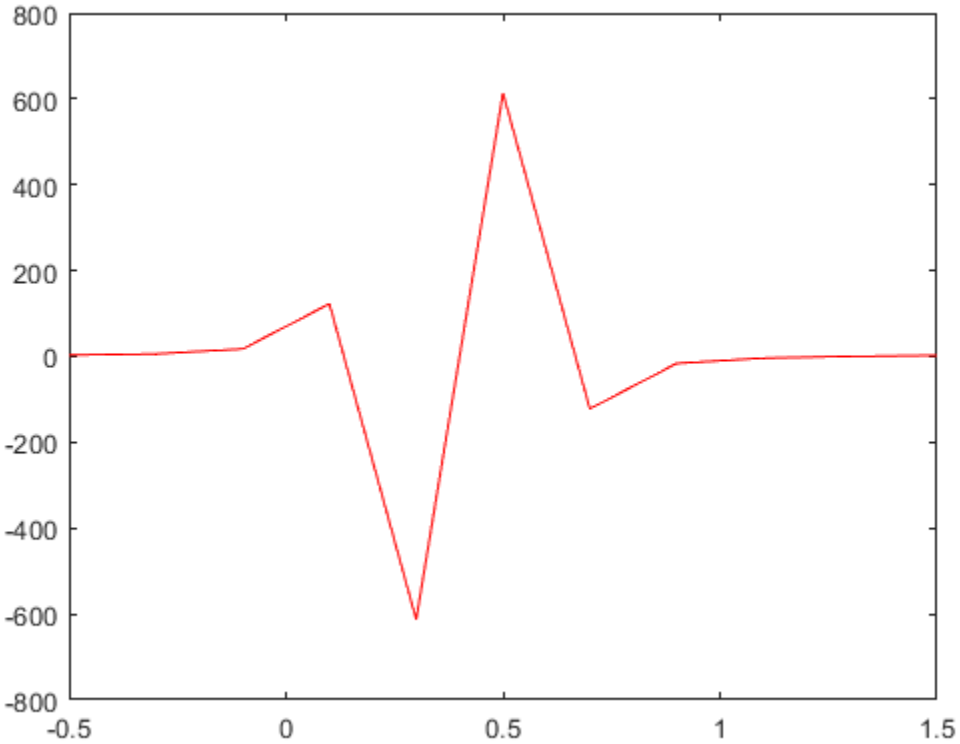
Table of Contents


```
%%%% Differential Equations Homework 3 - Romeo Perlstein %%%%

%%%% -- Problem C13 -- %%%%

dydt = @(t, y) 9.*t - (3./t) * y;

close all
```

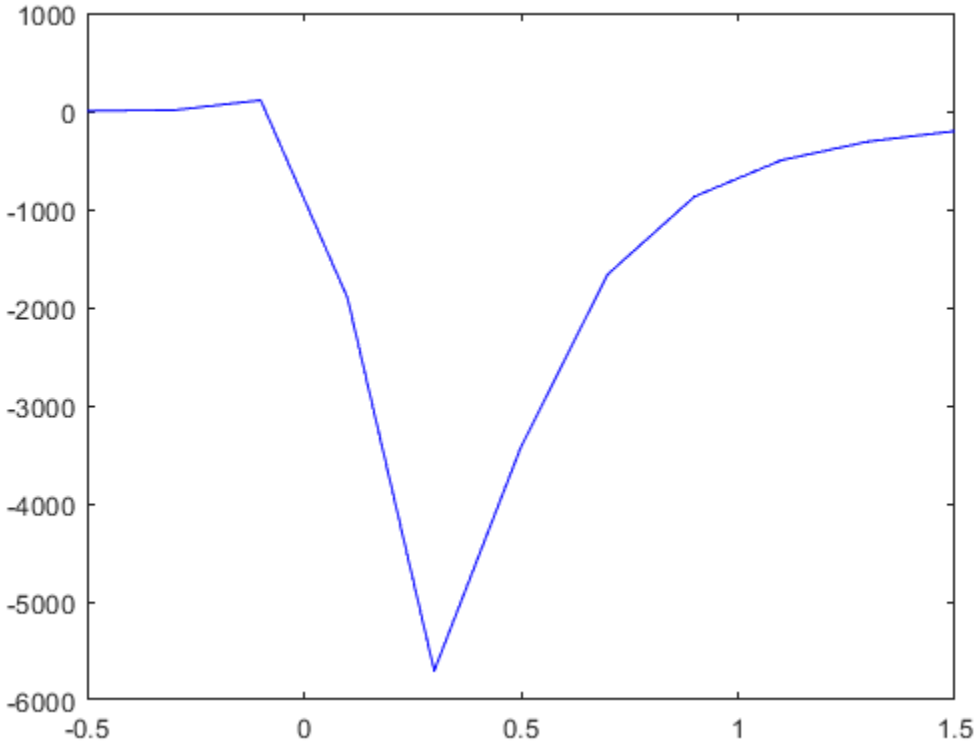
a)

```
[t, y] = myeuler(dydt, -.5, 3.15, 1.5, 10);
plot(t, y, "-r");
```



b)

```
%%%% plot using the "myeuler" funcshion"
[t1, y1] = imprvdmyeuler(dydt, -.5, 3.15, 1.5, 10);
plot(t1, y1, "-b");

% I cannot make sense of my answers, it seems that something weird is
% happening after 0 on the graph, in fact it looks similar to a
% electric signal wave (I think), but I'm not sure. The beginnning of
% each plot is somewhat similar but then it gets all weird after 0
```

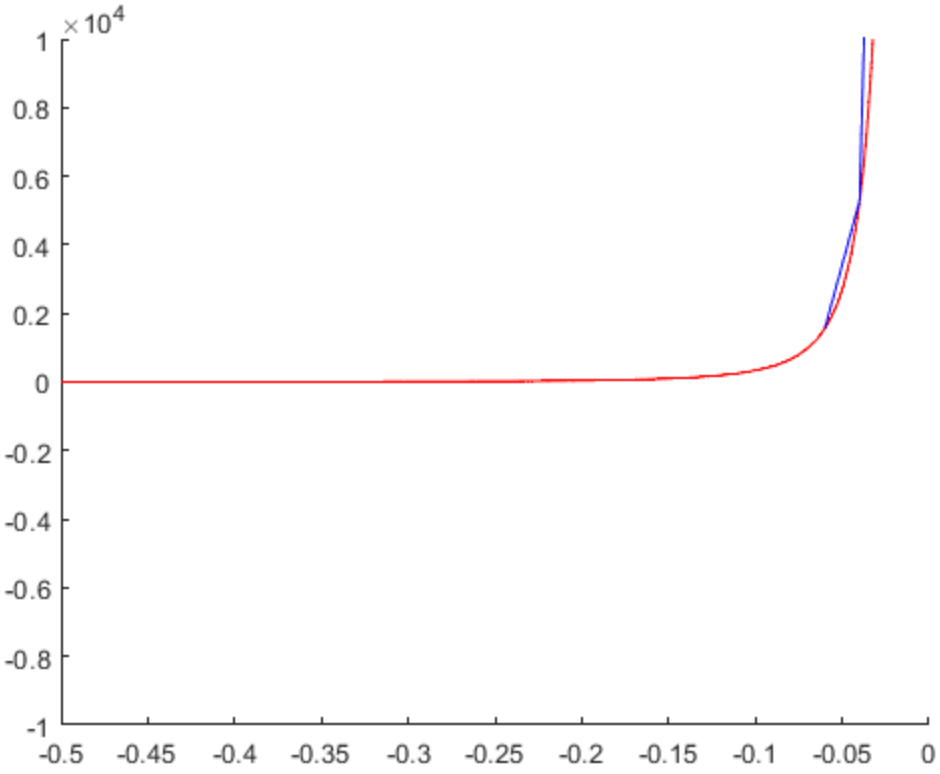


c)

```
[t2, y2] = ode45(dydt, [-0.5:.0001:0.5], 3.15);


figure
hold on
plot(t2, y2, "-r");
ylim([-10000 10000]);

check_length = (.06-(-.06))/.02 +1;
check_value = [-.06:.02:.06];

for ii = 1:1:check_length
```

```
    for i = 1:1:length(t2)
        if(t2(i) == check_value(ii))
            y_check_value(ii) = y2(i);
        end
    end
end

y_check_value()

[t3, y3] = ode45(dydt, [-0.06:.02:0.06], y_check_value(1))
plot(t3, y3, '-b')

% It seems that the approximate solution is only defined on -.5 to 0,
% where is goes to infinity as the limit approaches 0. Plotting
% numbers on the interval -.06:.02:.06 results in consistant behavior
% with this idea as well;
%
% as t approaches 0, the y values approach infinity, just like the
% approximate function

hold off
```

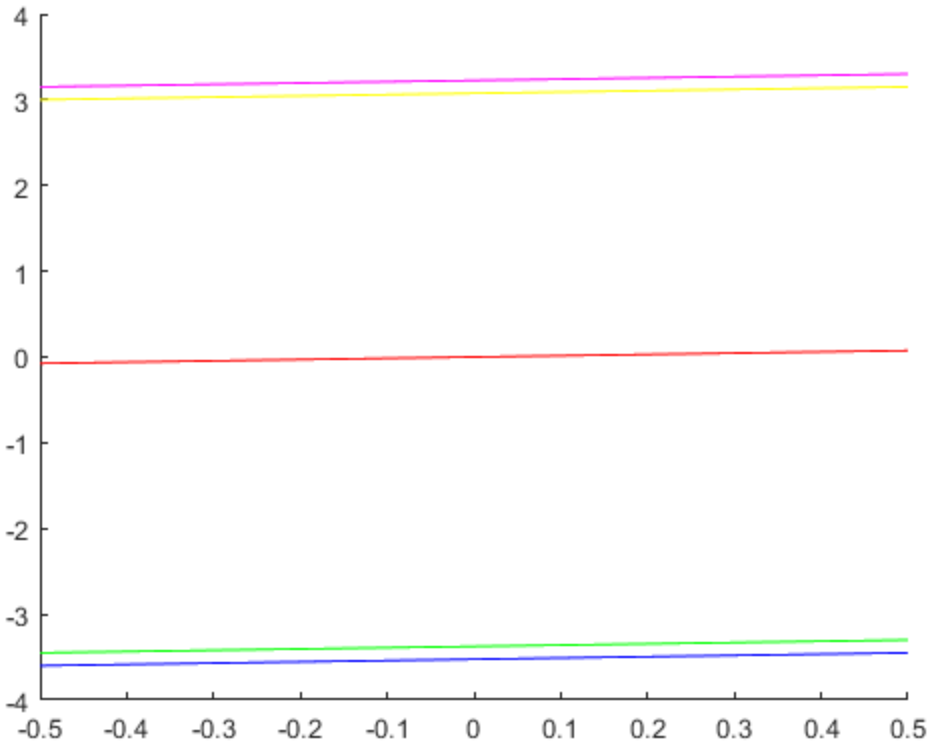
```
ans =

   1.0e+03 *

    1.5617         0         0       NaN         0         0       NaN


t3 =

   -0.0600
   -0.0400
   -0.0200
         0
    0.0200
    0.0400
    0.0600


y3 =

   1.0e+04 *

    0.1562
    0.5275
    4.2230
       NaN
       NaN
       NaN
       NaN
```



d)

```
%%%% Solve the ODE at different initial values (AKA find the C values for
%%%% each initial value - I think)
% y(0) = 0, y(-.5) = 3.15, y(.5) = 3.15, y(-.5) = -3.45, y(.5) = -3.45

% ty' + 3y -9t^2 = 0
% ty' + 3y = 9t^2
% y' + 3/t * y = 9t
% I.F. = e ^ 3ln(t)
% I.F, = t^3
% dy/dt (t^3 * 3/t * y) = t^3 * 9t
% 3t^2 *y = int(9t^4)
% 3t^2 * y = 9/5 * t^5

%%%% General solution:
% y = (3/5) * t^3 + C
y_ex = [0, 3.15, 3.15, -3.45, -3.45];
t_ex = [0, -.5, .5, -.5, .5];
for i = 1:1:5
    C(i) = y_ex(i) - (3/5)*t_ex(i)^3;
end
figure
hold on
t_plot = [-0.5:0.5];
```

```
y_plot1 = (3/5).*t_plot.^3 + C(1);
y_plot2 = (3/5).*t_plot.^3 + C(2);
y_plot3 = (3/5).*t_plot.^3 + C(3);
y_plot4 = (3/5).*t_plot.^3 + C(4);
y_plot5 = (3/5).*t_plot.^3 + C(5);

plot(t_plot, y_plot1, "-r")
plot(t_plot, y_plot2, "-m")
plot(t_plot, y_plot3, "-y")
plot(t_plot, y_plot4, "-g")
plot(t_plot, y_plot5, "-b")

% I dont think we could have known to expect meaningful data or not in parts
% a-b. Our manual approximation did not account for the function to approach
% infinty, so it did not really know what to do when it reached that
% and as such caused some weird behavior to occur after passing
% t=0. I believe ode45 solves this issue by account for limits in it's
% solution, which lets it understand that if a function has a limit, the
% values of y after a certain t value are going to be a specific thing,
% either infinty, -infinity, or a number, and it is then able to plot that
% correctly.
%
%
% To preface, after spending a day thinking about the outputs that were
% apart of this problem, I know realize what was going on. Using my Euler,
% we were able to approximate the outputs of the ODE until it reached 0,
% where, rather than illustrating the approach to infinity, it starts to
% act weird and inconsistant behavior wise. This issue is also prevalent in
% the improved Euler approximation, where our approxiation when t<0 is
% better, but we still are unable to correctly graph the solutions as it
% approaches 0 and beyond. ode45 solves this behavior by correctly solving
% for and understanding that the function approaches infinity as the limit
% goes to zero. With this being the case, I don't believe parts A and B
% would've provided meaningful data without the knowledge that both Euler
% methods are unable to accurately describe the function as it approaches
% 0. It's rather cool to see the differences between each method, however
```



Published with MATLAB® R2022b