MATLAB Marina: Interpolation

Student Learning Objectives

After completing this module, one should:

1. Be able to explain when interpolation is appropriate to use to estimate data values.
2. Be able to use MATLAB to perform linear and cubic spline interpolation of data.

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. Interpolation is used to estimate values that are not in an existing set of data. 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. 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 commonly used interpolation techniques are linear and cubic spline interpolation.

Piecewise 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 containing the original data and a vector of new independent variable values for which we want interpolated values of the function and returns the estimated values interpolated from the corresponding function values. The `interp1` function determines the values using table lookup. The new independent variable values should be within the range of the original independent variable values,

The MATLAB statements of Figure 1 illustrate estimating y values for a single x value and a vector of x values. The original data vectors must be the same length and the estimates will be an array of the same length as the number of independent variable values to interpolate at.

```
x = [0, 1, 2, 3, 4, 5, 6];
y = [0, 4, 9, 13, 14, 20, 25];
xEstA = 2.5;
yEstA = interp1(x,y,xEstA);
xEstB = 2:0.5:5;
yEstB = interp1(x,y,xEstB);
```

Figure 1. Linear Interpolation using interp1

The `interp1` function has two optional argument that allow one to specify how the interpolation between function values is performed (the default is linear interpolation) and to allow extrapolation for any elements outside the interval specified by the original data.

```
yEst = interp1(x,y,xEst,METHOD,'extrap');
```

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 2D and 3D 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. Cubic spline interpolation generally generates better estimates than linear interpolation when the data is nonlinear.

```
yEst = interp1(x,y,xEst,'spline');
```

or

```
yEst = spline(x,y,xEst);
```

Extrapolation

Extrapolation should generally only be used to estimate values when the measured data is known to be accurate, the extrapolation will not be very far off the edges of the measured data, and the general behavior is consistent off the edges of the measured data.

Last modified January 26, 2022