Polynomial Fitting and Excel

BY

Ben Rickman
03/09/06

1. Introduction

Assume that we are given some data in the form of N points (x_n, y_n), $1 \leqslant n \leqslant N$, and we wish to fit a polynomial $f(x)$ to the data. A **polynomial of degree M** is a function of the following form,

$$f(x) = \sum_{p=0}^{M} a_p \, x^p \tag{1}$$

where the a_p just denote real numbers. We adopt the convention that $x^0 = 1$, no matter what value x is. The following are polynomials $1 + 2\,x + 3\,x^2$ and $-3 + 4\,x^5$. A function such as $\sin(x)$ is not because it cannot be written in the form (1) for finite M.

We will assume that we are fitting a polynomial of degree $M + 1 \leqslant N$ to the N data points. In the case $M + 1 = N$ the fit will go through each point, i.e. the fit interpolates between points. We will consider two cases, $M + 1 = N$ and $M + 1 < N$.

In section 4, there is a brief account of how polynomial fits can be found in Matlab, Mathcad and Excel. It will be explained why the polynomial fit formula shown by Excel cannot be taken at face value.

2. Interpolation, $M + 1 = N$

In this case, we have to solve the following N equations for the polynomial coefficients a_p,

$$\sum_{p=0}^{M} a_p \, (x_n)^p = y_n, \quad 1 \leqslant n \leqslant N \tag{2}$$

To solve this, it is customary to write these equations in matrix form. So,

$$\begin{pmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^M \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^M \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_N & (x_N)^2 & \cdots & (x_N)^M \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \tag{3}$$

The matrix

$$V = \begin{pmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^M \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^M \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_N & (x_N)^2 & \cdots & (x_N)^M \end{pmatrix} \tag{4}$$

is a square matrix called the Vandermonde matrix, named for Alexandre-Theophile Vandermonde 1735 to 1796. This matrix is discussed in a number of references on the internet, including Wikipedia. Equation (3) can be solved when the matrix V is non-singular, i.e. it has a non-zero determinant. The determinant of V is

$$\det(V) = \prod_{1 \leqslant i < j \leq N} (x_i - x_j) \tag{5}$$

and hence is non-zero when all the ordinates of the points we are trying to fit are distinct.

The determinant of a Vandermonde matrix can be quite small, and so equation (3) is best solved using a very numerically stable method such as the QR method. This finds an orthogonal matrix Q and an upper triangular matrix R such that $V = Q\,R$.

Having done this, we can solved (3) in two stages as follows, remembering that for an orthogonal matrix, $Q^{-1} = Q^T$.

$$R \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{pmatrix} = Q^T \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \tag{6}$$

and then, using the fact that R is upper triangular, use back-substitution to solve for the polynomial coefficients, $a_0, a_1, \cdots, a_M$.

Back-substitution will be explained by an example

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 7 \\ 8 \end{pmatrix}$$

Looking at the last row, we get the equation $6\,z = 8$, so $z = 4/3$. The next row leads to the equation $4\,y + 5\,z = 7$, so $y = (7 - 5\,z)/4 = 1/12$. Taking the first row, $x + 2\,y + 3\,z = 6$, so $x = 11/6$.

The QR decomposition can be found in Matlab using the function qr(V). A function of the same name is available in Mathcad to do the QR decomposition. However the extraction of Q and R in Mathcad is slightly more difficult. A quicksheet is available to show how this can be done.

3. Regression, the case $M+1 < N$

The fit is done by making a the leat squares fit to the data points (x_n, y_n), $1 \leqslant n \leqslant N$, viz minimising the function

$$C(a_0, \cdots, a_M) = \sum_{n=1}^{N} (y_n - f(x_n))^2 = \sum_{n=1}^{N} \left[y_n - \sum_{p=0}^{M} a_p\,(x_n)^p \right]^2 \tag{7}$$

The best way to explain the solution of these equations is to write them in matrix form. Let

$$\vec{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{pmatrix}, \vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

and recall equation (4). Then

$$G(\vec{a}) = \vec{y} - V\vec{a} = \begin{pmatrix} y_1 - \sum_{p=0}^{M} a_p (x_1)^p \\ y_2 - \sum_{p=0}^{M} a_p (x_2)^p \\ \vdots \\ y_N - \sum_{p=0}^{M} a_p (x_N)^p \end{pmatrix} \tag{8}$$

and so equation (7) can be written as

$$C(\vec{a}) = G(\vec{a})^T G(\vec{a}) = (\vec{y} - V\vec{a})^T (\vec{y} - V\vec{a}) \tag{9}$$

We will assume that $V^T V$ is an invertible matrix. This matrix is an $(M+1) \times (M+1)$ matrix, i.e. it is of the order of the same number of coefficients as the polynomial has. In the case one has (say) 20 points to fit with a 5th order polynomial, $V^T V$ is of order 6×6. The objective of this section is to show that the coefficients of the polynomial can be obtained by solving $M+1$ linear equations, and that these equations can be solved using a numerically stable method.

The value of $\vec{a}$ that minimises $C(\vec{a})$ is the vector $\vec{b} = \left(V^T V\right)^{-1} V^T \vec{y}$. The proof of this is really easy. Firstly observe that, if I denotes the $N \times N$ identity matrix and $\vec{0}$ denotes the $N \times 1$ zero vector,

$$(\vec{y} - V\vec{b})^T V = \vec{y}^T \left(I - V \left(V^T V\right)^{-1} V^T\right) V = \vec{0} \tag{10}$$

Let $\vec{x} = \vec{b} - \vec{a}$. Then $\vec{y} - V\vec{a} = \vec{y} - V\vec{b} + V\vec{x}$, so (9) becomes

$$C(\vec{a}) = (\vec{y} - V\vec{b} + V\vec{x})^T (\vec{y} - V\vec{b} + V\vec{x})$$

This formula can be rewitten as

$$C(\vec{a}) = (\vec{y} - V\vec{b})^T (\vec{y} - V\vec{b}) + (\vec{y} - V\vec{b})^T V\vec{x} + \vec{x}^T V^T (\vec{y} - V\vec{b}) + x^T V^T V\vec{x}$$

Applying formula (10), we conclude that

$$C(\vec{a}) = (\vec{y} - V\vec{b})^T (\vec{y} - V\vec{b}) + \vec{x}^T V^T V\vec{x} \tag{11}$$

As $\vec{x}^T\, V^T\, V\, \vec{x}$ is the only term in equation (11) that depends on $\vec{a}$, $C(\vec{a})$ is minimized when $\vec{x}^T V^T V\vec{x}$ is minimized.

We note that $\vec{x}^T V^T V\vec{x} = (V\vec{x})^T (V\vec{x})$. Set $\vec{d} = V\vec{x}$. Then $\vec{x}^T V^T V\vec{x} = \vec{d}^T \vec{d}$ and $\vec{d}$ is an $N \times 1$ real valued vector.

But $\vec{d}^{\,T}\,\vec{d} = \sum_{n=1}^{N} (d_n)^2$, where d_n denotes the n'th entry in the column vector $\vec{d}$, i.e. the sum of the squares of every element in $\vec{d}$. As every entry in $\vec{d}$ is real, $\vec{d}^{\,T}\,\vec{d}$ is greater than or equal to zero.

So $C(\vec{a})$ is minimized when $\vec{d}^{\,T}\vec{d}$ equals 0, i.e. when every entry of $\vec{d}$ is zero, i.e. when $\vec{d} = 0$.

Let the symbol $\stackrel{\Delta}{=}$ denote "is equal to by definition".

So $C(\vec{a})$ is minimised implies that $\vec{d} \stackrel{\Delta}{=} V\vec{x} = 0$. So $V^T V\vec{x} = 0$. Because, by assumption, $V^T V$ is invertible, $\vec{x} = 0$.

As $\vec{x} \stackrel{\Delta}{=} \vec{b} - \vec{a}$, we conclude that $C(\vec{a})$ is minimised only when $\vec{a} = \vec{b} \stackrel{\Delta}{=} \left(V^T V\right)^{-1} V^T \vec{y}$, which is what we wanted to prove.

We deduce, using equation (10), that the minimum value of $C(\vec{a})$ is $C(\vec{b}) = (\vec{y} - V\vec{b})^T \vec{y}$.

To find $\vec{b}$, one needs to solve the $(M+1) \times (M+1)$ set of linear equations

$$\left(V^T V\right)\vec{b} = V^T \vec{y} \tag{12}$$

The matrix $V^T V$ is a positive definite $(M + 1) \times (M + 1)$ positive definite matrix. So all its eigenvalues are real and positive. In addition, there is an orthogonal matrix R of eigenvectors of $V^T V$ (so that $R^{-1} = R^T$) such that

$$V^T V = R\,\Lambda\,R^T \tag{13}$$

where Λ denotes a diagonal matrix of real positive eigenvalues. So (12) can be rewritten as

$$\Lambda\left(R^T\vec{b}\right) = R^T\,V\vec{y} \tag{14}$$

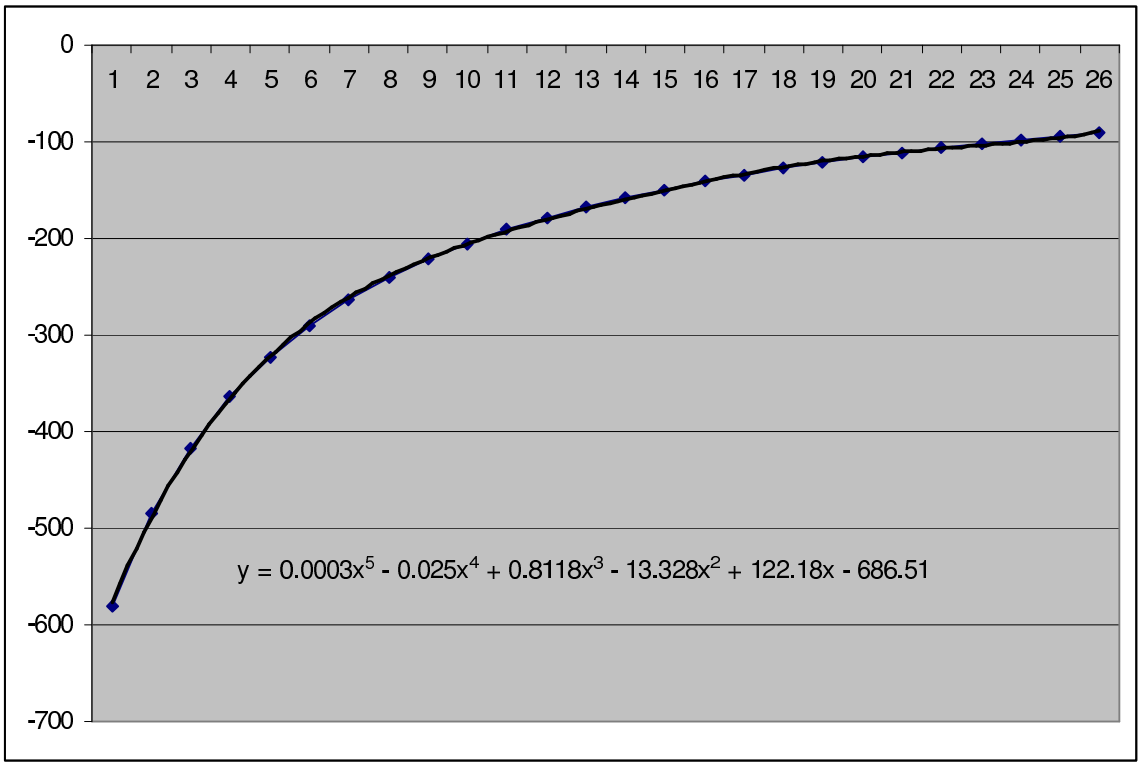
As Λ is diagonal, one can easily solve (14) for $\vec{z} = R^T\vec{b}$ and then determine $\vec{b}$ using the formula $\vec{b} = R\,\vec{z}$.

Advanced: The objective of this section is to find the vector $V\vec{a}$ which is closest to $\vec{y}$. Vectors $V\vec{a}$ are members of the vector space $\mathfrak{V}$ spanned by the columns of V. The space $\mathfrak{V}$ is a normed linear space with the norm $\left\|\vec{x}\right\| = \sqrt{\vec{x}^T\vec{x}}$. Equation (10) says that the vector between $\vec{y}$ and the closest vector to $\vec{y}$ in $\mathfrak{V}$ is orthogonal to all elements of $\mathfrak{V}$, i.e. it is the **orthogonal projection** from $\vec{y}$ onto $\mathfrak{V}$.

4. Use of Matlab, Mathcad and Excel for polynomial fitting

The theory used in sections 3 and 4 is the theory used by the Matlab polyfit function. The polyfit documentation in the Matlab help explains this. In Mathcad, the function **regress** will produce a polynomial fit.

When using Excel, first produce a chart. Clicking on the line, one can go to the chart menu and bring up the Add Trendline... option which brings up a dialog box, in which the degree of the polynomial fit can be selected and the equation of the polynomial fit can be added to the plot, as shown below. In the example below, a polynomial of degree 5 was chosen as the fit polynomial.



Result of applying Matlab polyfit

```
>> polyfit([1:26],mydata',5)

ans =

 3.0278e-004
-2.5021e-002
 8.1175e-001
-1.3328e+001
 1.2218e+002
-6.8651e+002
```

The error in the x^5 term is particularly serious, leading to an error of $0.0278e-4\times 26^5 \approx 33$.

This equation is therefore an approximate representation of the equation used by Excel itself, i.e. the full double precision coefficients. The author does not know of any way of extracting the polynomial coefficients that Excel actually uses, so the polynomial displayed by Excel has to be treated with caution.

5. Conclusion

This has been written from a personal perspective. I have not had to solve extremely large scale least squares problems. Different techniques are needed to solve these. The use of the QR decomposition to solve equation (3) is a personal preferance and one should note that there are other matrix decompositions such as LU which will also do the job.

The Excel polynomial fit formula must be treated with caution because Excel does not always display polynomial coefficients to sufficient accuracy to reproduce the trendline it displays.