<u>Problem:</u>

A jet fighter's position on an aircraft carrier's runway was timed during landing:

***t*, s**	0	0.52	1.04	1.75	2.37	3.25	3.83
***x*, m**	153	185	208	249	261	271	273

where *x* is the distance from the end of the carrier. Estimate (a) velocity (dx/dt) and (b) acceleration (dv/dt) using numerical differentiation.

For this problem I want you to use *forward*, *central*, and *backward* finite difference approximations with an error of order h^2. I want you to use a function program to calculate the derivate using the equations for a first order derivative. To calculate the acceleration you are calling the same function program you used to calculate the velocity but you are sending it velocity and time as inputs instead of position and time.

Because the time measurements of position are unevenly spaced we are going to use MATLABS ***spline*** function to determine position (*x*) at 30 evenly spaced points. To do this use ***linspace*** to generate 30 points for time starting at a time of 0s and an end time of 3.83s. Note that this is the xx vector you will use for the spline function.

• spline function:

yy=spline(x, y, xx)

- performs cubic spline interpolation
- x & y contain the original data
- xx are the interpolation locations
- yy are the function values at xx

<u>Requirements:</u>

1. Write a main program and a function program.
2. Main Program
    a. Define vectors for time and position.
    b. Define 30 points for time
    c. Use *spline* function to generate 30 points for position
    d. Call function program to determine velocity
    e. Call function program to determine acceleration
    f. Use subplot to generate a single figure window containing a graph for the velocity and a graph for the acceleration as shown below.
3. Function program
    a. Inputs are the vectors for the independent and dependent variables
    b. Output is a vector for the 1^{st} order derivative for the dependent variable

c. **Use *forward* finite differences for 1st point, *backward* finite differences for last point, and *central* finite differences for all other points. Use formulas with an error of order h^2.**

d. Function program should be generic enough to handle input vectors of any size. You don't have to worry about the vectors being too small to calculate the derivative.