MATLAB Marina: Interpolation Examples

Student Learning Objectives
After completing this module, one should:
1. Be able to use MATLAB to estimate data values using interpolation

Terms
NA

MATLAB Functions, Keywords, and Operators
NA

Using Interpolation to Estimate Additional Data

Say we have measured data over the range of -6.0 to 7.5 for a quadratic polynomial behavior. A plot of the measured data is shown in Figure 1a.

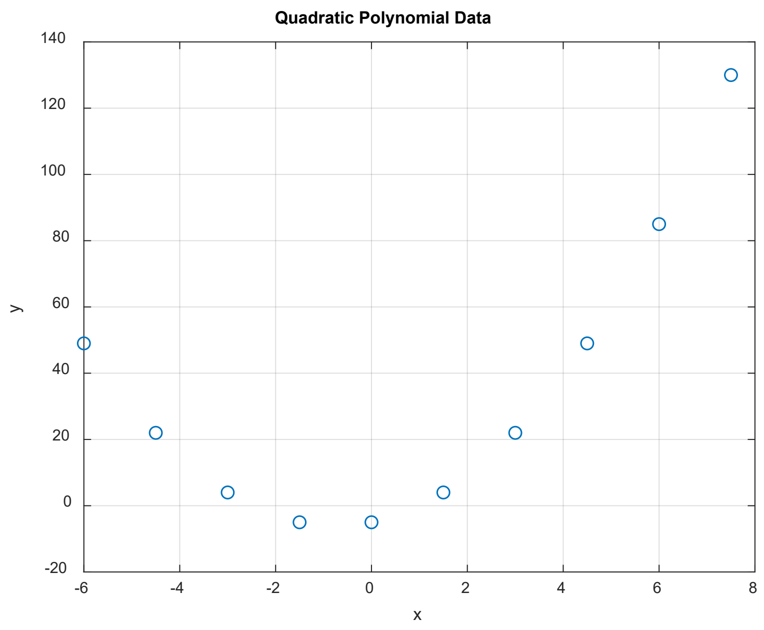

Figure 1a. Measured Data for Quadratic Polynomial Behavior

Figures 1b and 1c show the measured data and interpolated values between the measured data points obtained using linear interpolation and cubic spline interpolation. The measured data is plotting as circles and the interpolated data with xs and a solid line.

If you look carefully near the bottom of the plots of Figures 1b and 1c you can see that the plot created using estimates from cubic spline interpolation is smoother than the plot created using estimates from linear interpolation. Fitting a cubic spline rather than a linear spline between the measured points on a quadratic function provides “better” estimates.

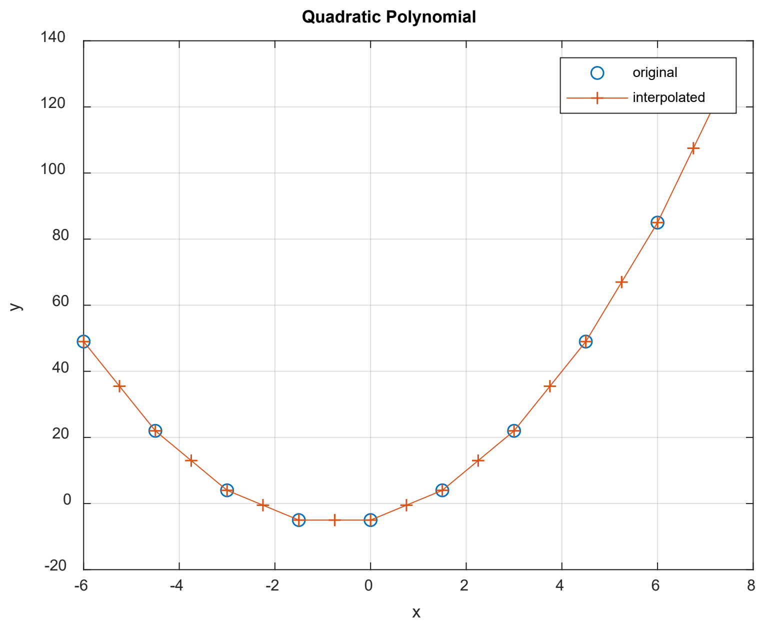

Figure 1b. Measured and Estimated Data for Quadradic Polynomial Behavior (Linear Spline)

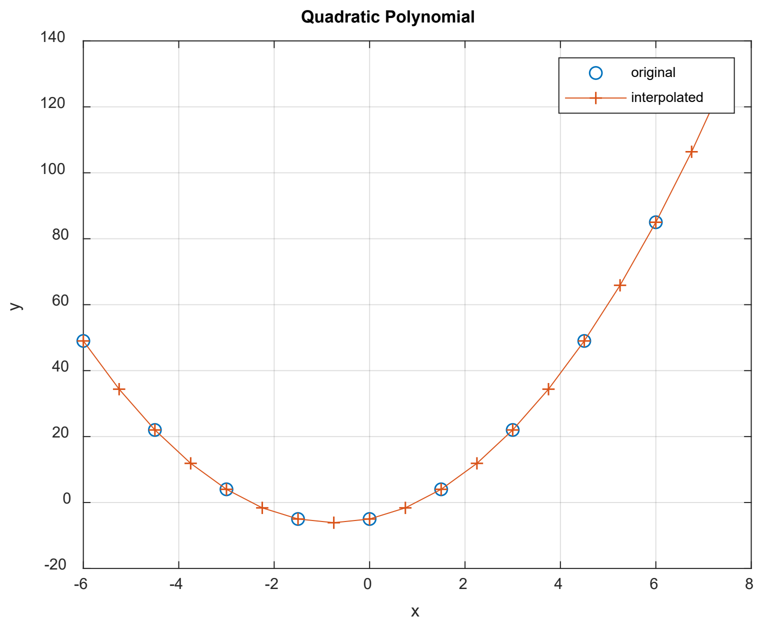

Figure 1c. Measured and Estimated Data for Quadradic Polynomial Behavior (Cubic Spline)

The MATLAB program of Figure 1d was used to estimate the additional data points and generate the plots.

```
% quadratic polynomial data
x = -6.0 : 1.5 : 7.5;
y = 2*x.^2 + 3*x - 5;
% estimating additional points between original data
xEst = -6.0 : 0.75 : 7.5;
yEstLin = interp1(x,y,xEst);
yEstSpl = interp1(x,y,xEst,'spline');

figure(1)
plot(x,y,'o'), xlabel('x'), ylabel('y'), grid
title('Quadratic Polynomial Data');

figure(2)
plot(x,y,'o',xEst,yEstLin,'-+'), xlabel('x'), ylabel('y'), grid
title('Quadratic Polynomial');
legend('original','interpolated');

figure(3)
plot(x,y,'o',xEst,yEstSpl,'-+'), xlabel('x'), ylabel('y'), grid
title('Quadratic Polynomial');
legend('original','interpolated');
```
Figure 1d. MATLAB Program

Using Interpolation to Aid in Determining Phenomenon Behavior
Say we have 0.4s of measured data taken 0.05s apart for a phenomenon. A plot of the measured data is shown in Figure 2a. It is difficult to tell much about the phenomenon from the plot of the measured data.

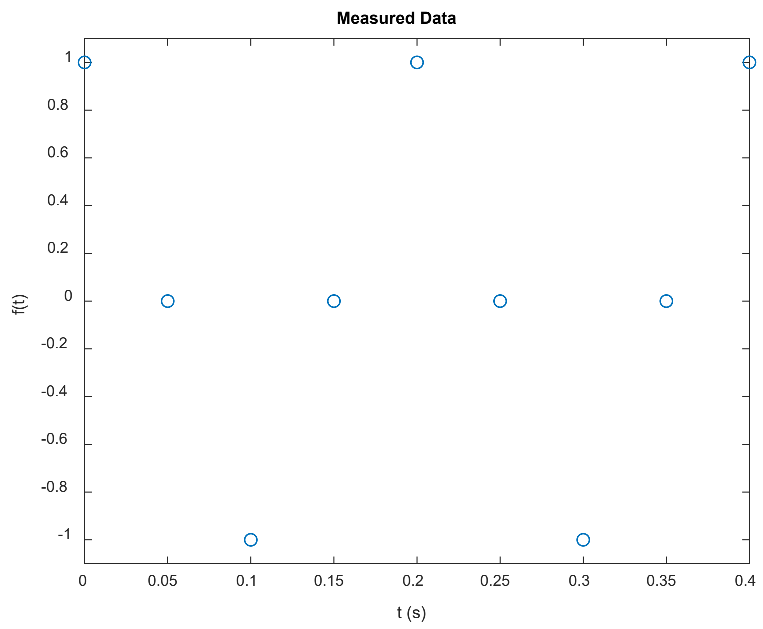

Figure 2a. Measured Data with No Apparent Pattern

If the data is known to be accurate, interpolation can be used to estimate additional data points and possibly help identify the general behavior of the phenomenon. Figures 2b and 2c show the measured data and interpolated values between the measured data points obtained using linear interpolation and cubic spline interpolation. The measured data is plotting as points (o) and the interpolated data with a solid line.

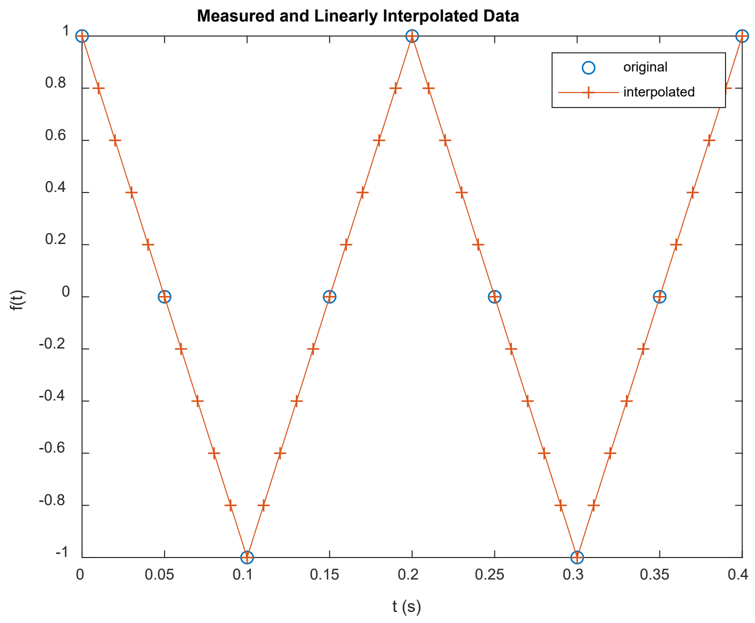

Figure 2b. Measured and Estimated Data (Linear Spline)

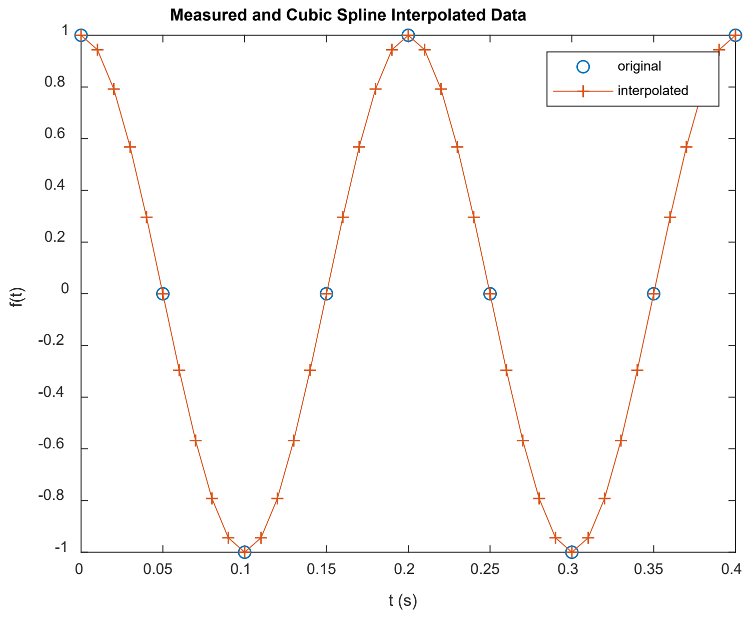

Figure 2c. Measured and Estimated Data (Cubic Spline)

The plot of Figure 2b corresponds to a triangular signal behavior whereas the plot of Figure 2c corresponds to a sinusoidal signal behavior. Both estimates were obtained using the same measured data; how do we know which set provides a better illustration of the behavior. For this particular example, the plot of Figure 2c is the better illustration of the behavior but we may not know this if we didn't know the behavior was sinusoidal in nature. The interpolation method used for the estimates should be chosen to match the general behavior of the phenomenon. If the general behavior is not known, the estimates obtained using interpolation may not be that accurate and identifying behavior from the estimates may be difficult.

The MATLAB program of Figure 2d was used to estimate the additional data points and generate the plots.

```
% original sinusoidal data
t = 0.0 : 0.05 : 0.4;
f = cos(10*pi*t);

figure(1)
plot(t,f,'o')
axis([t(1), t(end), 1.1*min(f), 1.1*max(f)]);
xlabel('t (s)'), ylabel('f(t)')
title('Measured Data');

% estimating additional points between original data
tEst = t(1): 0.01 : t(end);
fEstLin = interp1(t,f,tEst);
fEstSpline = interp1(t,f,tEst,'spline');

figure(2)
plot(t,f,'o',tEst,fEstLin,'-+')
xlabel('t (s)'), ylabel('f(t)')
title('Measured and Linearly Interpolated Data');
legend('original','interpolated');

figure(3)
plot(t,f,'o',tEst,fEstSpline,'-+')
xlabel('t (s)'), ylabel('f(t)')
title('Measured and Cubic Spline Interpolated Data');
legend('original','interpolated');
```

Figure 2d. MATLAB Program

Last modified March 30, 2022