Least Squares Fitting of Data to a Curve

Gerald Recktenwald
Portland State University
Department of Mechanical Engineering
gerry@me.pdx.edu

These slides are a supplement to the book *Numerical Methods with* MATLAB*: Implementations and Applications*, by Gerald W. Recktenwald, © 2000–2007, Prentice-Hall, Upper Saddle River, NJ.

The latest version of this PDF file, along with other supplemental material for the book, can be found at `www.prenhall.com/recktenwald` or `web.cecs.pdx.edu/~gerry/nmm/`.

Version 0.82 November 6, 2007

Overview

- Fitting a line to data
  - ▷ Geometric interpretation
  - ▷ Residuals of the overdetermined system
  - ▷ The normal equations
- Nonlinear fits via coordinate transformation
- Fitting arbitrary linear combinations of basis functions
  - ▷ Mathematical formulation
  - ▷ Solution via normal equations
  - ▷ Solution via QR factorization
- Polynomial curve fits with the built-in `polyfit` function
- Multivariate fitting

Fitting a Line to Data

Given m pairs of data:

$$(x_i, y_i),\ i = 1, \ldots, m$$

Find the coefficients α and β such that

$$F(x) = \alpha x + \beta$$

is a good fit to the data

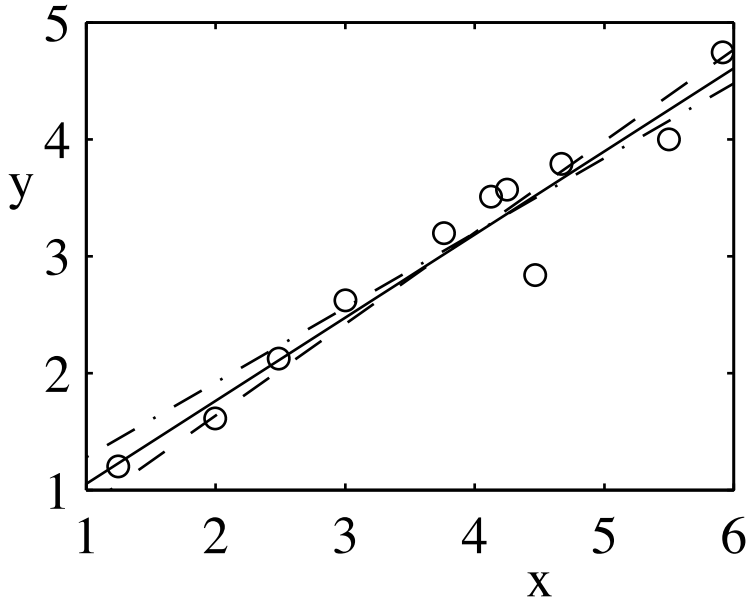
Questions:

- How do we define good fit?
- How do we compute α and β after a definition of "good fit" is obtained?

Plausible Fits

Plausible fits are obtained by adjusting the slope (α) and intercept (β). Here is a graphical representation of potential fits to a particular set of data

Which of the lines provides the best fit?



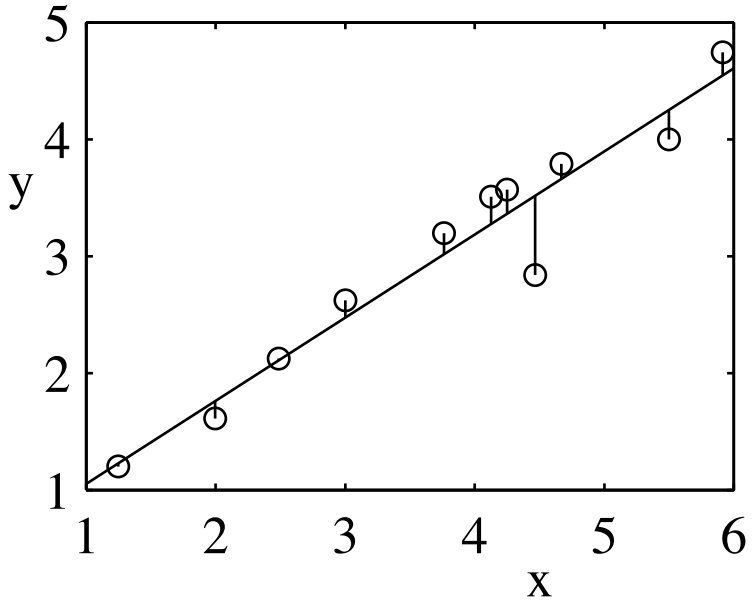
The Residual

The difference between the given y_i value and the fit function evaluated at x_i is

$$\begin{aligned} r_i &= y_i - F(x_i) \\ &= y_i - (\alpha x_i + \beta) \end{aligned}$$

r_i is the *residual* for the data pair (x_i, y_i).

r_i is the vertical distance between the known data and the fit function.



Minimizing the Residual

Two criteria for choosing the "best" fit

$$\text{minimize} \sum |r_i| \qquad \text{or} \qquad \text{minimize} \sum r_i^2$$

For statistical and computational reasons choose minimization of $\rho = \sum r_i^2$

$$\rho = \sum_{i=1}^{m} \left[y_i - (\alpha x_i + \beta) \right]^2$$

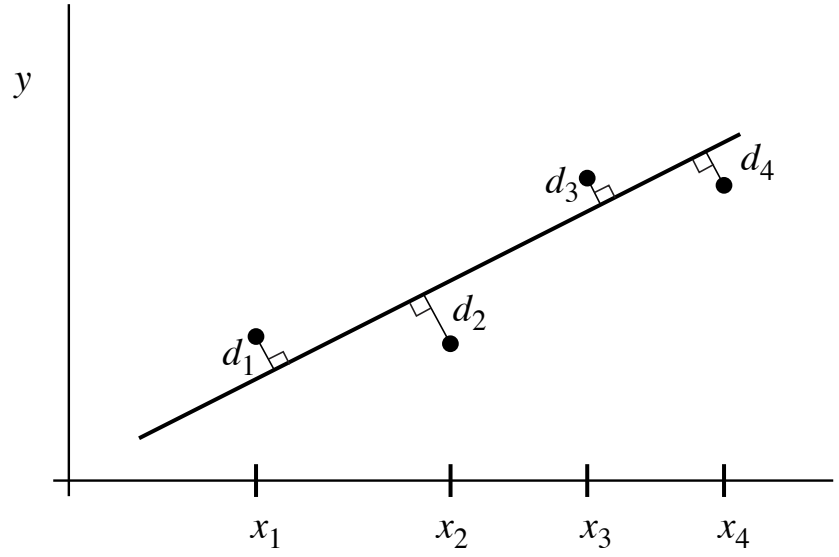
The best fit is obtained by the values of α and β that minimize ρ.

Orthogonal Distance Fit

An alternative to minimizing the residual is to minimize the orthogonal distance to the line.

Minimizing $\sum d_i^2$ is known as the *Orthogonal Distance Regression* problem.

See, e.g., Åke Björk, *Numerical Methods for Least Squares Problems*, 1996, SIAM, Philadelphia.



Least Squares Fit (1)

The *least squares fit* is obtained by choosing the α and β so that

$$\sum_{i=1}^{m} r_i^2$$

is a minimum. Let $\rho = \|r\|_2^2$ to simplify the notation.

Find α and β by minimizing $\rho = \rho(\alpha, \beta)$. The minimum requires

$$\left.\frac{\partial \rho}{\partial \alpha}\right|_{\beta=\text{constant}} = 0$$

and

$$\left.\frac{\partial \rho}{\partial \beta}\right|_{\alpha=\text{constant}} = 0$$

Least Squares Fit (2)

Carrying out the differentiation leads to

$$S_{xx}\alpha + S_x\beta = S_{xy} \quad (1)$$

$$S_x\alpha + m\beta = S_y \quad (2)$$

where

$$S_{xx} = \sum_{i=1}^{m} x_i x_i \qquad S_x = \sum_{i=1}^{m} x_i$$

$$S_{xy} = \sum_{i=1}^{m} x_i y_i \qquad S_y = \sum_{i=1}^{m} y_i$$

Note: S_{xx}, S_x, S_{xy}, and S_{yy} can be directly computed from the given (x_i, y_i) data. Thus, Equation (1) and (2) are two equations for the two unknowns, α and β.

Least Squares Fit (3)

Solving equations (1) and (2) for α and β yields

$$\alpha = \frac{1}{d}\left(S_x S_y - m S_{xy}\right) \tag{3}$$

$$\beta = \frac{1}{d}\left(S_x S_{xy} - S_{xx} S_y\right) \tag{4}$$

with

$$d = S_x^2 - m S_{xx} \tag{5}$$

Overdetermined System for a Line Fit (1)

Now, let's rederive the equations for the fit. This will give us insight into the process or fitting arbitrary linear combinations of functions.

For any two points we can write

$$\alpha x_1 + \beta = y_1$$
$$\alpha x_2 + \beta = y_2$$

or

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

But why just pick two points?

Overdetermined System for a Line Fit (2)

Writing out the $\alpha x + \beta = y$ equation for *all* of the known points (x_i, y_i), $i = 1, \ldots, m$ gives the *overdetermined* system.

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad \text{or} \qquad Ac = y$$

where

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix} \qquad c = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Note: We cannot solve $Ac = y$ with Gaussian elimination. Unless the system is consistent (i.e., unless y lies in the column space of A) it is impossible to find the $c = (\alpha, \beta)^T$ that exactly satisfies all m equations. The system is consistent *only* if all the data points lie along a single line.

Normal Equations for a Line Fit

Compute $\rho = \|r\|_2^2$, where $r = y - Ac$

$$\begin{aligned}\rho = \|r\|_2^2 = r^T r &= (y - Ac)^T (y - Ac) \\ &= y^T y - (Ac)^T y - y^T (Ac) + c^T A^T Ac \\ &= y^T y - 2y^T Ac + c^T A^T Ac.\end{aligned}$$

Minimizing ρ requires

$$\frac{\partial \rho}{\partial c} = -2A^T y + 2A^T Ac = 0$$

or

$$(A^T A)c = A^T b$$

This is the matrix formulation of equations (1) and (2).

linefit.m

The **linefit** function fits a line to a set of data by solving the normal equations.

```
function [c,R2] = linefit(x,y)
% linefit    Least-squares fit of data to y = c(1)*x + c(2)
%
% Synopsis:   c     = linefit(x,y)
%            [c,R2] = linefit(x,y)
%
% Input:   x,y = vectors of independent and dependent variables
%
% Output:  c  = vector of slope, c(1), and intercept, c(2) of least sq. line fit
%          R2 = (optional) coefficient of determination; 0 <= R2 <= 1
%               R2 close to 1 indicates a strong relationship between y and x
if length(y)~= length(x),  error('x and y are not compatible');  end

x = x(:);  y = y(:);    %  Make sure that x and y are column vectors
A = [x ones(size(x))];  %  m-by-n matrix of overdetermined system
c = (A'*A)\(A'*y);      %  Solve normal equations
if nargout>1
  r = y - A*c;
  R2 = 1 - (norm(r)/norm(y-mean(y)))^2;
end
```

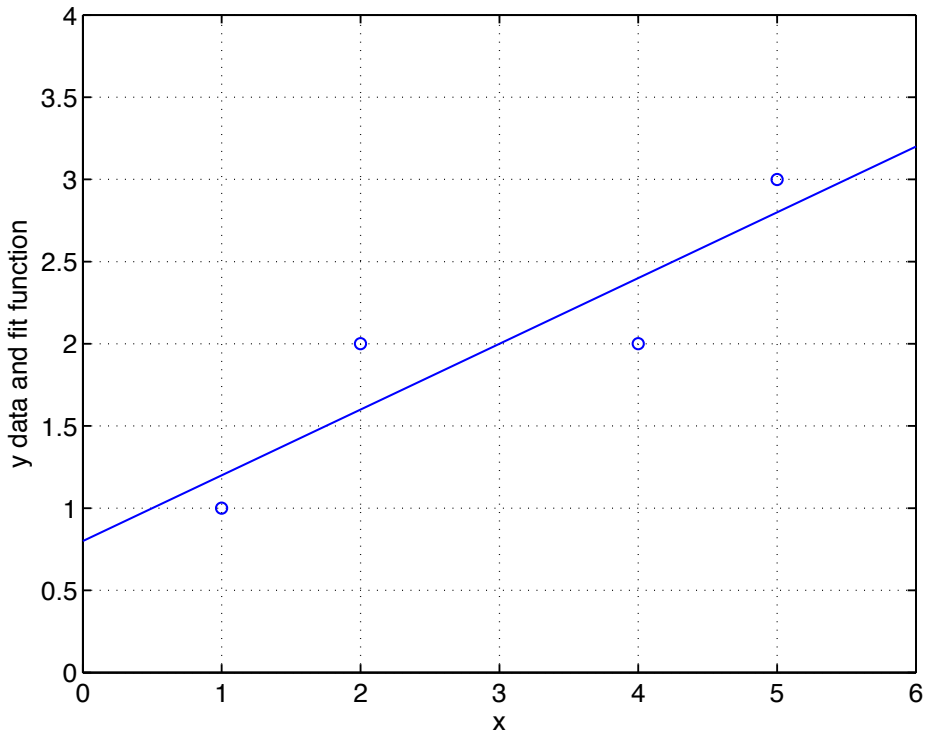
Line Fitting Example

Store data and perform the fit

```
>> x = [1 2 4 5];
>> y = [1 2 2 3];
>> c = linefit(x,y)
c =
    0.4000
    0.8000
```

Evaluate and plot the fit

```
>> xfit = linspace(min(x),max(x));
>> yfit = c(1)*xfit + c(2)
>> plot(x,y,'o',xfit,yfit,'-');
```



R^2 Statistic (1)

R^2 is a measure of how well the fit function follows the trend in the data. $0 \leq R^2 \leq 1$.

Define:

$\hat{y}$	is the value of the fit function at the known data points.	For a line fit $\hat{y}_i = c_1 x_i + c_2$
$\bar{y}$	is the average of the y values	$\bar{y} = \frac{1}{m} \sum y_i$

Then:

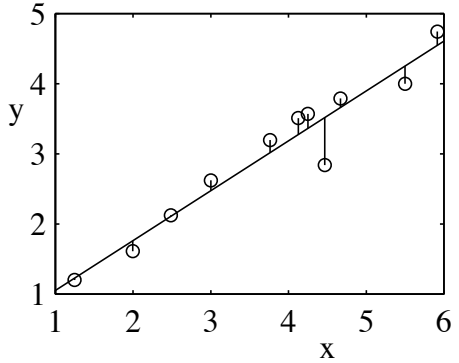
$$R^2 = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2} = 1 - \frac{\|r\|_2^2}{\sum (y_i - \bar{y})^2}$$

When $R^2 \approx 1$ the fit function follows the trend of the data.
When $R^2 \approx 0$ the fit is not significantly better than approximating the data by its mean.

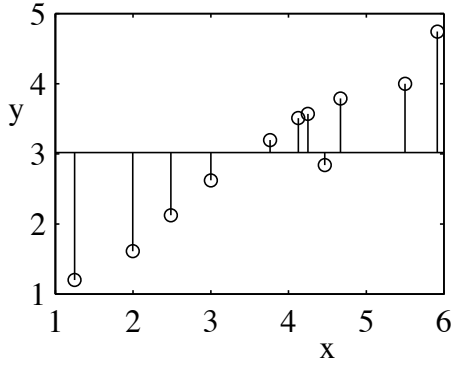
Graphical Interpretation of the R^2 Statistic

Consider a line fit to a data set with $R^2 = 1 - \dfrac{\|r\|_2^2}{\sum(y_i - \bar{y})^2} = 0.934$

Vertical distances between given y data and the least squares line fit. Vertical lines show contributions to $\|r\|_2$.



Vertical distances between given y data and the average of the y. Vertical lines show contributions to $\sum(y_i - \bar{y})^2$

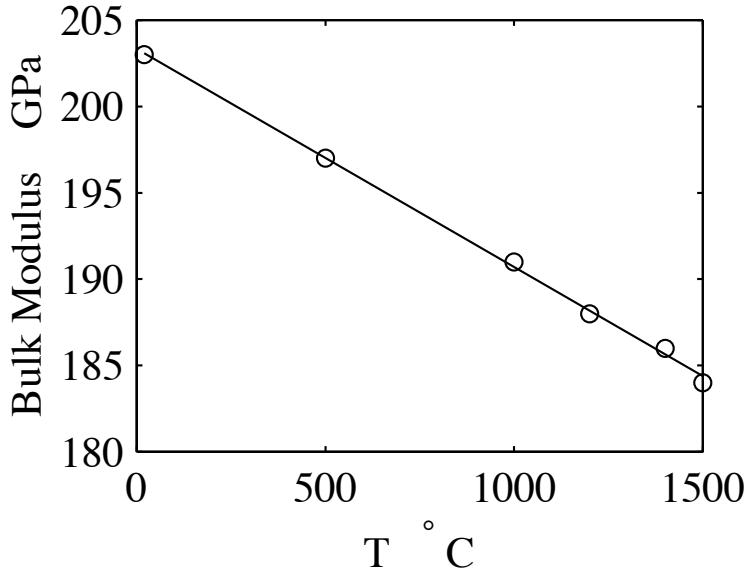


R^2 Statistic: Example Calculation

Consider the variation of the bulk modulus of Silicon Carbide as a function of temperature (Cf. Example 9.4)

T ($^\circ$C)	20	500	1000	1200	1400	1500
G (GPa)	203	197	191	188	186	184

```
>> [t,D,labels] = loadColData('SiC.dat',6,5);
>> g = D(:,1);
>> [c,R2] = linefit(t,g);
c =
    -0.0126
    203.3319
R2 =
    0.9985
```



Fitting Transformed Non-linear Functions (1)

- Some nonlinear fit functions $y = F(x)$ can be transformed to an equation of the form $v = \alpha u + \beta$
- Linear least squares fit to a line is performed on the transformed variables.
- Parameters of the nonlinear fit function are obtained by transforming back to the original variables.
- The linear least squares fit to the transformed equations does not yield the same fit coefficients as a direct solution to the nonlinear least squares problem involving the original fit function.

Examples:

$$y = c_1 e^{c_2 x} \quad \longrightarrow \quad \ln y = \alpha x + \beta$$

$$y = c_1 x^{c_2} \quad \longrightarrow \quad \ln y = \alpha \ln x + \beta$$

$$y = c_1 x e^{c_2 x} \quad \longrightarrow \quad \ln(y/x) = \alpha x + \beta$$

Fitting Transformed Non-linear Functions (2)

Consider

$$y = c_1 e^{c_2 x} \tag{6}$$

Taking the logarithm of both sides yields

$$\ln y = \ln c_1 + c_2 x$$

Introducing the variables

$$v = \ln y \quad b = \ln c_1 \quad a = c_2$$

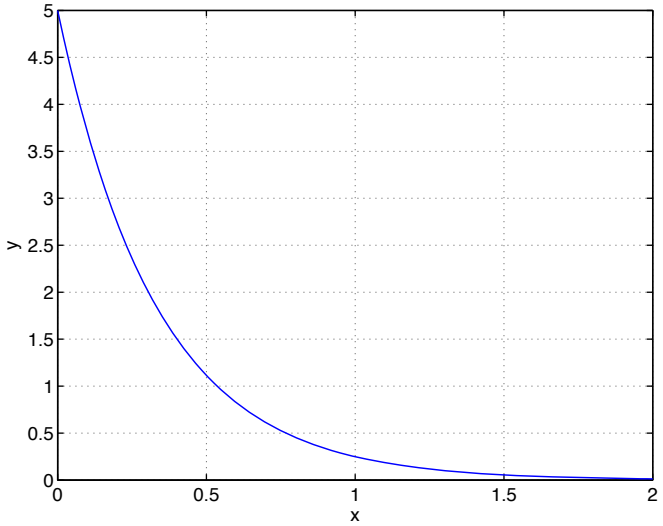
transforms equation (6) to

$$v = ax + b$$

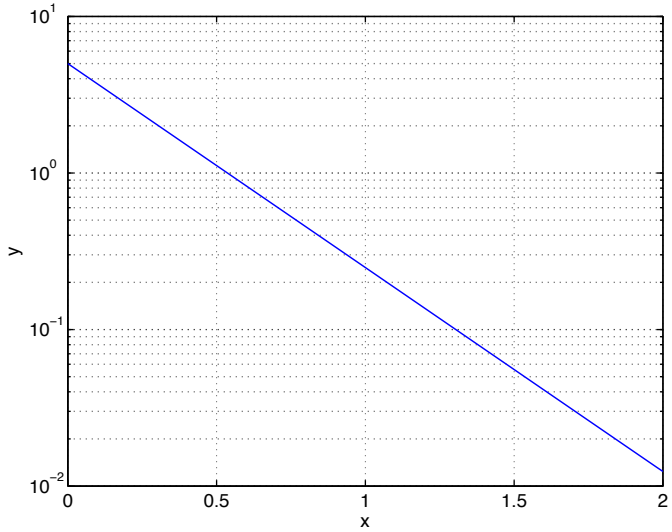
Fitting Transformed Non-linear Functions (3)

The preceding steps are equivalent to graphically obtaining c_1 and c_2 by plotting the data on semilog paper.

$$y = c_1 e^{c_2 x}$$



$$\ln y = c_2 x + \ln c_1$$



Fitting Transformed Non-linear Functions (4)

Consider $y = c_1 x^{c_2}$. Taking the logarithm of both sides yields

$$\ln y = \ln c_1 + c_2 \ln x \tag{7}$$

Introduce the transformed variables

$$v = \ln y \quad u = \ln x \quad b = \ln c_1 \quad a = c_2$$

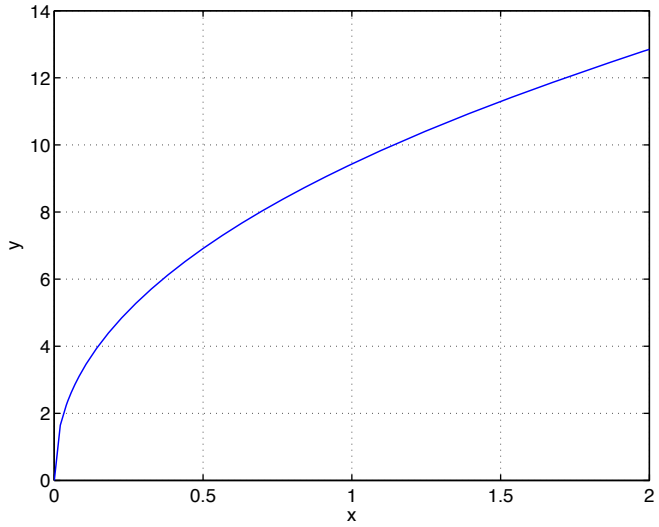
and equation (7) can be written

$$v = au + b$$

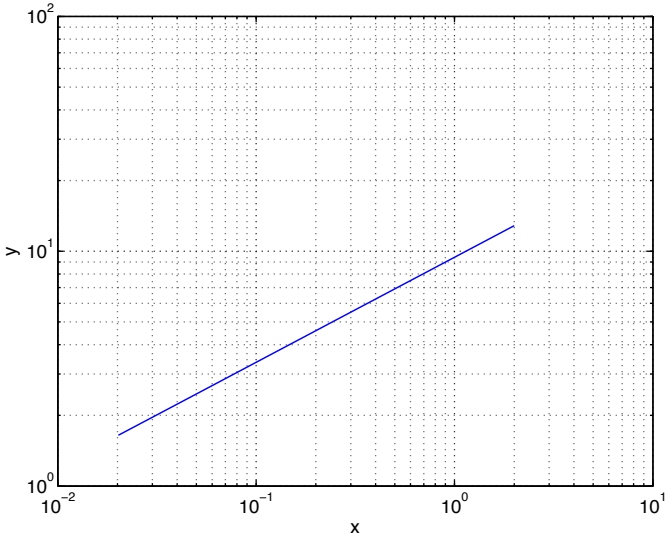
Fitting Transformed Non-linear Functions (5)

The preceding steps are equivalent to graphically obtaining c_1 and c_2 by plotting the data on log-log paper.

$$y = c_1 x^{c_2}$$



$$\ln y = c_2 \ln x + \ln c_1$$



Example: Fitting Data to $y = c_1 x e^{c_2 x}$

Consider $y = c_1 x e^{c_2 x}$. The transformation

$$v = \ln\left(\frac{y}{x}\right) \qquad a = c_2 \qquad b = \ln c_1$$

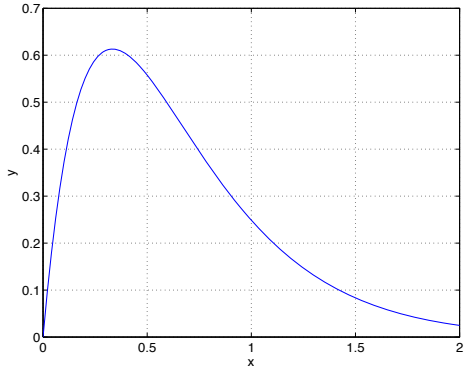
results in the linear equation

$$v = ax + b$$

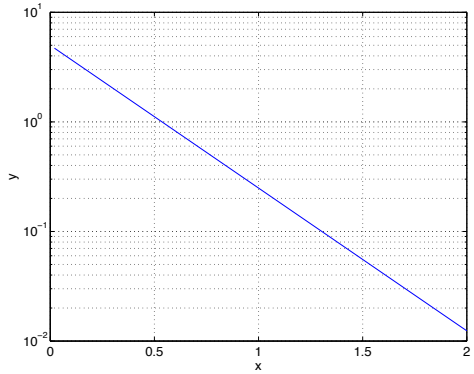
Fitting Transformed Non-linear Functions (6)

The preceding steps are equivalent to graphically obtaining c_1 and c_2 by plotting the data on semilog paper.

$$y = c_1 x e^{c_2 x}$$



$$\ln(y/x) = c_2 x + \ln c_1$$



xexpfit.m

The `xexpfit` function uses a linearizing transformation to fit $y = c_1 x e^{c_2 x}$ to data.

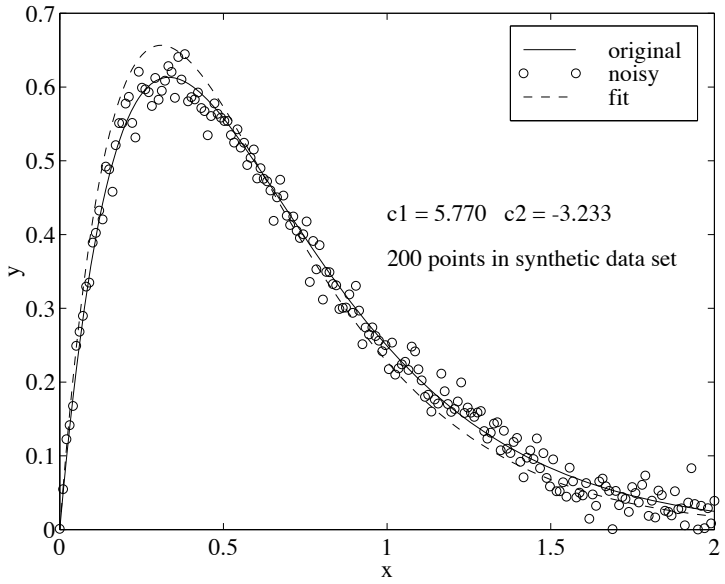
```
function c = xexpfit(x,y)
% xexpfit    Least squares fit of data to y = c(1)*x*exp(c(2)*x)
%
% Synopsis:  c = xexpfit(x,y)
%
% Input:     x,y = vectors of independent and dependent variable values
%
% Output:    c = vector of coefficients of  y = c(1)*x*exp(c(2)*x)

z = log(y./x);              %  Natural log of element-by-element division
c = linefit(x,z);           %  Fit is performed by linefit
c = [exp(c(2)); c(1)];      %  Extract parameters from transformation
```

Example: Fit Synthetic Data

Fit $y = c_1 x e^{c_2 x}$ to synthetic data. See `demoXexp`

```
>> %  Synthetic data with noise, avoid x=0
>> x0 = 0.01;
>> noise = 0.05;
>> x = linspace(x0,2,200);
>> y = 5*x.*exp(-3*x);
>> yn = y + noise*(rand(size(x))-0.5);
>> %  Guarantee yn>0 for log(yn)
>> yn = abs(yn);
>> c = xexpfit(x,yn);
c =
    5.7701
   -3.2330
```



Summary of Transformations

- Transform (x, y) data as needed
- Use `linefit`
- Transform results of `linefit` back

```
>> x = ...            %  original data
>> y = ...
>> u = ...            %  transform the data
>> v = ...
>> a = linefit(u,v)
>> c = ...            %  transform the coefficients
```

Summary of Line Fitting (1)

1. m data pairs are given: (x_i, y_i), $i = 1, \ldots, m$.
2. The fit function $y = F(x) = c_1 x + c_2$ has $n = 2$ basis functions $f_1(x) = x$ and $f_2(x) = 1$.
3. Evaluating the fit function for each of the m data points gives an overdetermined system of equations $Ac = y$ where $c = [c_1, c_2]^T$, $y = [y_1, y_2, \ldots, y_m]^T$, and

$$A = \begin{bmatrix} f_1(x_1) & f_2(x_1) \\ f_1(x_2) & f_2(x_2) \\ \vdots & \vdots \\ f_1(x_m) & f_2(x_m) \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix}.$$

Summary of Line Fitting (2)

4. The least-squares principle defines the best fit as the values of c_1 and c_2 that minimize

$$\rho(c_1, c_2) = \|y - F(x)\|_2^2 = \|y - Ac\|_2^2.$$

5. Minimizing of $\rho(c_1, c_2)$ leads to the normal equations

$$(A^T A)c = A^T y,$$

6. Solving the normal equations gives the slope c_1 and intercept c_2 of the best fit line.

Fitting Linear Combinations of Functions

- Definition of fit function and basis functions
- Formulation of the overdetermined system
- Solution via normal equations: `fitnorm`
- Solution via QR factorization: `fitqr` and `\`

Fitting Linear Combinations of Functions (1)

Consider the fitting function

$$F(x) = cf_1(x) + c_2 f_2(x) + \ldots + c_n f_k(x)$$

or

$$F(x) = \sum_{j=1}^{n} c_j f_j(x)$$

The basis functions

$$f_1(x), f_2(x), \ldots, f_n(x)$$

are chosen by you — the person making the fit.

The coefficients

$$c_1, c_2, \ldots, c_n$$

are determined by the least squares method.

Fitting Linear Combinations of Functions (2)

$F(x)$ function can be any combination of functions that are linear in the c_j. Thus

$$1,\ x,\ x^2,\ x^{2/3},\ \sin x,\ e^x,\ xe^{4x},\ \cos(\ln 25x)$$

are all valid basis functions. On the other hand,

$$\sin(c_1 x),\ e^{c_3 x},\ x^{c_2}$$

are not valid basis functions as long as the c_j are the parameters of the fit.

For example, the fit function for a cubic polynomial is

$$F(x) = c_1 x^3 + c_2 x^2 + c_3 x + c_4,$$

which has the basis functions

$$x^3,\ x^2,\ x,\ 1.$$

Fitting Linear Combinations of Functions (3)

The objective is to find the c_j such that $F(x_i) \approx y_i$.

Since $F(x_i) \neq y_i$, the residual for each data point is

$$r_i = y_i - F(x_i) = y_i - \sum_{j=1}^{n} c_j f_j(x_i)$$

The least-squares solution gives the c_j that minimize $\|r\|_2$.

Fitting Linear Combinations of Functions (4)

Consider the fit function with three basis functions

$$y = F(x) = c_1 f_1(x) + c_2 f_2(x) + c_3 f_3(x).$$

Assume that $F(x)$ acts like an interpolant. Then

$$\begin{aligned} c_1 f_1(x_1) + c_2 f_2(x_1) + c_3 f_3(x_1) &= y_1, \\ c_1 f_1(x_2) + c_2 f_2(x_2) + c_3 f_3(x_2) &= y_2, \\ &\vdots \\ c_1 f_1(x_m) + c_2 f_2(x_m) + c_3 f_3(x_m) &= y_m. \end{aligned}$$

are all satisfied.

For a least squares fit, the equations are *not* all satisfied, i.e., the fit function $F(x)$ does not pass through the y_i data.

Fitting Linear Combinations of Functions (5)

The preceding equations are equivalent to the overdetermined system

$$Ac = y,$$

where

$$A = \begin{bmatrix} f_1(x_1) & f_2(x_1) & f_3(x_1) \\ f_1(x_2) & f_2(x_2) & f_3(x_2) \\ \vdots & \vdots & \vdots \\ f_1(x_m) & f_2(x_m) & f_3(x_m) \end{bmatrix},$$

$$c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

Fitting Linear Combinations of Functions (6)

If $F(x)$ cannot interpolate the data, then the preceding matrix equation cannot be solved exactly: b does not lie in the column space of A.

The least-squares method provides the compromise solution that minimizes $\|r\|_2 = \|y - Ac\|_2$.

The c that minimizes $\|r\|_2$ satisfies the normal equations

$$(A^T A)c = A^T y.$$

Fitting Linear Combinations of Functions (7)

In general, for n basis functions

$$A = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_n(x_2) \\ \vdots & \vdots & & \vdots \\ f_1(x_m) & f_2(x_m) & \dots & f_n(x_m) \end{bmatrix},$$

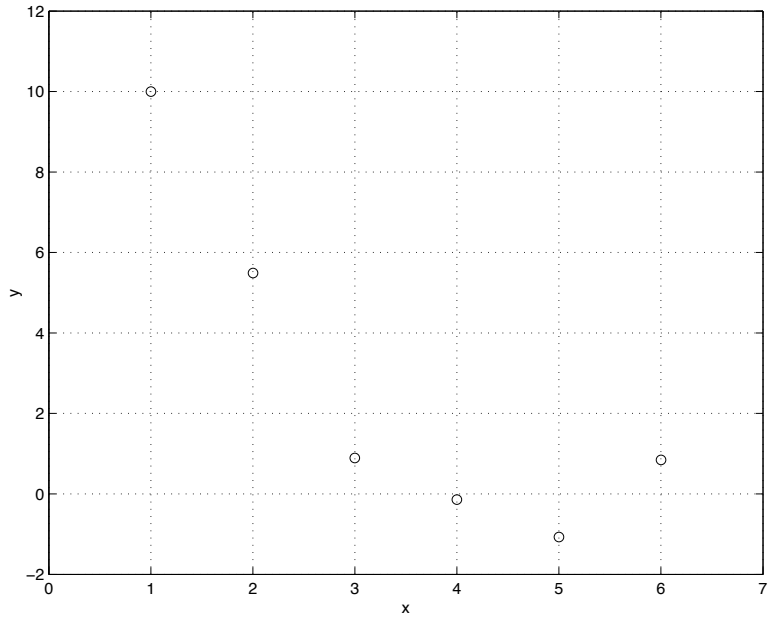
$$c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

Example: Fit a Parabola to Six Points (1)

Consider fitting a curve to the following data.

x	1	2	3	4	5	6
y	10	5.49	0.89	-0.14	-1.07	0.84

Not knowing anything more about the data we can start by fitting a polynomial to the data.



Example: Fit a Parabola to Six Points (2)

The equation of a second order polynomial can be written

$$y = c_1 x^2 + c_2 x + c_3$$

where the c_i are the coefficients to be determined by the fit and the basis functions are

$$f_1(x) = x^2, \qquad f_2(x) = x, \qquad f_3(x) = 1$$

The A matrix is

$$A = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_m^2 & x_m & 1 \end{bmatrix}$$

where, for this data set, $m = 6$.

Example: Fit a Parabola to Six Points (3)

Define the data

```
>> x = (1:6)';
>> y = [10  5.49  0.89  -0.14  -1.07  0.84]';
```

Notice the transposes, x and y must be column vectors.

The coefficient matrix of the overdetermined system is

```
>> A = [ x.^2  x  ones(size(x)) ];
```

The coefficient matrix for the normal equations is

```
>> disp(A'*A)
        2275         441          91
         441          91          21
          91          21           6
```

Example: Fit a Parabola to Six Points (4)

The right-hand-side vector for the normal equations is

```
>> disp(A'*y)
A'*y =
   41.2200
   22.7800
   16.0100
```

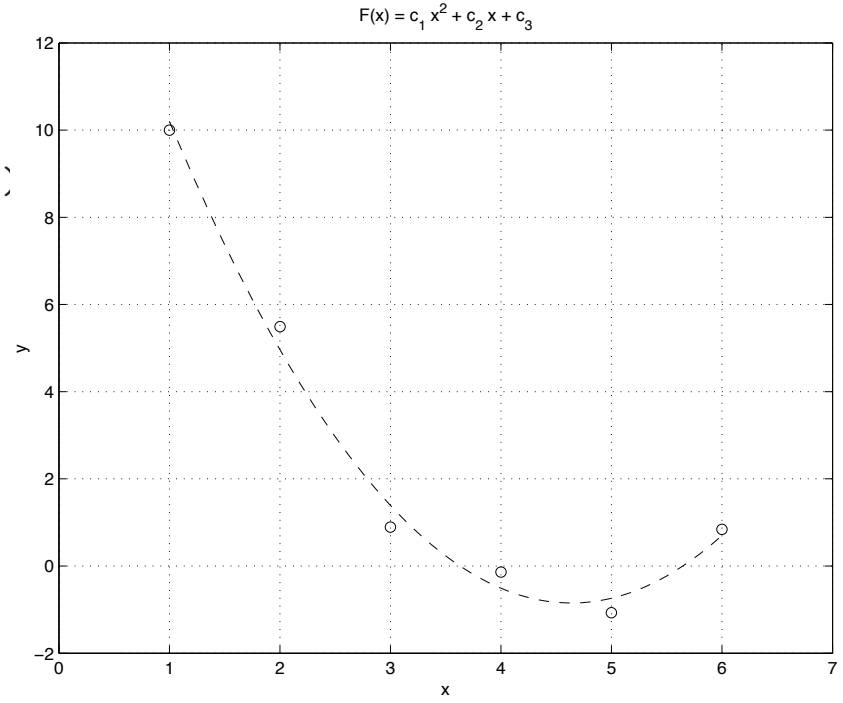
Solve the normal equations

```
>> c = (A'*A)\(A'*y)
c =
    0.8354
   -7.7478
   17.1160
```

Example: Fit a Parabola to Six Points (5)

Evaluate and plot the fit

```
>> xfit = linspace(min(x),max(x));
>> yfit = c(1)*xfit.^2 + c(2)*xfit + c(3)
>> plot(x,y,'o',xfit,yfit,'--');
```



Example: Alternate Fit to Same Six Points (1)

Fit the same points to

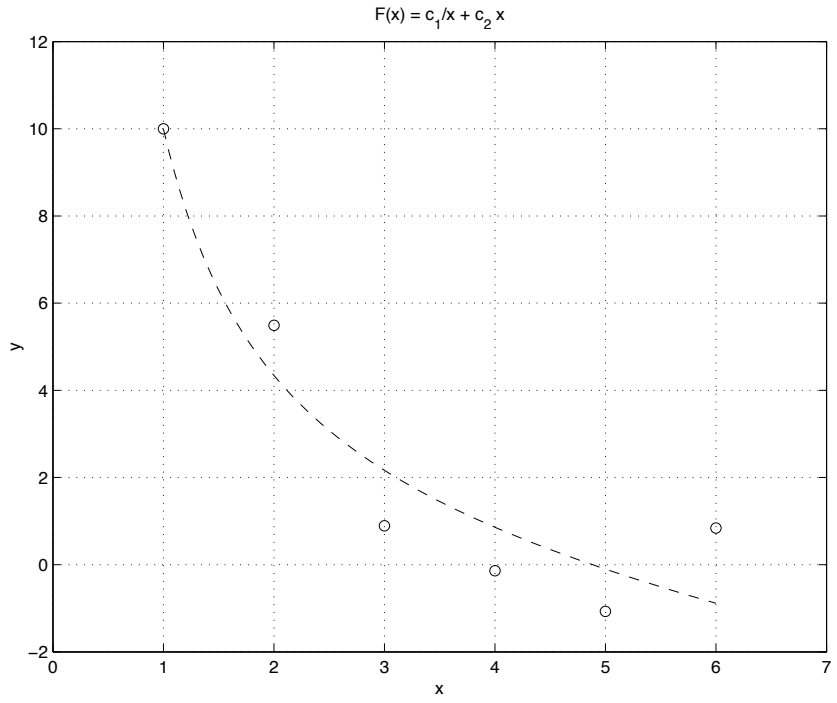
$$F(x) = \frac{c_1}{x} + c_2 x$$

The basis functions are

$$\frac{1}{x}, \quad x$$

In MATLAB:

```
>> x = (1:6)';
>> y = [10 5.49 0.89 -0.14 -1.07 0.84]';
>> A = [1./x  x];
>> c = (A'*A)\(A'*y)
```



Evaluating the Fit Function as a Matrix–Vector Product (1)

We have been writing the fit function as

$$y = F(x) = c_1 f_1(x) + c_2 f_2(x) + \cdots + c_n f_n(x)$$

The overdetermined coefficient matrix contains the basis functions evaluated at the known data

$$A = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_n(x_2) \\ \vdots & \vdots & & \vdots \\ f_1(x_m) & f_2(x_m) & \dots & f_n(x_m) \end{bmatrix}$$

Thus, if A is available

$$F(x) = Ac$$

evaluates $F(x)$ at all values of x, i.e., $F(x)$ is a vector-valued function.

Evaluating the Fit Function as a Matrix–Vector Product (2)

Evaluating the fit function as a matrix–vector product can be performed for any x. Suppose then that we have created an m-file function that evaluates A for any x, for example

```
function A = xinvxfun(x)
A = [ 1./x(:)  x(:) ];
```

We evaluate the fit coefficients with

```
>> x = ...,  y = ...
>> c = fitnorm(x,y,'xinvxfun');
```

Then, to plot the fit function after the coefficients of the fit

```
>> xfit = linspace(min(x),max(x));
>> Afit = xinvxfun(xfit);
>> yfit = Afit*c;
>> plot(x,y,'o',xfit,yfit,'--')
```

Evaluating the Fit Function as a Matrix–Vector Product (3)

Advantages:

- The basis functions are defined in only one place: in the routine for evaluating the overdetermined matrix.
- Automation of fitting and plotting is easier because all that is needed is one routine for evaluating the basis functions.
- End-user of the fit (not the person performing the fit) can still evaluate the fit function as $y = c_1 f_1(x) + c_2 f_2(x) + \cdots + c_n f_n(x)$.

Disadvantages:

- Storage of matrix A for large x vectors consumes memory. This should not be a problem for small n.
- Evaluation of the fit function may not be obvious to a reader unfamiliar with linear algebra.

MATLAB **Implementation in** `fitnorm`

Let A be the $m \times n$ matrix defined by

$$A = \begin{bmatrix} \vdots & \vdots & & \vdots \\ f_1(x) & f_2(x) & \dots & f_n(x) \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

The columns of A are the basis functions evaluated at each of the x data points.

As before, the normal equations are

$$A^T A c = A^T y$$

The user supplies a (usually small) m-file that returns A.

fitnorm.m

```
function [c,R2,rout] = fitnorm(x,y,basefun)
% fitnorm   Least-squares fit via solution to the normal equations
%
% Synopsis:  c       = fitnorm(x,y,basefun)
%           [c,R2]   = fitnorm(x,y,basefun)
%           [c,R2,r] = fitnorm(x,y,basefun)
%
% Input:   x,y     = vectors of data to be fit
%          basefun = (string) m-file that computes matrix A with columns as
%                    values of basis basis functions evaluated at x data points.
%
% Output:  c = vector of coefficients obtained from the fit
%          R2 = (optional) adjusted coefficient of determination; 0 <= R2 <= 1
%          r  = (optional) residuals of the fit

if length(y)~= length(x);  error('x and y are not compatible');  end
A = feval(basefun,x(:)); %  Coefficient matrix of overdetermined system
c = (A'*A)\(A'*y(:));    %  Solve normal equations, y(:) is always a column
if nargout>1
  r = y - A*c;           %  Residuals at data points used to obtain the fit
  [m,n] = size(A);
  R2 = 1 - (m-1)/(m-n-1)*(norm(r)/norm(y-mean(y)))^2;
  if nargout>2,  rout = r;  end
end
```

Example of User-Supplied m-files

The basis functions for fitting a parabola are

$$f_1(x) = x^2, \qquad f_2(x) = x, \qquad f_3(x) = 1$$

Create the m-file `poly2Basis.m`:

```
function A = poly2Basis(x)
A = [ x(:).^2  x(:)  ones(size(x(:)) ];
```

then at the command prompt

```
>> x = ...;  y = ...;
>> c = fitnorm(x,y,'poly2Basis')
```

or use an in-line function object:

```
>> x = ...;  y = ...;
>> Afun = inline('[ x(:).^2  x(:)  ones(size(x(:)) ]');
>> c = fitnorm(x,y,Afun);
```

Example of User-Supplied m-files

To the basis functions for fitting $F(x) = c_1/x + c_2 x$ are

$$\frac{1}{x}, \quad x$$

Create the m-file `xinvxfun.m`

```
function A = xinvxfun(x)
A = [ 1./x(:)  x(:) ];
```

then at the command prompt

```
>> x = ...;  y = ...;
>> c = fitnorm(x,y,'xinvxfun')
```

or use an in-line function object:

```
>> x = ...;  y = ...;
>> Afun = inline('[ 1./x(:)  x(:) ]');
>> c = fitnorm(x,y,Afun);
```

R^2 Statistic (1)

R^2 can be applied to linear combinations of basis functions.

Recall that for a line fit (Cf. § 9.1.4.)

$$R^2 = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2} = 1 - \frac{\|r\|_2^2}{\sum (y_i - \bar{y})^2}$$

where $\hat{y}_i$ is the value of the fit function evaluated at x_i, and $\bar{y}$ is the average of the (known) y values.

For a linear combination of basis functions

$$\hat{y}_i = \sum_{j=1}^{n} c_j f_j(x_i)$$

R^2 Statistic (2)

To account for the reduction in degrees of freedom in the data when the fit is performed, it is technically appropriate to consider the *adjusted coefficient of determination*

$$R^2_{\text{adjusted}} = 1 - \left(\frac{m-1}{m-n-1}\right)\frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2},$$

`fitnorm` provides the option of computing R^2_{adjusted}

Polynomial Curve Fits with `polyfit` (1)

Built-in commands for polynomial curve fits:

`polyfit`	Obtain coefficients of a least squares curve fit of a polynomial to a given data set
`polyval`	Evaluate a polynomial at a given set of x values.

Polynomial Curve Fits with `polyfit` (2)

Syntax:

```
c = polyfit(x,y,n)
[c,S] = polyfit(x,y,n)
```

x and *y* define the data
n is the desired degree of the polynomial.

c is a vector of polynomial coefficients stored in order of descending powers of x

$$p(x) = c_1 x^n + c_2 x^{n-1} + \cdots + c_n x + c_{n+1}$$

S is an optional return argument for `polyfit`. *S* is used as input to `polyval`

Polynomial Curve Fits with `polyfit` (3)

Evaluate the polynomial with `polyval`

Syntax:

```
yf = polyval(c,xf)
[yf,dy] = polyval(c,xf,S)
```

c contains the coefficients of the polynomial (returned by `polyfit`)

xf is a scalar or vector of x values at which the polynomial is to be evaluated

yf is a scalar or vector of values of the polynomials: *yf*$= p($*xf*$)$.

If *S* is given as an optional input to `polyval`, then *dy* is a vector of estimates of the uncertainty in *yf*

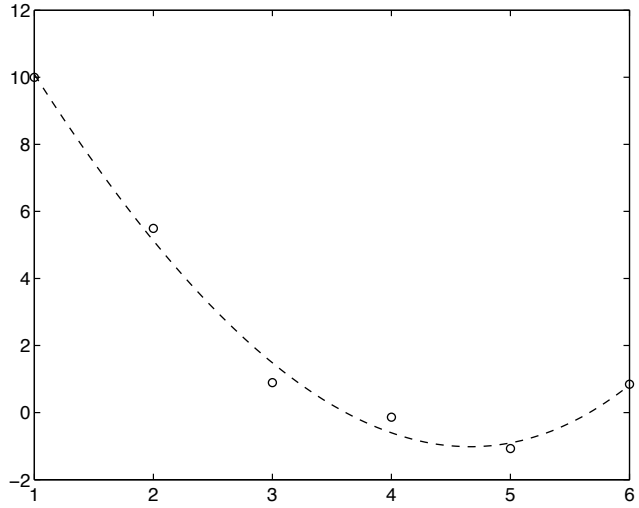
Example: Polynomial Curve Fit (1)

Fit a polynomial to Consider fitting a curve to the following data.

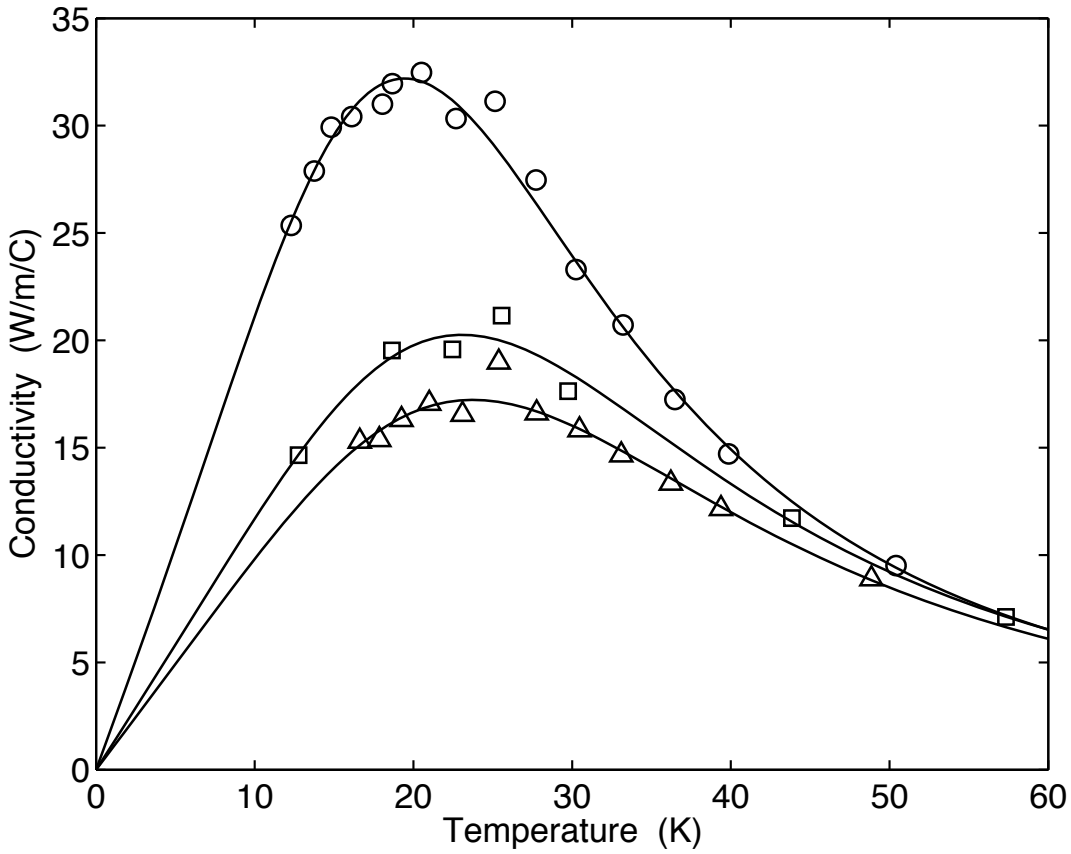
x	1	2	3	4	5	6
y	10	5.49	0.89	-0.14	-1.07	0.84

In MATLAB:

```
>> x = (1:6)';
>> y = [10  5.49  0.89  -0.14  -1.07  0.84]';
>> c = polyfit(x,y,3);
>> xfit = linspace(min(x),max(x));
>> yfit = polyval(c,xfit);
>> plot(x,y,'o',xfit,yfit,'--')
```


12
10
8
6
4
2
0
−2
1
2
3
4
5
6

Example: Conductivity of Copper Near $0\ K$ (1)



Example: Conductivity of Copper Near $0\ K$ (2)

Theoretical model of conductivity is

$$k(T) = \frac{1}{\frac{c_1}{T} + c_2 T^2}$$

To fit using linear least squares we need to write this as

$$\gamma(T) = \frac{1}{k(T)} = \frac{c_1}{T} + c_2 T^2$$

which has the basis functions

$$\frac{1}{T}, \qquad T^2$$

Example: Conductivity of Copper Near $0\ K$ (3)

The m-file implementing these basis functions is

```
function y = cuconBasis1(x)
% cuconBasis1  Basis fcns for conductivity model:  1/k = c1/T + c2*T^2
y = [1./x  x.^2];
```

An m-file that uses `fitnorm` to fit the conductivity data with the `cuconBasis1` function is listed on the next page.

Example: Conductivity of Copper Near $0\ K$ (4)

```
function conductFit(fname)
% conductFit  LS fit of conductivity data for Copper at low temperatures
%
% Synopsis:  conductFit(fname)
%
% Input:  fname   = (optional, string) name of data file;
%                   Default: fname = 'conduct1.dat'
%
% Output:  Print out of curve fit coefficients and a plot comparing data
%          with the curve fit for two sets of basis functions.

if nargin<1,  fname = 'cucon1.dat';   end   %  Default data file

% --- define basis functions as inline function objects
fun1 = inline('[1./t  t.^2]');        %  t must be a column vector
fun2 = inline('[1./t  t  t.^2]');

% --- read data and perform the fit
[t,k] = loadColData(fname,2,0,2);     %  Read data into t and k
[c1,R21,r1] = fitnorm(t,1./k,fun1);   %  Fit to first set of bases
[c2,R22,r2] = fitnorm(t,1./k,fun2);   %  and second set of bases
```

```
% --- print results
fprintf('\nCurve fit to data in %s\n\n',fname);
fprintf(' Coefficients of      Basis Fcns 1       Basis Fcns 2\n');
fprintf('      T^(-1)       %16.9e   %16.9e\n',c1(1),c2(1));
fprintf('      T            %16.9e   %16.9e\n',0,c2(2));
fprintf('      T^2          %16.9e   %16.9e\n',c1(2),c2(3));
fprintf('\n    ||r||_2        %12.5f      %12.5f\n',norm(r1),norm(r2));
fprintf('      R2           %12.5f      %12.5f\n',R21,R22);

% --- evaluate and plot the fits
tf = linspace(0.1,max(t))';     %  100 T values: 0 < t <= max(t)
Af1 = feval(fun1,tf);           %  A matrix evaluated at tf values
kf1 = 1./ (Af1*c1);             %  Af*c is column vector of 1/kf values
Af2 = feval(fun2,tf);
kf2 = 1./ (Af2*c2);
plot(t,k,'o',tf,kf1,'--',tf,kf2,'-');
xlabel('Temperature  (K)');   ylabel('Conductivity  (W/m/C)');
legend('data','basis 1','basis 2');
```