# Armstrong State University
# Engineering Studies
# MATLAB Marina – Interpolation Primer

**Prerequisites**
The Interpolation Primer assumes knowledge of the MATLAB IDE, MATLAB help, arithmetic operations, built in functions, scripts, variables, arrays, logic expressions, conditional structures, iteration, functions, debugging, characters and strings, cell arrays, structures, file input and output, and 2D plotting. Material on these topics is covered in the MATLAB Marina Introduction to MATLAB module, MATLAB Marina Variables module, MATLAB Marina Arrays module, MATLAB Marina Logic Expressions module, MATLAB Marina Conditional Structures module, MATLAB Marina Iteration module, MATLAB Marina Functions module, MATLAB Marina debugging module, MATLAB Marina Character and Strings module, MATLAB Marina Cell Arrays module, MATLAB Marina Structures module, MATLAB Marina File Input and Output module, and MATLAB Marina Plotting module.

**Learning Objectives**
1. Be able to use MATLAB to perform linear and cubic spline interpolation of data.
2. Be able to use MATLAB to estimate data values using interpolation
3. Understand when interpolation is appropriate to use.

**Terms**
interpolation, linear interpolation, cubic spline interpolation

**MATLAB Functions, Keywords, and Operators**
interp1, spline

**Interpolation**
Interpolation involves estimating a variable's value between two known values. For example, we might want to know the y value for x = 1 when we have data for x = 0 and x = 2 (say y = 5 and y = 14 respectively). Two of the more common interpolation techniques are linear and cubic spline interpolation.

**Piecewise (Spline) Interpolation**
Piecewise or spline interpolation involves fitting a polynomial function in each interval of the original function points and using the polynomial function to compute interpolated points between existing points in the interval. The polynomials for each interval are typically the same order. For n points, there are n-1 intervals and n-2 interior points. The spline must pass through the endpoints of the interval and the derivatives (up to order one less than the polynomial order) of the polynomial functions for adjacent intervals must match at the interior points.

For linear splines, a linear function, $f(x) = a_1 x + a_0$ is determined for each interval, and this function is used to compute interpolated points between the existing function points. The linear function can be determined using the two endpoints of the interval.

$f(x_1) = a_1 x_1 + a_0$

$f(x_2) = a_1 x_2 + a_0$

The system of two equations in two unknowns can be solved for a1 and a0, and the estimate of $f$ at x using linear interpolation can be determined from

$$f(x) = f(x_1) + \frac{x - x_1}{x_2 - x_1}\left(f(x_2) - f(x_1)\right)$$

For cubic splines, a cubic polynomial function $f(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0$ is determined for each two point interval, and this function is used to compute interpolated points between the existing function points. Four function points are needed to determine the coefficients for each interval and the first and second derivative of the polynomials must match for adjacent intervals.

**Linear Interpolation**

Linear interpolation assumes data between two known points can be estimated using a straight line between the known points. The closer the known points, the better the estimate. Linear interpolation is often used to determine values between given values when performing table look up, for example in Thermodynamics tables. Linear interpolation does not aid in getting smoother plots from coarse data as most plotting packages including MATLAB use straight line approximations (linear interpolation) between points to generate a continuous curve so the plots before and after linear interpolation will look the same.

MATLAB has a built in function `interp1` (interp one) for performing interpolation. The `interp1` function takes three arguments, two vectors `x` and `y` containing the original data and a vector of new x values `xnew` for which we want interpolated values of the function y. The `xnew` values should be within the range of the original x values. The function `interp1` performs interpolation by table lookup; it looks up the elements of `xnew` in `x` and returns values `ynew` interpolated from the corresponding `y` values.

For example, the MATLAB statements of Figure 1 generate the vector of estimated values `ynew`. The vector of data x and y must be the same length. The estimates `ynew` will be an array of the same length as `xnew`.

The MATLAB function `interp1` has two optional argument that allows one to specify how the interpolation between y values is performed (the default is linear interpolation) and to allow extrapolation for any elements in xnew outside the interval specified in x.

```
ynew = interp1(x,y,xnew,METHOD,'extrap');
```

```
>> x = [0, 1, 2, 3, 4, 5, 6];
>> y = [0, 4, 9, 13, 14, 20, 25];
>> xnew = 2.5;
>> ynew = interp1(x,y,xnew)
ynew =  11.00
>> xnew = 2:0.5:5;
>> ynew = interp1(x,y,xnew)
ynew =  9.00  11.00   13.00   13.50   14.00   17.00   20.00
```

Figure 1, MATLAB Statements Estimating y Values

The methods for interpolation are: nearest, linear, spline, pchirp, cubic, and v5cubic. The second optional argument, extrap, specifies to perform extrapolation using the specified interpolation method (the default extrapolation value is NaN, not a number). There are also 2-D and 3-D versions of this function (`interp2` and `interp3`).

**Cubic Spline Interpolation**

Cubic spline interpolation fits a cubic curve between the known points to estimate the unknown values. Cubic spline interpolation can be done with the `interp1` function using the spline optional argument or with the `spline` function. The `spline` function takes three arguments like the `interp1` function. Cubic spline interpolation generally generates better estimates than linear interpolation.

```
ynew = interp1(x,y,xnew,'spline');
```

or

```
ynew = spline(x,y,xnew);
```

```
% original sinusoid data
t = [0.0 : 0.05 : 0.4];
f = cos(10*pi*t);
% linear and cubic spline interpolation for vector of values
tnew = [t(1): 0.01 : t(end)];
fnewLinear = interp1(t,f,tnew);
fnewSpline = interp1(t,f,tnew,'spline');

figure(1)
plot(t,f,'o',tnew,fnewLinear,'-+'), xlabel('t'), ylabel('f')
title('Sinusoid, Original and Linearly Interpolated Data');
legend('original','interpolated');

figure(2)
plot(t,f,'o',tnew,fnewSpline,'-+'), xlabel('t'), ylabel('f')
title('Sinusoid, Original and Cubic Spline Interpolated Data');
legend('original','interpolated');
```

Figure 2a, MATLAB Program Performing Interpolation of Sinusoidal Function

The MATLAB programs of Figure 2a and 3a and the plots of Figure 2bc and 3bc illustrates linear and cubic spline interpolation for sinusoidal and quadratic functions.

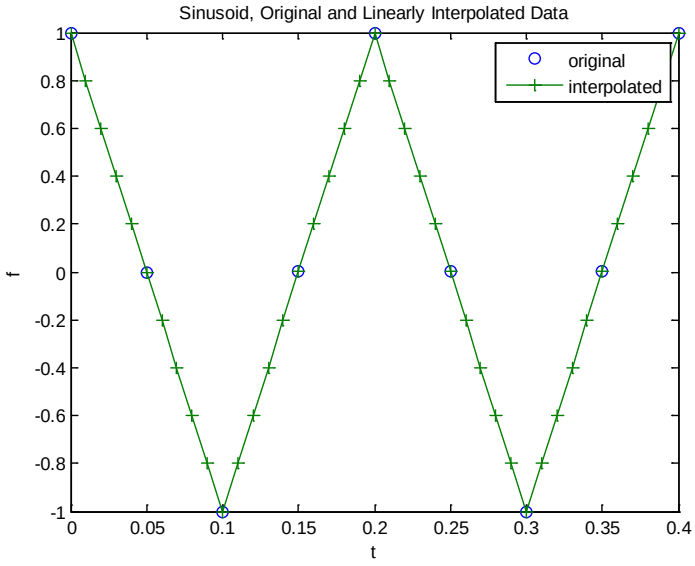


Figure 2b, Linear Interpolation of Sinusoidal Function

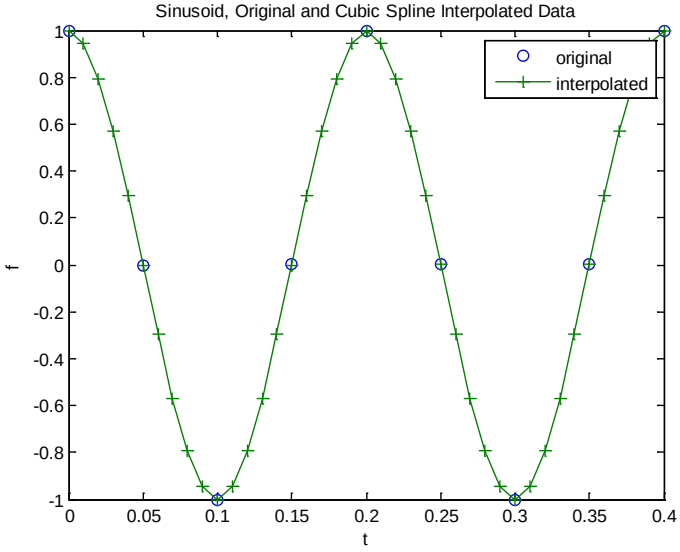


Figure 2c, Cubic Spline Interpolation of Sinusoidal Function

```
% original quadratic function data
x = [-5 : 1 : 5];
y = 2*x.^2 + 3*x - 5;
% linear and cubic spline interpolation for vector of values
xnew = [-5 : 0.5 : 5];
ynewLinear = interp1(x,y,xnew);
ynewSpline = interp1(x,y,xnew,'spline');

figure(3)
plot(x,y,'o',xnew,ynewLinear,'-+'), xlabel('x'), ylabel('y')
title('Quadratic, Original and Linearly Interpolated Data');
legend('original','interpolated');

figure(4)
plot(x,y,'o',xnew,ynewSpline,'-+'), xlabel('x'), ylabel('y')
title('Quadratic, Original and Cubic Spline Interpolated Data');
legend('original','interpolated');
```

Figure 3a, MATLAB Script Performing Interpolation of Quadratic Function

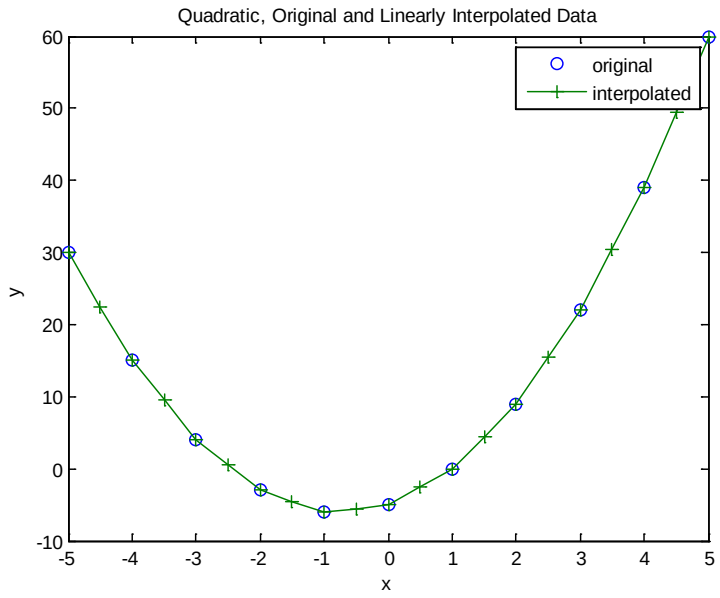


Figure 3b, Linear Interpolation of Quadratic Function

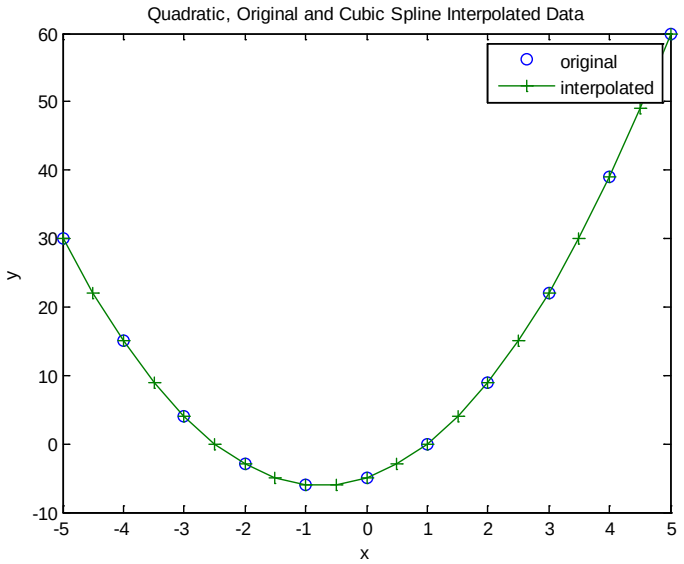


Figure 3c, Spline Interpolation of Quadratic Function

Interpolation is generally used to estimate values between existing values in a set of data. MATLAB's `interp1` function can be used to estimate a single value or multiple values based on the size of the variable holding the independent variable values for the estimate. Interpolation can also be used to estimate trends in data and provide more values for plotting when the available data values are known to be accurate.

Last modified Thursday, November 13, 2014