AMSC/CMSC 460 Computational Methods, Fall 2007
UNIT 7: Solving Linear Least Squares Problems

Dianne P. O'Leary

Least Squares approximation

Read: Chapter 7. Skip Section 7.4.

The plan:

- Why least squares is useful.
- Solving least squares problems using the normal equations.
- Solving least squares problems using orthogonal factorizations.
- Computing a QR factorization (Given's rotations).
- Examples.

Note: In this chapter, all norms are meant to be 2-norms.

Why least squares is useful

Motivating example

Fit a model to data in order to reduce the effects of noise in the measurements.

Given: a set of basis functions $\phi_1(t), \phi_2(t), \ldots, \phi_n(t)$ that we believe to model the behavior of some function or set of data,

Find: coefficients $x_1, x_2, \ldots, x_n$ so that

$$u(t) = \sum_{j=1}^{n} x_j \phi_j(t)$$

We measure this difference at a set of m $t_1, \ldots, t_m$ at which we have measurements b_i:

$$\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{u}\|^2 = \min_{x} \sum_{i=1}^{m} [b_i - u(t_i)]^2 \,.$$

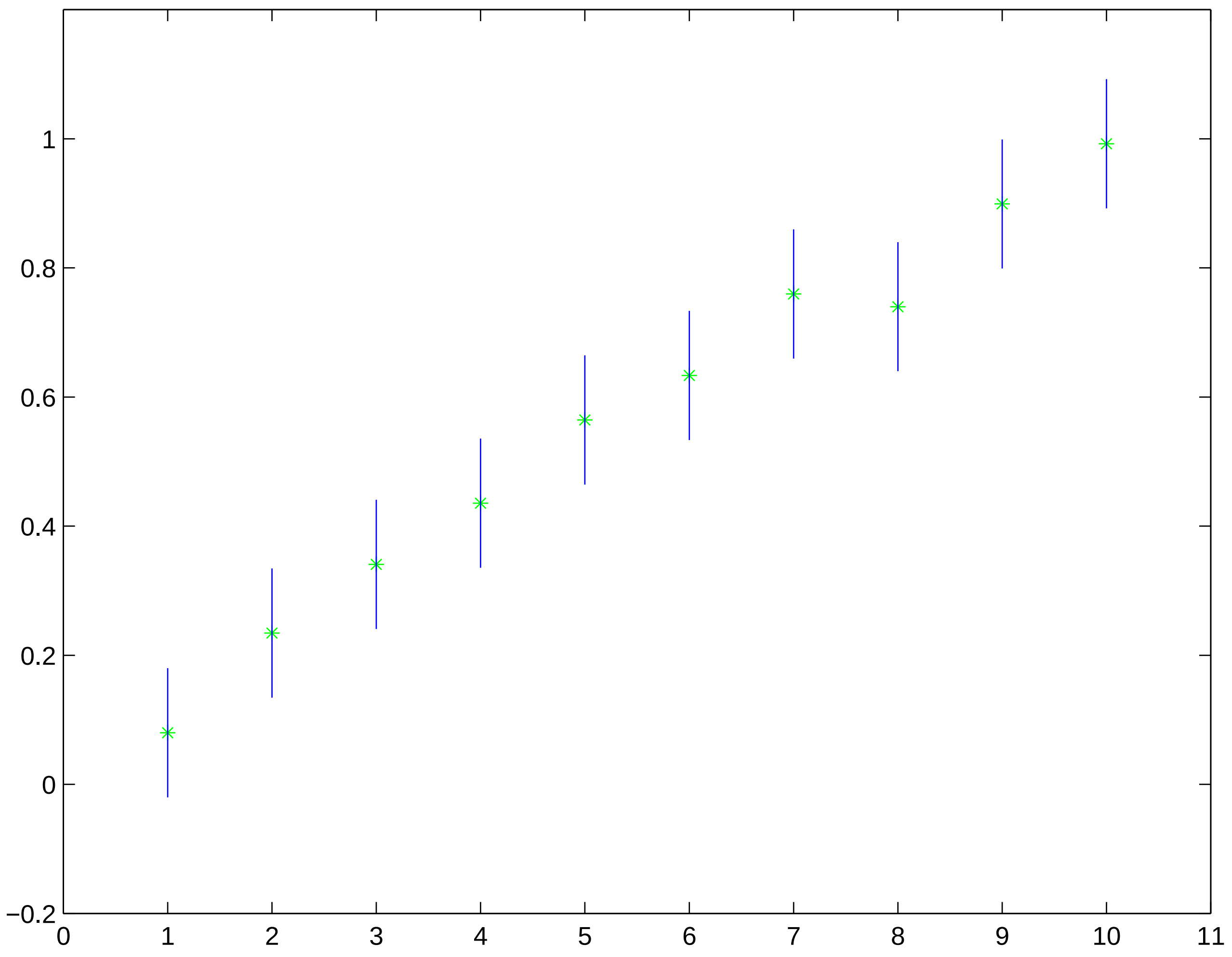

Figure 1: Is a straight line a good model to this data?

Transformation of the problem

Let's see if we can get some insight into our minimization problem.

$$\|\mathbf{b}-\mathbf{u}\|^2 = \sum_{i=1}^{m}[b_i - u(t_i)]^2$$

Now we know that

$$u(t) = \sum_{j=1}^{n} x_j \phi_j(t)$$

so

$$\|\mathbf{b}-\mathbf{u}\|^2 = \sum_{i=1}^{m}[b_i - \sum_{j=1}^{n} x_j \phi_j(t_i)]^2 .$$

If we define the residual vector r so that

$$r_i = b_i - \sum_{j=1}^{n} x_j \phi_j(t_i)$$

then our minimization problem is to minimize $\|\mathbf{r}\|$ over all choices of $\mathbf{x}$.

There is a further very nice simplification. Define the matrix $\mathbf{A}$ to have entries $a_{ij} = \phi_j(t_i)$. Then

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x} ,$$

and we have expressed our problem as a matrix one:

$$\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|^2 .$$

This quantity that we minimize is $\mathbf{b}^T\mathbf{b} - 2\mathbf{x}^T\mathbf{A}^T\mathbf{b} + \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}$, and we will let $\mathbf{B} = \mathbf{A}^T\mathbf{A}$ and $\mathbf{c} = \mathbf{A}^T\mathbf{b}$, reducing the problem to minimizing $\mathbf{b}^T\mathbf{b} - 2\mathbf{x}^T\mathbf{c} + \mathbf{x}^T\mathbf{B}\mathbf{x}$.

Note: We could use a weighted norm here if some observations were more important than others. This would yield the problem

$$\|\mathbf{b}-\mathbf{u}\|^2 = \sum_{i=1}^{m} w_i[b_i - \sum_{j=1}^{n} x_j \phi_j(t_i)]^2$$

where the weights w_i are positive.

Solution: via calculus

Let's solve using calculus. Necessary conditions to have a minimizer are:

1. The first derivative must be zero.

$$-2\mathbf{c} + 2\mathbf{B}\mathbf{x} = \mathbf{0}\,.$$

   Jargon: The n linear equations $\mathbf{B}\mathbf{x} = \mathbf{c}$ are called the normal equations.

2. The second derivative matrix, $\mathbf{B}$ must be positive semi-definite.

   Since $\mathbf{B} = \mathbf{A}^T\mathbf{A}$, we see that, for any vector $\mathbf{x}$,

$$\mathbf{x}^T\mathbf{B}\mathbf{x} = \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} = \|\mathbf{A}\mathbf{x}\|^2 \geq 0$$

   and therefore $\mathbf{B}$ is symmetric positive semi-definite.

   (In other words, $\mathbf{B}$ has no negative eigenvalues.)

If $\mathbf{B}$ is full rank, then the solution exists and is unique. If $\mathbf{B}$ is rank deficient (which happens when $\mathbf{A}$ fails to have linearly independent columns), then it is fair to say that we have chosen a poor set of basis functions ϕ_j.

Note: Unfortunately, real problems often have bad bases.

If $\mathbf{B}$ is full rank, then we can solve the linear system $\mathbf{B}\mathbf{x} = \mathbf{c}$ by using an algorithm similar to the LU factorization. Taking advantage of the fact that $\mathbf{B}$ is symmetric, we look for a factorization in which $\mathbf{U} = \mathbf{L}^T$. We can derive the formulas for the entries of $\mathbf{L}$ by looking at the equation $\mathbf{B} = \mathbf{L}\mathbf{L}^T$. Here is how it looks for $n = 3$. (See Section 7.3.3 for more detail.)

$$\begin{aligned}
\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} &= \begin{bmatrix} \ell_{11} & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} \end{bmatrix} \begin{bmatrix} \ell_{11} & \ell_{21} & \ell_{31} \\ 0 & \ell_{22} & \ell_{32} \\ 0 & 0 & \ell_{33} \end{bmatrix} \\
&= \begin{bmatrix} \ell_{11}^2 & \ell_{11}\ell_{21} & \ell_{11}\ell_{31} \\ \ell_{11}\ell_{21} & \ell_{21}^2 + \ell_{22}^2 & \ell_{21}\ell_{31} + \ell_{22}\ell_{32} \\ \ell_{11}\ell_{31} & \ell_{21}\ell_{31} + \ell_{22}\ell_{32} & \ell_{31}^2 + \ell_{32}^2 + \ell_{33}^2 \end{bmatrix}
\end{aligned}$$

Using these formulas, we can compute the elements of $\mathbf{L}$ column by column.

Unanswered questions

- Are the normal equations the best formulation numerically?
- How should we choose the basis in order to get good numerical stability?

To answer the first question "no", we will take a digression and learn about QR factorizations and then sensitivity.

Solving least squares problems using orthogonal factorizations

The QR factorization

The best tool for solving least squares problems in which $\mathbf{A}$ is a full rank matrix is a factorization called the QR factorization. We compute two matrices $\mathbf{Q}$ of dimension $m \times n$ and $\mathbf{R}$ of dimension $n \times n$ so that $\mathbf{A} = \mathbf{QR}$, $\mathbf{R}$ is upper triangular, and $\mathbf{Q}$ has orthonormal columns (i.e., $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_{n\times n}$).

Some algorithms give us a little more: $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$, where $\hat{\mathbf{R}}$ is $m \times n$,

$$\hat{\mathbf{R}} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$$

$\hat{\mathbf{Q}} = [\mathbf{Q}, \bar{\mathbf{Q}}]$ is $m \times m$ with orthonormal columns.

What we can do with a QR factorization

1. If $\mathbf{y}$ is in the range of $\mathbf{A}$, then $\mathbf{y} = \mathbf{Ax}$ for some vector $\mathbf{x}$, so

   $$\mathbf{y} = (\mathbf{QR})\mathbf{x} = \mathbf{Q}(\mathbf{Rx}),$$

   so $\mathbf{y}$ is in the range of $\mathbf{Q}$.

   Similarly, if $\mathbf{y} = \mathbf{Qz}$, then $\mathbf{y} = \mathbf{Ax}$ with $\mathbf{x} = \mathbf{R}^{-1}\mathbf{z}$, so $\mathbf{y}$ is in the range of $\mathbf{A}$.

   Therefore, the columns of $\mathbf{Q}$ form an orthonormal basis for the range of $\mathbf{A}$.

2. If $\mathbf{z}$ is in the nullspace of $\mathbf{A}^T$, then $\mathbf{A}^T\mathbf{z} = \mathbf{0}$. Therefore,

   $$(\mathbf{QR})^T\mathbf{z} = \mathbf{R}^T\mathbf{Q}^T\mathbf{z} = \mathbf{0},$$

   and, since the vector $\mathbf{Q}^T\mathbf{z}$ forms the coefficients for the linear combination of columns of $\mathbf{R}$, it must be that $\mathbf{Q}^T\mathbf{z} = \mathbf{0}$ (if $\mathbf{R}$ is full rank), and therefore $\mathbf{z}$ is a linear combination of columns of $\bar{\mathbf{Q}}$. Thus, the columns of $\bar{\mathbf{Q}}$ form an orthonormal basis for the nullspace of $\mathbf{A}^T$ (if $\mathbf{A}$ is full rank).

Two convenient facts about orthogonal matrices

- Suppose $\hat{\mathbf{Q}}$ is any square matrix satisfying $\hat{\mathbf{Q}}^T\hat{\mathbf{Q}} = \mathbf{I}$. Then for any m-vector $\mathbf{r}$,

  $$\|\hat{\mathbf{Q}}\mathbf{r}\|^2 = (\hat{\mathbf{Q}}\mathbf{r})^T\hat{\mathbf{Q}}\mathbf{r} = \mathbf{r}^T\hat{\mathbf{Q}}^T\hat{\mathbf{Q}}\mathbf{r} = \mathbf{r}^T\mathbf{r} = \|\mathbf{r}\|^2 .$$

  So we say that the Euclidean norm is invariant under orthogonal transformations.

- If $\mathbf{Q}$ is an orthogonal matrix, then so is

  $$\mathbf{Q}_e = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}$$

  since $\mathbf{Q}_e^T\mathbf{Q}_e = \mathbf{I}$.

Why this helps us solve the least squares problem

Let $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ and let $\mathbf{c} = \hat{\mathbf{Q}}^T\mathbf{b}$ be partitioned into two pieces:

- $\mathbf{c}_1$ of dimension n and
- $\mathbf{c}_2$ of dimension $m - n$.

Then

$$\begin{aligned}
\|\mathbf{r}\|^2 &= \|\mathbf{b} - \mathbf{A}\mathbf{x}\|^2 \\
&= \|\mathbf{b} - \hat{\mathbf{Q}}\hat{\mathbf{R}}\mathbf{x}\|^2 \\
&= \|\hat{\mathbf{Q}}^T[\mathbf{b} - \hat{\mathbf{Q}}\hat{\mathbf{R}}\mathbf{x}]\|^2 \\
&= \|\mathbf{c} - \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}\mathbf{x}\|^2 \\
&= \|\mathbf{c}_1 - \mathbf{R}\mathbf{x}\|^2 + \|\mathbf{c}_2 - \mathbf{0}\mathbf{x}\|^2 .
\end{aligned}$$

Now, no matter what $\mathbf{x}$ is, the second term remains unchanged.

If we want to minimize this quantity with respect to $\mathbf{x}$, what we need to do is to solve a least squares problem involving $\mathbf{c}_1$ and $\mathbf{R}$. If $\mathbf{R}$ is full rank, then we can make the first term zero, and the norm of the residual $\mathbf{r}$ is simply $\|\mathbf{c}_2\|$.

Algorithm

1. Compute the QR factorization of $\mathbf{A}$.
2. Form $\mathbf{c}_1 = \mathbf{Q}^T\mathbf{b}$.
3. Solve the square, triangular system $\mathbf{R}\mathbf{x} = \mathbf{c}_1$. (Solve it in the least squares sense, if $\mathbf{R}$ is rank deficient.)

4. If $\bar{\mathbf{Q}}$ is available and $\mathbf{R}$ is full rank, then the norm of the residual can be computed as $\|\bar{\mathbf{Q}}^T\mathbf{b}\|$.

   Otherwise the residual is computed as $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$.

Remaining computational issue:

How to compute the QR factorization

We use Givens rotations.

Computing a QR factorization (Given's rotations)

QR factorization by Givens rotations

A simple orthogonal matrix, a rotation, can be used to introduce one zero at a time into our matrix.

We'll write the Givens matrix as

$$\mathbf{G} = \begin{bmatrix} c & s \\ s & -c \end{bmatrix}$$

where $c^2 + s^2 = 1$. (Thus, c and s have the geometric interpretion of the cosine and sine of an angle.)

Note: Most sources put the negative sign in the (2,1) position rather than the (2,2) position. This has the disadvantage of making the matrix nonsymmetric. The way we have written it means that a vector multiplied by it is rotated through an angle and then reflected.

How Givens rotations can be used

Problem: Given a vector $\mathbf{z}$ of dimension 2×1, find $\mathbf{G}$ so that $\mathbf{G}\mathbf{z} = x\mathbf{e}_1$.

Solution:

$$\mathbf{G}\mathbf{z} = \begin{bmatrix} cz_1 + sz_2 \\ sz_1 - cz_2 \end{bmatrix} = x\mathbf{e}_1$$

Multiplying the first equation by c, the second by s, and adding yields

$$(c^2 + s^2)z_1 = cx\,,$$

so

$$c = z_1/x\,.$$

Similarly, we can determine that

$$s = z_2/x\,.$$

Since $c^2 + s^2 = 1$, we conclude that

$$z_1^2 + z_2^2 = x^2\,,$$

so

$$\begin{aligned} c &= \frac{z_1}{\sqrt{z_1^2 + z_2^2}} \\ s &= \frac{z_2}{\sqrt{z_1^2 + z_2^2}} \end{aligned}$$

Givens QR factorization

So now we know how to use Givens matrices to zero out single components of a matrix. We will use the notation $\mathbf{G}_{ij}$ to denote an $n \times n$ identity matrix with its ith and jth rows modified to include the Givens rotation: for example, if $n = 6$, then

$$\mathbf{G}_{25} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & 0 & s & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & s & 0 & 0 & -c & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

and multiplication of a vector by this matrix leaves all but rows 2 and 5 of the vector unchanged.

So we can reduce a matrix $\mathbf{A}$ to upper trapezoidal form by the following sequence of rotations:

```
for $i = 1, \ldots, n$,
    for $j = i + 1, \ldots, m$,
        Choose the matrix $\mathbf{G}_{ij}$ to zero out position $(j, i)$ of the
        matrix, using the current value in position $(i, i)$:
            $\mathbf{A} \leftarrow \mathbf{G}_{ij}\mathbf{A}$
    end for
```

end for

A nice feature of the QR factorization: in general, we don't need to "pivot" to preserve numerical stability.

This makes QR a nice alternative to LU for solving linear systems. Although the operations count is twice as big, the data handling is simpler.

However, if the columns of $\mathbf{A}$ are linearly dependent, or are close to being linearly dependent, then the QR factorization does not behave well.

In that case, we need to use a more expensive but completely reliable algorithm, the singular value decomposition. We won't discuss it in these notes, but it available in Matlab as `svd`.

Matlab Least Squares

The Matlab backslash command, `x = A \ b`, which solves a linear system when $\mathbf{A}$ is square and nonsingular, solves the problem in the least squares sense when $\mathbf{A}$ has more rows than columns.

Examples

An example: our straight line, revisited

Problem: Fit a model to data in order to reduce the effects of noise in the measurements.

Recall the data from the polynomial section. Is a straight line a good fit to the data (t_i, f_i), $i = 1, \ldots, 10$?

Data:

$$\mathbf{A} = \begin{bmatrix} 1 & t_1 \\ . & . \\ . & . \\ . & . \\ 1 & t_{10} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} f_1 \\ . \\ . \\ . \\ f_{10} \end{bmatrix}$$

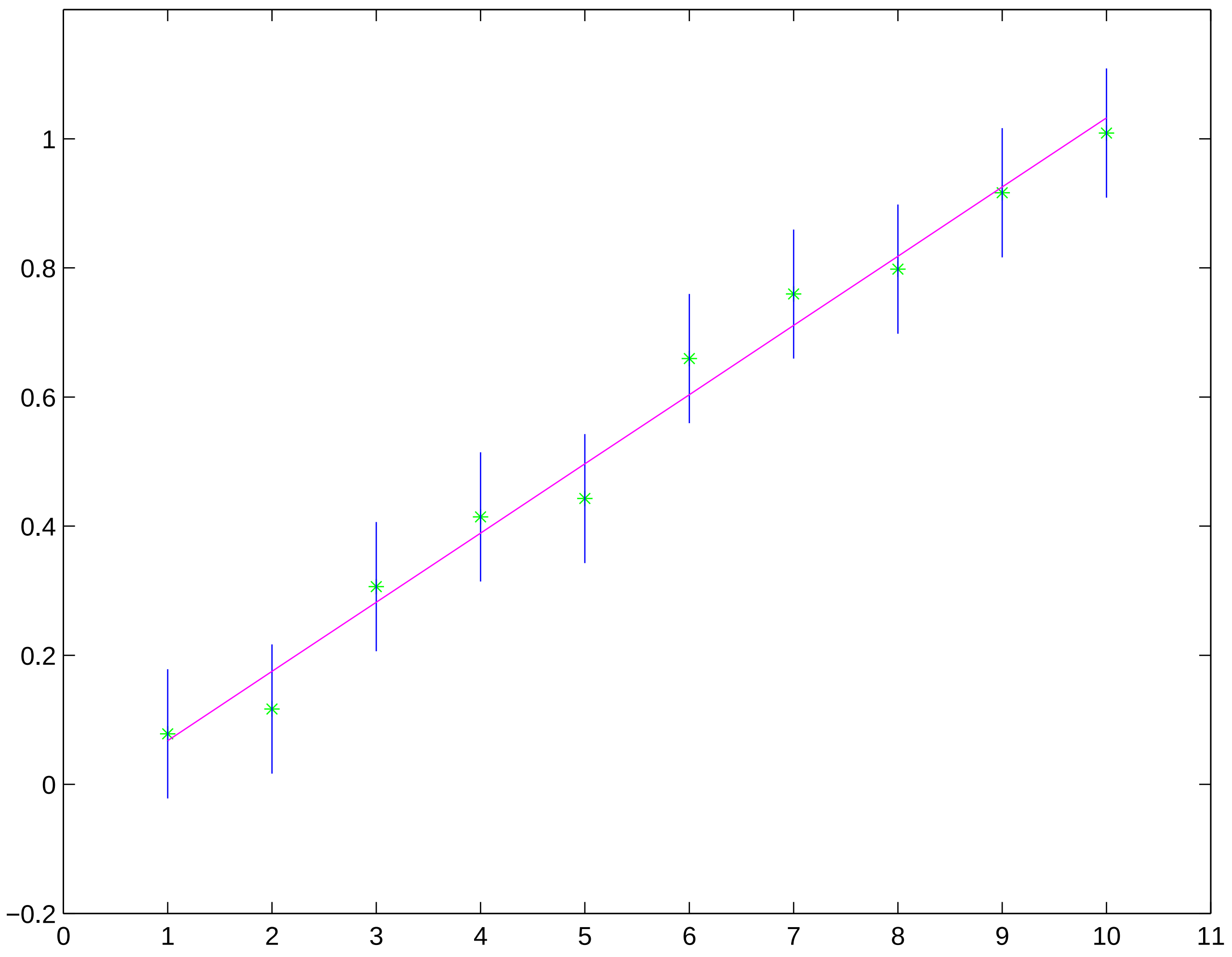

Figure 2: Is a straight line a good model to this data?

```
sigma=.05
t = [1:10];
ve = ...
plot(t,ve,'g*')
hold on
for i=1:10,
    plot([t(i),t(i)],[ve(i)+2*sigma,ve(i)-2*sigma])
end
axis([0 11 -.2 1.2])
a = [ones(10,1),t'];
coef = a \ ve';
plot(t,a*coef,'m')
```

A second example: sensitivity can hurt

Suppose we want to fit a polynomial to some data and suppose that we are ignorant enough to use the power basis.

Our data is taken at $1, 2, \ldots, 30$.

A Matlab program:

```
% Compute the condition numbers of the power-basis matrix
% for various degrees n-1 of the polynomial

t=[1:30]';
m = length(t)
disp('    n        cond(A)')
for n=2:10,
a=ones(m ,1);
for i=2:n,
   a = [a, a(:,i-1).*t];
end
s = sprintf('%5d %15.5e',n,cond(a));
disp(s)
end

pause

% Perform a fit for a polynomial of degree 8
% assuming that the true data is from a polynomial
% with coefficients 1, 2, ..., 9, and some
% random noise (mean zero, standard deviation 1)
% has been added.

n=9
a=ones(m ,1);
```

```
for i=2:n,
   a = [a, a(:,i-1).*t];
end

v = a * [1:n]' + rand(m ,1);
format long
coef1 = a \ v;
coef2 = (a'*a) \ (a'*v);
disp('          QR       normal equations')
disp([coef1,coef2])
```

Results:

```
    n        cond(A)
    2     3.65007e+01
    3     1.35936e+03
    4     5.07537e+04
    5     1.93735e+06
    6     7.68136e+07
    7     3.18589e+09
    8     1.38024e+11
    9     6.21454e+12
   10     2.88547e+14
```

```
          QR       normal equations
   1.0e+02 *

   0.02592099356981  -2.35878463268852
   0.00880868395348   3.56794884601309
   0.03422551695005  -1.66156270129563
   0.03923888065992   0.40975832216991
   0.05007398826829   0.00699043677858
   0.05999594170322   0.06283289495303
   0.07000012452822   0.06989408241536
   0.07999999802536   0.08000209368768
   0.09000000001244   0.08999998301781
```