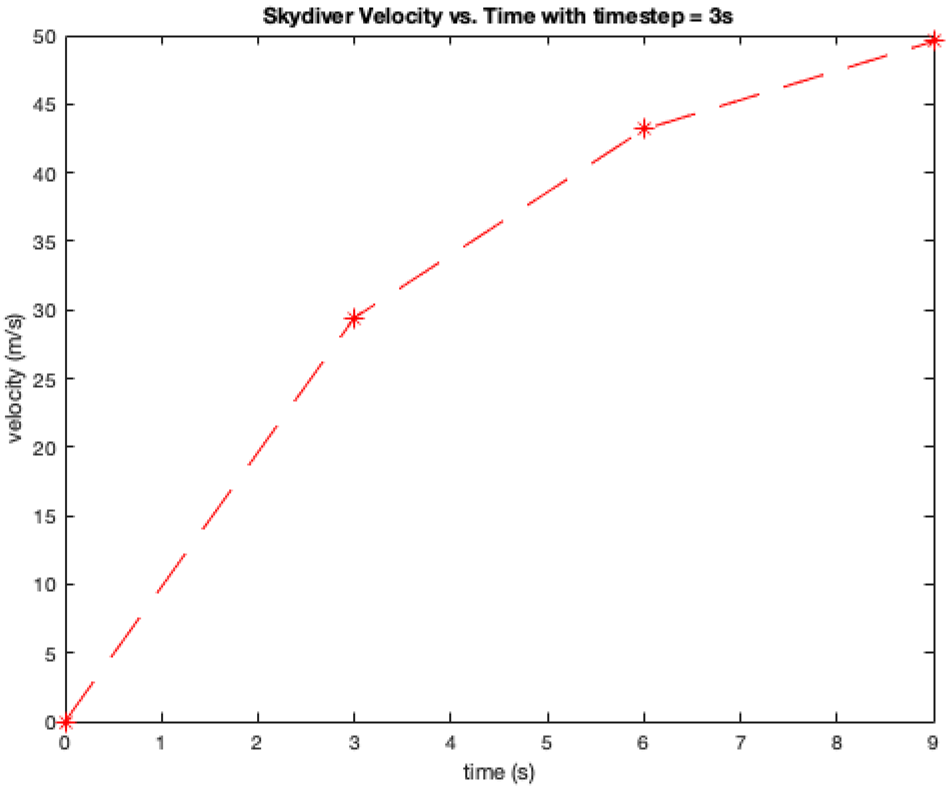
Table of Contents

#1, Part b

```
clear all; clc; close all;

m = 74.8;
c = 13.3;
g = 9.81;
deltat = 3;
t = linspace(0, 9, deltat + 1);
v = zeros(1, deltat + 1);

for T = 1:deltat
    v(T + 1) = (deltat)*(g - c/m*v(T)) + v(T);
end

plot(t, v, 'r*--');
title('Skydiver Velocity vs. Time with timestep = 3s');
xlabel('time (s)');
ylabel('velocity (m/s)');
xlim([0 9]);
```

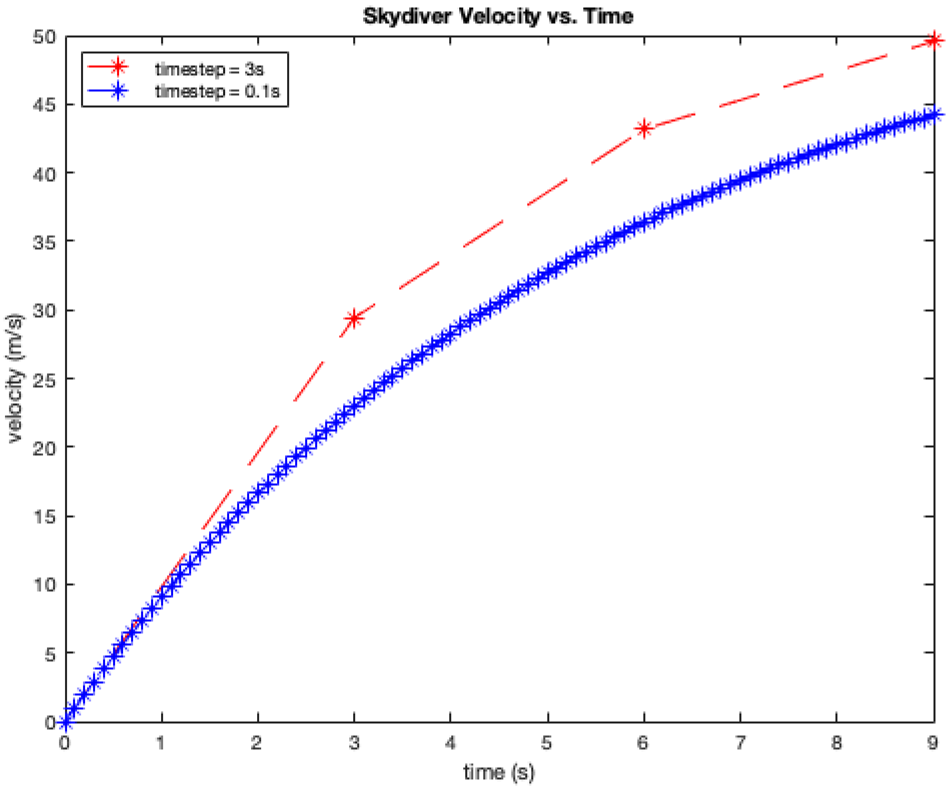


#1, Part c

```
clear all; clc; close all;

m = 74.8;
c = 13.3;
g = 9.81;
deltat = 3;
t = linspace(0, 9, deltat + 1);
v = zeros(1, deltat + 1);

for T = 1:((9-0)/deltat)
    v(T + 1) = (deltat)*(g - c/m*v(T)) + v(T);
end

deltat2 = 0.1;
t2 = linspace(0, 9, (9-0)/deltat2 + 1);
v2 = zeros(1, (9-0)/deltat2);

for T = 1:((9-0)/deltat2)
    v2(T + 1) = (deltat2)*(g - c/m*v2(T)) + v2(T);
end

plot(t, v, 'r*--');
```

```
hold on
plot(t2, v2, 'b*--');

title('Skydiver Velocity vs. Time');
xlabel('time (s)');
ylabel('velocity (m/s)');
xlim([0 9]);

legend('timestep = 3s', 'timestep = 0.1s', 'Location','northwest');

% Percent difference between timesteps:
pd3 = (v(2) - v2(31))/v(2) * 100;
pd6 = (v(3) - v2(61))/v(3) * 100;
pd9 = (v(4) - v2(91))/v(4) * 100;

fprintf('The percent difference at t = 3 is %f\n', pd3);
fprintf('The percent difference at t = 6 is %f\n', pd6);
fprintf('The percent difference at t = 9 is %f\n', pd9);

%{
I would expect the graph of v2, with a timestep of 0.1s, to be more
accurate than the graph of v, with a timestep of 3s.
I would expect the timestep of 0.1s to be more accurate because a
 smaller
timestep better approximates the instaneous derivate of dv/dt that
 this
model is representing. Euler approximations are more accurate the
 smaller
the timestep, as the smaller the time interval, the closer the model
represents an actual derivative.
%}
```

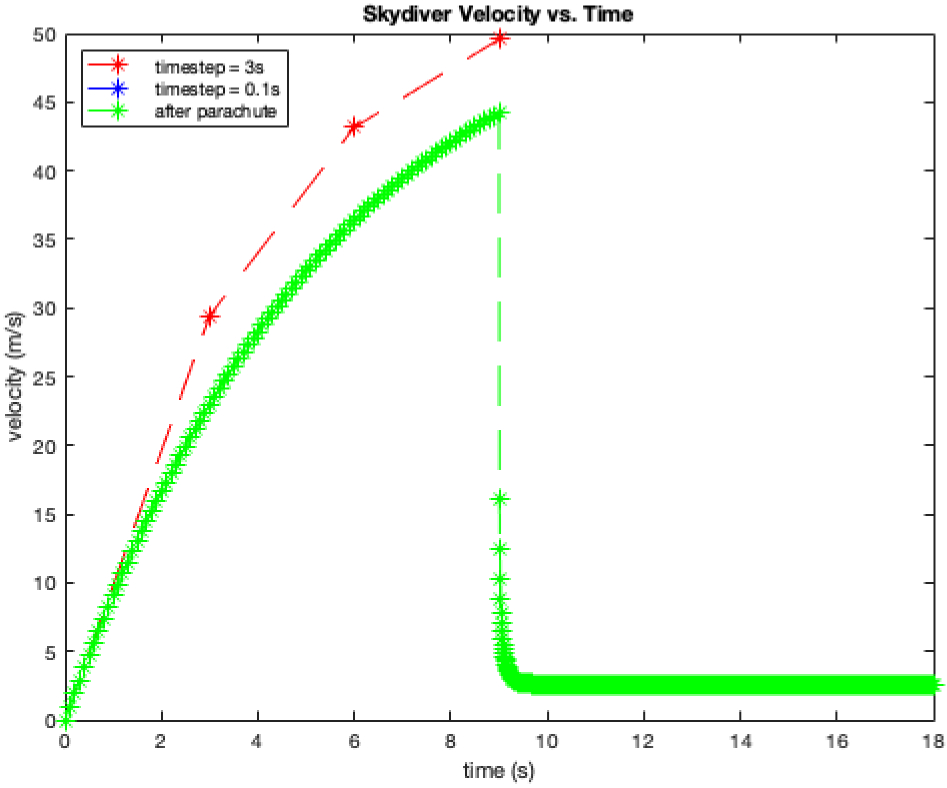
```
The percent difference at t = 3 is 21.972809
The percent difference at t = 6 is 15.736928
The percent difference at t = 9 is 10.839575
```



#1, Part d

```
clear all; clc; close all;

m = 74.8;
c = 13.3;
g = 9.81;
deltat = 3;
t = linspace(0, 9, deltat + 1);
v = zeros(1, deltat + 1);

for T = 1:((9-0)/deltat)
    v(T + 1) = (deltat)*(g - c/m*v(T)) + v(T);
end

deltat2 = 0.1;
t2 = linspace(0, 9, (9-0)/deltat2 + 1);
v2 = zeros(1, (9-0)/deltat2);

for T = 1:((9-0)/deltat2)
    v2(T + 1) = (deltat2)*(g - c/m*v2(T)) + v2(T);
end

c2 = 108.1;
deltat3 = 0.01;
```

```
v3 = v2;
t3(1:91) = t2;
t3(91:91+(18-9)/deltat3) = linspace(9, 18, (18-9)/deltat3 + 1);

for T = 91:(90 + (18-9)/deltat3)
    v3(T + 1) = (deltat3)*(g - c2/m*(v3(T))^2) + v3(T);
end

plot(t, v, 'r*--');

hold on
plot(t2, v2, 'b*--');
plot(t3, v3, 'g*--');
% Note: Since v3 is tracing over v2 for the first 9 seconds, the blue
 graph
% of v2 will not be visible due to the presence of the green v3 graph.

title('Skydiver Velocity vs. Time');
xlabel('time (s)');
ylabel('velocity (m/s)');
xlim([0 18]);

legend('timestep = 3s', 'timestep = 0.1s', 'after
 parachute', 'Location','northwest');

%{
In part d, you should use a timestep of 0.01s vs a timestep of 0.1s,
because it makes the model more accurate by reducing rounding error.
 By
using a timestep closer to an actual derivative function, the result
 is
a more accurate graph.

Additionally, there is also a source of modeling error at play here.
 The
Euler approximation equation we derived in part a first calculates a
 velocity based on
a previous point, and then adds the previous velocity to it. This
 model is
ill-equipped to handle a negative velocity, especially because a
 negative
velocity means the skydiver is suddenly moving upwards (as up is the
negative direction).
A negative velocity at one point will force every single subsequent
 velocity
to also be negative. The first term in the Euler approximation
 multiplies
that negative number by the timestep, which means that with each
 iteration,
the velocity will continue to become more negative. Moreover, the term
that adds the prior velocity is ALSO negative, which means that the
 0.1s
model will continue to be negative. In conclusion, as soon as the
model with timestep of 0.1 seconds encounters a negative velocity, the
entire model plummets continuously, approaching negative infinity.
 Using a
smaller timestep of 0.01s ensures that the velocity changes at a small
enough rate such that the velocity never actually becomes negative.
%}
```



Published with MATLAB® R2020a