Exercises

A fitting problem

Given a set of n+1 data points:

$$\begin{matrix}(x_0\,,y_0\,)\\(x_1\,,y_1\,)\\\vdots\\(x_n\,,y_n\,)\end{matrix}$$

we can construct a simple polynomial interpolant

$$p(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_2x^2+a_1x+a_0 \qquad (1)$$

to the data points by solving a certain set of linear equations.

If the polynomial *p(x)* is an interpolant then evaluating *p(x)* at the original data points x_i will give the original y_i, i.e.

$$p(x_i)=y_i \qquad \text{for all } i\in\{0,1,\ldots,n\}\,.$$

Using equation (1), we get a system of linear equations in the coefficients a_i. In matrix-vector form, the system looks like this:

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \ldots & x_0 & 1\\ x_1^n & x_1^{n-1} & x_1^{n-2} & \ldots & x_1 & 1\\ \vdots & \vdots & \vdots & & \vdots & \vdots\\ x_n^n & x_n^{n-1} & x_n^{n-2} & \ldots & x_n & 1\end{bmatrix}\begin{bmatrix}a_n\\a_{n-1}\\\vdots\\a_0\end{bmatrix}=\begin{bmatrix}y_0\\y_1\\\vdots\\y_n\end{bmatrix} \qquad (2)$$

The matrix in equation (2) is known as a *Vandermonde* matrix, named after the French 18th century musician, chemist and mathematician *Alexandre-Théophile Vandermonde*.

Generating the coefficients a_i of $p(x)$

Given this set of (x,y) data pairs:

x	-1.5	-1.25	-1.0	-0.75	-0.5	-0.25	0.0	0.25	0.5	0.75	1.0	1.25	1.5
y	0.0	0.25	0.5	1.25	3.0	6.0	8.0	6.0	3.0	1.25	0.5	0.25	0.0

construct the set of simultaneous equations described in (2), and solve them to find the coefficients a_i of the interpolating polynomial *p(x)*. To solve the system of equations, you may use a NAG routine (e.g. F07AA), or you may use some of MATLAB's built-in linear algebra functionality. (Hint: You can do everything with MATLAB's polyfit)

Having found the coefficients of the polynomial, evaluate *p(x)* at the input data points x_i. This is a check on whether or not *p(x)* is indeed an interpolating polynomial - and we hope that $p(x_i)$ is more or less equal to y_i for all our data points.

What happens if we then evaluate the polynomial *p(x)* at points *x* other than the original data points? The reason for generating an interpolating polynomial in the first place might well be to do something like this. Try evaluating *p(x)* at many points in between the x_i data points, and plot the results on a graph. You can use MATLAB's polyval. What does it look like?

What happened?

The polynomial that we fitted above was a high-order polynomial. It is well known that using such a polynomial to estimate a function at a value not in the original data set is liable to cause trouble. Far preferable is to fit the original data set using a piecewise polynomial method, such as cubic splines.

A monotonic interpolant alternative

One way to avoid unwanted oscillations in a fitted interpolant is to use a *monotonic interpolant*. By *monotonic* we mean that the derivative (or gradient) of the fitted curve always slopes in the same direction as the derivative of the original data points. The NAG routine E01BE performs this task. Try using E01BE, which constructs the interpolant, followed by E01BF, to evaluate the interpolant. Plot the results on a graph and compare them with the results you got using the simple polynomial. Which results do you prefer?