INTERPOLATION USING MATLAB

MATLAB provides many functional ways to do interpolations in data tables and curve fitting. We will explore a few here. Warning, these type of functions change for different versions from time to time.

LINEAR INTERPOLATION

% Reminder what is linear interpolation?
Given a set of data points x, y usually obtained via some experiment, we can show that a point within the data set can be obtained by using a straight line approximation (the simplest approximation).
Typically if we have two points namely y1, x1 and y2, x2 and we would like
to know the value of y=f(x) at a value of x lying between x1 and x2 it is not difficult to show that from the following illustrated table

X'S	Y 'S	algebra
1.2	34	X1, Y1
2.3	Y?	X, Y(F(X))
3.9	67	X2, Y2

Linear interpolation formula is the following to get the unknown y value.

Y = F(X) = Y1 + (X-X1)(Y2-Y1)/(X2-X1)

In Matlab we can do this task by simply setting up the x and y values in two corresponding Vectors for each table. And use the function interp1()

```
>> help interp1
   % current version info reads
  Vq = interp1(X,V,Xq) interpolates to find Vq, the values of the
  underlying function V=F(X) at the query points Xq. X must
  be a vector of length N.
  % there are a lot of other options using this function as in other advanced Matlab functions.
```

Consider the following table which represents wind tunnel data with Settings, s, that determine, the velocity of the wind, v, (<0 is reverse).

s	v
-4	-0.202
-2	-0.502
0	0.108
2	0.264
4	0.421
6	0.573
8	0.727

Visualizing the interp1 function for our goal of expressing the above algebra solution for y, it looks like

y= f(x)= interp1(x_vector, y_vector, x)

For the wind tunnel if we want the value of the velocity v at s=1.5 we set up two vectors(1 D array) for the data with 7 components each as follows

```
>>s= [ -4 -2 0 2 4 6 8];
>>v=[ -0.202 -0.502 0.108 0.264 0.421 0.573 0.727];
```

the interpolated value we want

```
>>x=1.5
>> y= interp1(s,v,x)
y = 0.225
```

You can see this answer makes sense looking at the table of values of s vs v.
If we need for future use the data in the vectors we can save them as follows in the file

windtunnel.mat.
save ('windtunnel','v','s')
and retrieve them with
load windtunnel

The closer the original data lies on a straight line or the finer the interval between point then the better the answer for the above linear technique.

We will now look at more accurate techniques for data that deviates from straight line behavior.

POLYNOMIAL INTERPOLATION

% In scientific and engineering work we often have a set of measurements
from which we wish to draw conclusions. One aspect of our interpretation
of the measurements is a curve to approximate the data we have on hand.
We saw above that given a set of values we could use a Linear Interpolation
to estimate intermediate values. In many cases the linear method is not very
accurate because the behavior of many systems are not linear. In this
situation we use polynomial methods which usually means we fit a polynomial
of a degree that closely matches the graphical trend of our data.
The resulting curves and/or formulas can predict intermediate values.
We will now discuss several approaches to obtaining such data fitting.

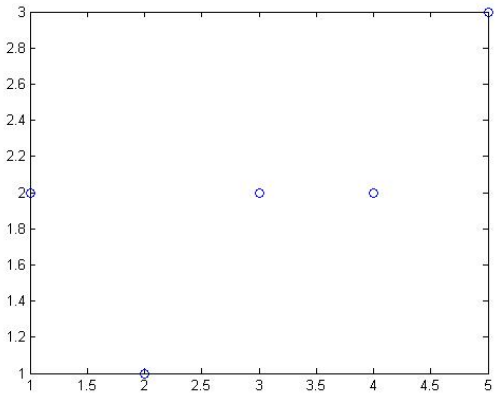
CUBIC SPLINE INTERPOLATION

% Third order polynomials used to fit the intervals between our data points
are known as "splines". In MATLAB they are obtained as in the following
illustrative example.
Given the follow tabulated values which may have arisen from some experiment

X values	Y values
1	2
2	1
3	2
4	2
5	3

% We set this data into two vectors, namely
x =[1 2 3 4 5]
x = 1 2 3 4 5
» y=[2 1 2 2 3]
y = 2 1 2 2 3
% plotting the experimentally determined points only and marking them with
'o' we get with

» plot(x,y,'o')



% One sees the points here but since the plot routine automatically scaled our values we have the axis going through the points. It is helpful for visualization to re-scale the axis with the 'axis' command

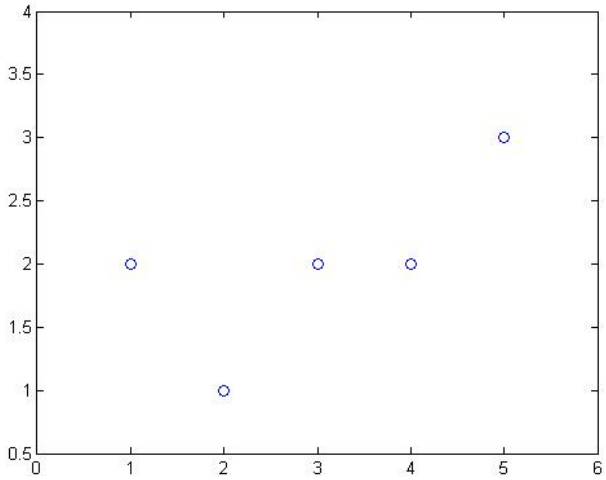
» help axis

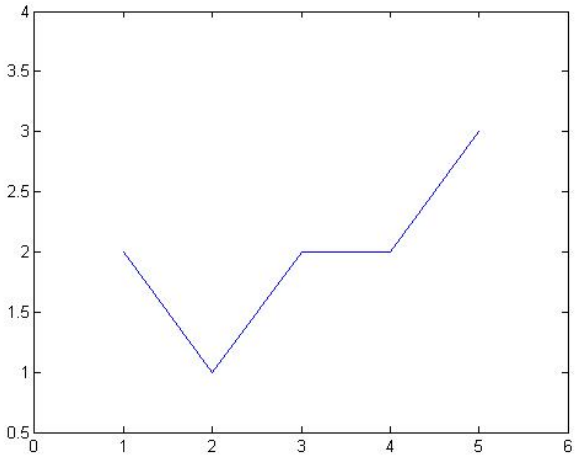
AXIS Manual axis scaling on plots. Typing AXIS by itself freezes
the current axis scaling for subsequent plots. Typing AXIS
again resumes auto-scaling. AXIS returns a 4 element row
vector containing the [x_min, x_max, y_min, y_max] used on
the last plot.
AXIS(V) where V is a 4 element vector sets the axis scaling
to the prescribed limits.....

% From the above definition we see we can set up the four element vector
v=[x_min, x_max, y_min, y_max]
to change the range of the graph. If we
change our x value range to go from 0 to 6 and y range .5 to 4 by setting up the vector 'v' to be

» v=[0 6 .5 4]
v = 0 6.0000 0.5000 4.0000
The scaling is changed with the axis command

% when we plot the points again we get better looking distribution
» plot(x,y,'o')
>>**axis(v)**





If we use the basic plot we get the curve to the left
>> plot(x,y)
>> axis(v)

% And we see that the basic 'plot' function uses a linear interpolation to fill the
interval between the data points. It should be clear from the figure that a straight line or series of line segments do not represent the true shape or behavior of the data or function (possibly a natural law) that may be behind the generation of the values.

THE STRATEGY OF CUBIC SPLINES

% The "cubic spline" that we will generate to connect the data points consists of third degree polynomials. The interval between each point is fitted with a Third degree polynomial of the form

$$y = Ax^3 + Bx^2 + Cx + D.$$

Each polynomial, ONE PER INTERVAL, requires finding the 4 coefficients. Or, in other words, each interval must have values of A,B,C and D to specify the curve connecting the data points on each side of the interval. The full technique to solve for all the polynomials requires that we generate enough equations to solve for 4 unknowns per pair of data points. We do this in standard programming techniques (e.g. FORTRAN) by requiring the following conditions which set up enough equations to solve for all unknowns.

1. The curves match at each point.. The values of the two polynomials on each side of the point be equal at the point.
2. The slopes of the curves match at the data points. This is done by requiring that the first derivatives of the "splines" match at the data points.
3. We require the "curvature or bending" of the curves which is found from the second derivative to match at the boundary points.
4. The end points leave an open ambiguity so they are traditional treated to have the curve go straight at these points or equivalently have the second derivative be equal to zero at the end points.

These last 4 conditions give rise to systems of equations that can be written in matrix form and then by using Matrix rules in our programming
code we can get 4 polynomials to match the above example of 5 data points.

% MATLAB permits us to solve for the "spline" curve in a relatively simple approach when we call upon the **spline()** function.
» help spline

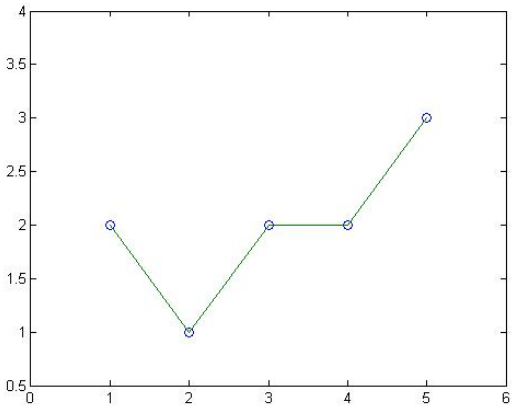
SPLINE Cubic spline data interpolation.
Given data vectors X and Y, and a new abscissa vector XI, the function YI = SPLINE(X,Y,XI) uses cubic spline interpolation to find a vector YI corresponding to XI that is a fine division of the x axis.

% Given our original problem
I

% we might initially try

Rough curve is linear interpolation!
» xi=x
xi = 1 2 3 4 5
» yi=spline(x,y,xi);

% Plotting the values of our spline "yi" with the original x,y points as in
» plot(x,y,'o',xi,yi)
>>axis(v)



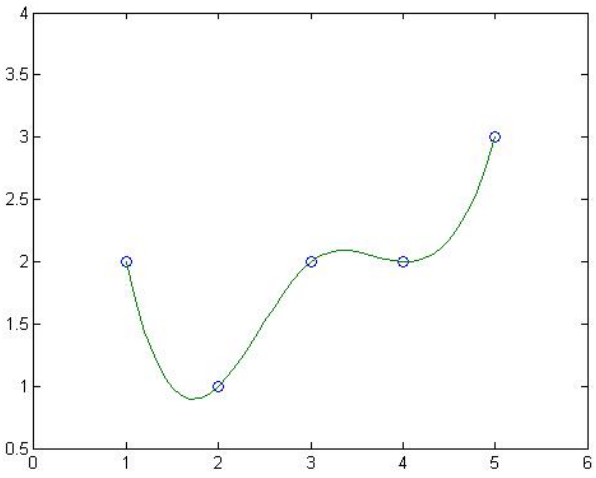
% We see by the last curve we wound up with an almost linear fit between the points of our data because we used too few points to generate our spline.

% We can increase the smoothness of our fit by choosing more points in the range by increasing the number points xi used to find the splines yi. As follows

» xi=1:.1:5;

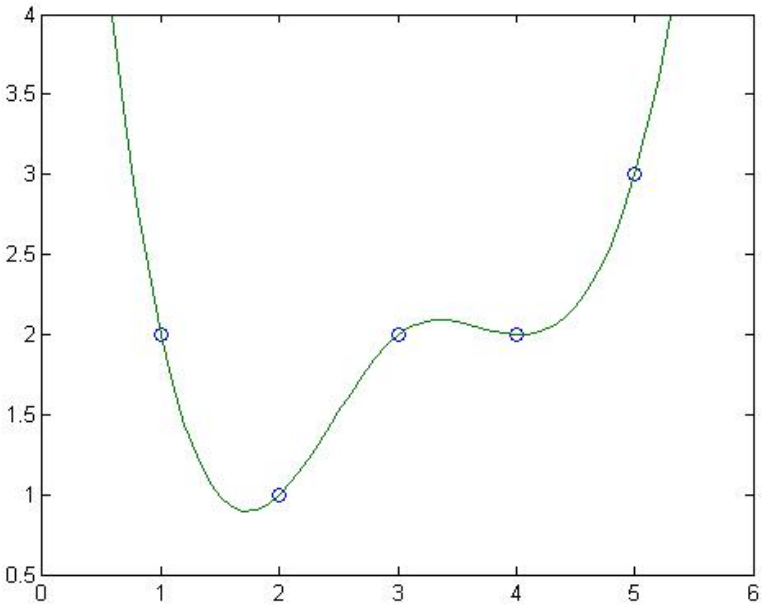
» yi=spline(x,y,xi);

» plot(x,y,'o',xi,yi)
>>axis(v)



% We get a much smoother curve and see what a spline is.
To try to improve on the last curve we can increase the xi points and to see the linear behavior at the end points (condition 4 above) we extend the range of xi.
% we proceed as follows to carry this out.

» xi=0.5:.01:5.5; NOTE: RANGE IS NOW 0.5 to 5.5
» yi=spline(x,y,xi);
» plot(x,y,'o',xi,yi)
>> axis(v)



POLYNOMIAL INTERPOLATION WITH LEAST SQUARE-METHODS

% In the "spline" case we had curves for a graph and could estimate intermediate values from the plot. We also need a way of expressing a 'formula' for our data set which can match our data points and also be a reasonable representation for the intervals between our data points. We also have the problem that when we measure physical processes the data we obtain may usually have some error in it.
So we use techniques to generate a curve that goes through the points in a best fit manner. One technique that minimizes the distances of the polynomial generated from each of the known data point is called the **Least Square method.**

% MATLAB provides a few functions that will permit us to calculate a least square fit to a known set of data. First is "polyfit"
» help polyfit
POLYFIT POLYFIT(x,y,n) finds the coefficients of a polynomial formed from the data in vector x of degree n that fits the data in vector y in a least-squares sense.

% So we see "polyfit" requires us to put the known data in the vectors x and y and we have a choice of determining the type of curve to generate by choosing the degree n of the polynomial.
% In other words we get a..
line (linear fit) with n=1
parabola with n=2
cubic with n=3
.....etc
nth degree polynomial with a value of n

% Polyfit gives us the coefficients of the polynomial in a vector form.
for example:
» x = 1 2 3 4 5
» y = 2 1 2 2 3
» A=polyfit(x,y,4)

A = 0.2083 -2.5833 11.2917 -19.9167 13.0000

% we asked for a 4th order polynomial and we got the coefficient that satisfy the 4th degree polynomial equation

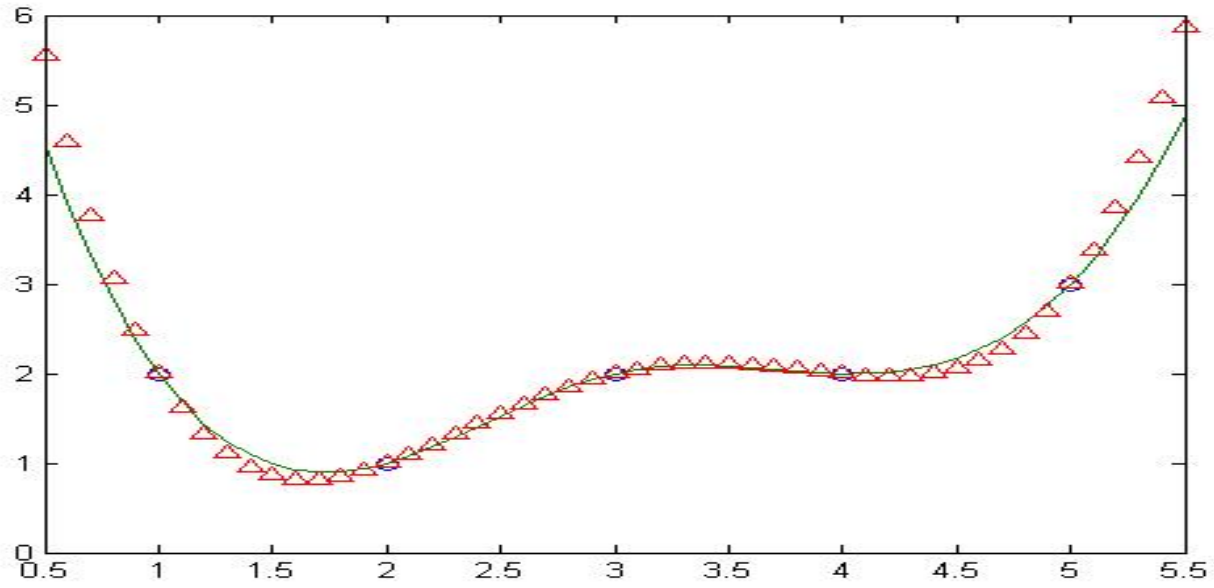
% $y = 0.2083 x^4 - 2.5833 x^3 + 11.2917 x^2 - 19.9167 x + 13.000$

% Instead of constructing a MATLAB function for the 4th order polynomial with the above 'A' coefficients as we did before we can generate the 4th order points with the "polyval" function
» help polyval
POLYVAL Polynomial evaluation.
If V is a vector whose elements are the coefficients of a polynomial, then y=POLYVAL(V,s) is the value of the polynomial evaluated at s. if s is a vector than y is a correspond vector solution for each point of s,

% We pick a range of x as we did before to generate the 4th order equation
We will use a lot of x values (xi values) to avoid a linear look
» xi=.5:.01:5.5;
% We pass the coefficients to 'polyval' through the vector 'A' and get the y values of the 4th order polynomial by
» y4=polyval(A,xi);

% We can also caluclate the spline fit for comparison
» yi=spline(x,y,xi); A continuous line in the next graph.

% We can see the results by graphing the data points ,the spline and 4th order polynomial with
» plot(x,y,'o',xi,yi,xi,y4,'^') The Polynomial solution will be a graph of triangles here.



We see by this last plot that the resulting 4th order curve (marked with triangles) closely follows the spline solution.
% If we need the value of y for some value of x we can just use the "polyval"
function to solve for this intermediate point. simply y =polyval(A,x).

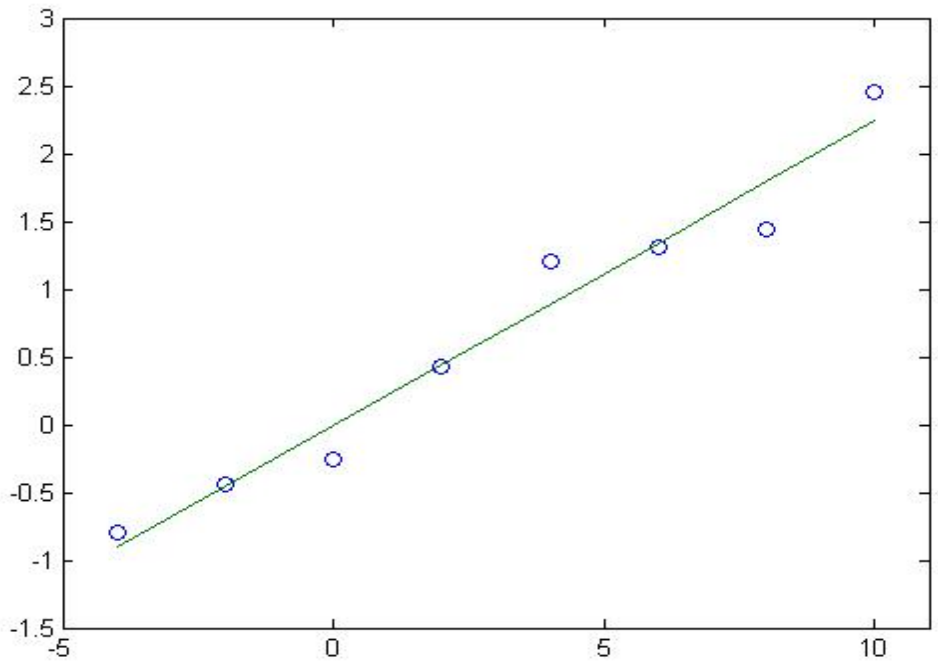
LINEAR CASE n =1

% If the trend of the data looks linear which does happen a lot we use the least square technique to find the best straight line to fit the data
that is, we are looking for the curve y =A(1)x +A(2) which is generated with
the polyfit function
consider the data set
» x = [-4 -2 0 2 4 6 8 10]
» y =[-0.795 -0.436 -0.261 .425 1.20 1.313 1.446 2.458]

% We generate the straight line coefficients with
» A=polyfit(x,y,1)
A = 0.2243 -0.0041

% We generate the straight line that best fits the data with
yl =polyval(A,x);
Expand the axis to get a view of all points
v = [-5.0000 11.0000 -1.5000 3.0000]

% How well this fits can be seen in a plot
» plot(x,y,'o',x,yl)
>> axis(v)
% The results you see are very good. The straight line goes through the points in a way that minimizes its distance from each point.
%



LABORATORY TASKS

. GIVEN THE FOLLOWING SET OF X, Y POINTS

X	Y
2.0	7.5
4.5	7.1
5.5	6.0
7.0	5.9
8.0	4.3
9.5	3.5
11.0	3.2
12.0	1.1

35.(5pts) Create the vectors and save in a “mat” file. Set up a program that reads the mat file and uses a linear interpolation to find a value in this table when a user enters a value of x they want to find the value of y they need. The program asks the user for x and then produces the value of y from the table data. It will then ask for another x value and so forth and print a message if the value of x is out of the range of the table. The program terminates on some special value of x (your choice). Be sure to let the user know what it is all about.

36 (8 pts) Develop a program that again gets the data from the mat file and then

1. PLOTS THE INDIVIDUAL POINTS WITH 'o' circle MARKS
2. FITS A CUBIC SPLINE TO THE DATA use star marks
3. FITS A LINEAR LEAST SQUARE CURVE use diamond marks
4. FITS A 7TH ORDER POLYNOMIAL use x-marks
5 PLOTS ALL CURVES SIMULTANEOUSLY using different symbols for the

plots.. Be sure to use an AXIS command to expand the range for better visibility of the final graph see also **help plot**

Do this in an m-file and attach the final graph.
Answer this question.
WHICH CURVE FITS THE DATA THE BEST AND WHY?