Curve Fitting Part II and Spline Interpolation

A. Curve Fitting

As we have seen, the \texttt{polyfit} command fits a polynomial function to a set of data points. However, sometimes it is appropriate to use a function other than a polynomial. The following types of functions are often used to model a data set.

- \( y = bx^m \) (power function)
- \( y = be^{mx} \) (exponential function)
- \( y = m \ln(x) + b \) (logarithmic function)
- \( y = \frac{1}{mx + b} \) (reciprocal function)

Note that \( y = bx^m \Rightarrow \ln(y) = \ln(b) + m \ln(x) \), so a loglog plot of a set of data points obeying a power law is a straight line.

\( y = be^{mx} \Rightarrow \ln(y) = \ln(b) + mx \), so a semilog plot (linear horizontal axis, logarithmic vertical axis) of a set of data points obeying an exponential law is a straight line.

A semilog plot (logarithmic horizontal axis, linear vertical axis) of a set of data points obeying a logarithmic law is a straight line.

\( y = \frac{1}{mx + b} \Rightarrow \frac{1}{y} = mx + b \), so a linear plot of \(1/y\) vs. \( x \) is a straight line if the data obey a reciprocal law.

Once you have chosen the type of function you want to use to model your data, you can use the \texttt{polyfit} command to calculate the values of \( b \) and \( m \). If you have a theoretical basis for choosing a particular type of function to model your data, use that type of function. If you have no idea what type of function to use, you can look at a loglog plot, two semilog plots, and a linear plot of \(1/y\) vs. \( x \) to see if any of the graphs are close to a straight line. If one of the four graphs looks like a line, use the corresponding function to model your data.

Here is an example. I obtained the data by measuring the temperature of water in a hot pot every three minutes.

\[
x = [0 3 6 9 12 15 18 21 24 27 30 33];
y = [50.6 46.8 43.2 40.0 37.0 34.2 31.6 29.2 27.0 25.0 23.1 21.4];
\text{subplot}(2,2,1)
\text{loglog}(x,y)
\text{subplot}(2,2,2)
\text{semilogy}(x,y)
\text{subplot}(2,2,3)
\text{semilogx}(x,y)
\text{subplot}(2,2,4)
\text{plot}(x,1./y)\]
If the first graph looks like a line, you can use the command `polyfit(log(x), log(y), 1)` to calculate the values of $m$ and $\ln(b)$.

If the second graph looks like a line, you can use the command `polyfit(x, log(y), 1)` to calculate the values of $m$ and $\ln(b)$.

If the third graph looks like a line, you can use the command `polyfit(log(x), y, 1)` to calculate the values of $m$ and $b$.

If the fourth graph looks like a line, you can use the command `polyfit(x, 1./y, 1)` to calculate the values of $m$ and $b$.

**B. Spline Interpolation**

If you have no reason to choose a functional model to fit a set of data but you want to use the data to make predictions, you might want to use a curve that passes through all the data points. As we have seen, you can always find a polynomial of degree $n - 1$ that passes through a set of $n$ data points, but this might not be a good idea because high-degree polynomials can oscillate quite a bit. An alternative is to use a piecewise polynomial, also known as a **spline**. A popular choice is a piecewise cubic function. The degree is high enough to provide a fair degree of smoothness but not so high as to cause large oscillations.

The MATLAB command `spline` produces a cubic spline, given a set of data points as input. The command `ppval` can be used to evaluate splines produced by the `spline` command. Here is an example.

```matlab
x_data = [-1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75 1];
y_data = [0.3333 0.4324 0.5714 0.7619 1.0000 1.2308 1.3333 1.2308 1.0000];
pp = spline(x_data, y_data);
x_plot=linspace(-1, 1, 50);
y_plot = ppval(pp, x_plot);
plot(x_data, y_data, 'o', x_plot, y_plot, '-b')
```

If you wanted to estimate a $y$ value at an $x$ value not among the given data, you can use the `ppval` command. For example, to estimate the $y$ value corresponding to $x = 0.1$ you can use the command `ppval(pp, 0.1)`

**C. Alternate Interpolation Methods**

The MATLAB command `interp1` offers several interpolation options. Try these commands.

```matlab
x = linspace(0, 2*pi, 11);
y = sin(x);
x_plot = linspace(0, 2*pi, 51);
y_plot = interp1(x, y, x_plot, 'linear');
plot(x, y, 'o', x_plot, y_plot, '-b')
The `interp1` command with the 'linear' option produces a piecewise linear function that passes through the data points given by the $x$ and $y$ arrays.

If you use 'spline' instead of 'linear' you will generate a piecewise cubic interpolant, just like the `spline` command generates. Try these commands:
x = linspace(0, 2*pi, 10);
y = sin(x);
x_plot = linspace(0, 2*pi, 25);
y_plot = interp1(x, y, x_plot, 'spline');
plot(x, y, 'o', x_plot, y_plot, '-b')

Practice Problems (from Gilat, MATLAB: An Introduction with Applications.)

1. Below are data showing how the stress concentration factor \( k \) in a stepped shaft depends on the ratio of two shaft dimensions.
   (a) Use a power function \( k = b(r/d)^m \) to model the relationship between \( k \) and \( r/d \). Determine the values of \( b \) and \( m \) that best fit the data.
   (b) Plot the data points and the curve-fitted model.
   (c) Use the model to predict the stress concentration factor for \( r/d = 0.04 \).

<table>
<thead>
<tr>
<th>( r/d )</th>
<th>0.3</th>
<th>0.26</th>
<th>0.22</th>
<th>0.18</th>
<th>0.14</th>
<th>0.1</th>
<th>0.06</th>
<th>0.02</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td>1.18</td>
<td>1.19</td>
<td>1.21</td>
<td>1.26</td>
<td>1.32</td>
<td>1.43</td>
<td>1.6</td>
<td>1.98</td>
</tr>
</tbody>
</table>

2. The population of the world for selected years from 1750 to 2009 is given in the following table.

<table>
<thead>
<tr>
<th>Year</th>
<th>1750</th>
<th>1800</th>
<th>1850</th>
<th>1900</th>
<th>1950</th>
<th>1990</th>
<th>2000</th>
<th>2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population (millions)</td>
<td>791</td>
<td>980</td>
<td>1,260</td>
<td>1,650</td>
<td>2,520</td>
<td>5,270</td>
<td>6,060</td>
<td>6,800</td>
</tr>
</tbody>
</table>

Fit the data with a cubic spline. Estimate the population in 1975. Make a plot of the data points and the spline function.