We can fit the stress strain data with a quadratic polynomial instead of a linear one. Let us compare the two. Here is the data again

```
>> strain=[0:1:5]

strain =
       0    1    2    3    4    5
>> stress=[0:10:50]

stress =
       0   10   20   30   40   50
>> randn('state',0)
>> error_m=randn(1,6)

error_m =
   -0.4326 -1.6656  0.1253  0.2877 -1.1465  1.1909
>> stress_m=stress+error_m

stress_m =
   -0.4326  8.3344  20.1253  30.2877  38.8535  51.1909
```

Now the two fits.

```
>> [p s]=polyfit(strain,stress_m,1)

p =
   10.2811  -0.9761

s =
   R: [2x2 double]
   df: 4
```
normr: 1.9903

>> [p2 s2]=polyfit(strain,stress_m,2)

p2 =
\[
0.0884 \quad 9.8389 \quad -0.6813
\]

s2 =
\[
\text{R: [3x3 double]}
\]
\[
df: 3
\]

normr: 1.9156

We see that the norm of the residuals hardly changed by going to a quadratic function, which tells us that we did not gain much. We compare standard errors

\[
\text{std_err1} =
0.9951
\]

>> std_err2=1.9156/sqrt(3)

std_err2 =
\[
1.1060
\]

The fact that the standard error increased warns us of the danger of using a quadratic. This manifests itself mostly in extrapolation. So let us plot the two fits over a larger range of the strain.

>> strain_w=[0:1:20];

>> stress_fit1=polyval(p,strain_w);

>> stress_fit2=polyval(p2,strain_w);

>> plot(strain_w,stress_fit1,strain_w,stress_fit2)
Remember that the true Young modulus is 10. With the linear fit, for a strain of 20 we get a stress value very close to 200, but with the quadratic fit is around 230

```
>> stress_fit1(21), stress_fit2(21)
ans =
204.6451
ans =
231.4672
```
We first take the log in order to create a linear model

\[ \log Q = \log \alpha_0 + \alpha_1 \log D + \alpha_2 \log S \]

We enter the data

\[
\text{D} = [.3 .6 .9 .3 .6 .9 .3 .6 .9]';
\]
\[
\text{S} = [.001 .001 .001 .01 .01 .05 .05 .05]';
\]
\[
\text{Q} = [.04 .24 .69 .13 .82 2.38 .31 1.95 5.66]';
\]

Add a column of ones

\[
\text{o} = [1 1 1 1 1 1 1 1 1]';
\]

Create the Z matrix

\[
\text{Z} = [\text{o} \log10(\text{D}) \log10(\text{S})]
\]
Finally solve for the coefficients

\[
\begin{align*}
    a &= (Z'Z)^{-1}Z' \log_{10}(Q) \\
    a &= 1.5609 \quad 2.6279 \quad 0.5320
\end{align*}
\]

Thus the solution is

\[
\log(Q) = 1.5609 + 2.6279 \log(D) + 0.5320 \log(S)
\]

Or

\[
Q = 10^{1.5609} D^{2.6279} S^{0.5320}
\]

We could have obtained the results also with Matlab regress. Here is an extract from help regress
REGRESS Multiple linear regression using least squares.

\[ B = \text{REGRESS}(Y,X) \]

returns the vector \( B \) of regression coefficients in the

linear model \( Y = X \cdot B \). \( X \) is an \( n \times p \) design matrix, with rows
corresponding to observations and columns to predictor variables. \( Y \) is
an \( n \times 1 \) vector of response observations.


\[ [B,BINT] = \text{REGRESS}(Y,X) \]

returns a matrix \( BINT \) of 95\% confidence
intervals for \( B \).

\[ [B,BINT,R] = \text{REGRESS}(Y,X) \]

returns a vector \( R \) of residuals.

\[ [B,BINT,R,RINT] = \text{REGRESS}(Y,X) \]

returns a matrix \( RINT \) of intervals that
can be used to diagnose outliers. If \( RINT(i,:) \) does not contain zero,
then the \( i \)-th residual is larger than would be expected, at the 5\%
significance level. This is evidence that the \( i \)-th observation is an
outlier.

\[ [B,BINT,R,RINT,STATS] = \text{REGRESS}(Y,X) \]

returns a vector \( STATS \) containing
the R-square statistic, the F statistic and \( p \) value for the full model,
and an estimate of the error variance.

\[
>> [B,BINT,R,RINT,STATS]=\text{regress}(\log10(Q),Z)
\]
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1.5609</td>
<td>2.6279</td>
</tr>
<tr>
<td></td>
<td>0.5320</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BINT</td>
<td>1.5414</td>
<td>1.5804</td>
</tr>
<tr>
<td></td>
<td>2.5992</td>
<td>2.6567</td>
</tr>
<tr>
<td></td>
<td>0.5239</td>
<td>0.5401</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.0112</td>
<td>-0.0017</td>
<td>-0.0058</td>
<td>-0.0089</td>
<td>-0.0001</td>
<td>-0.0001</td>
<td>-0.0033</td>
</tr>
<tr>
<td></td>
<td>0.0043</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RINT</td>
<td>0.0061</td>
<td>0.0163</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.0172</td>
<td>0.0138</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.0183</td>
<td>0.0067</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The statistics with R-square of 0.99992 and a standard error of 4.8e-5 look wonderful, but we have to remember that these are errors related to the log. It is a good idea to calculate the errors in Q

\[
Q_{\text{cal}} = 10^{B(1)} D^{B(2)} S^{B(3)}
\]

Qcal =

3.8978e-002
2.4094e-001
6.9931e-001
1.3268e-001
8.2016e-001
2.3804e+000
3.1236e-001
1.9308e+000
5.6040e+000

>> Qcal-Q

ans =
-1.0216e-003
9.4161e-004
9.3081e-003
2.6818e-003
1.6122e-004
4.3297e-004
2.3594e-003
-1.9177e-002
-5.5986e-002

We see that the fitting errors are of the order of 1%.

Nonlinear models can some time be linearized by a transformation, but can also be solved directly by minimizing the sum of the squares of the error using optimization, as in fminsearch. As an example, let us do Problem 14.13.
We first key in the data

```matlab
>> S = [.01 .05 .1 .5 1 5 10 50 100];
>> v0 = [6.078e-11 7.595e-9 6.063e-8 5.788e-6 1.737e-5 2.423e-5 2.431e-5 2.431e-5];
```

We can transform it to a linear regression problem as

\[
\frac{1}{v_0} = K \frac{1}{k_m S^3} + \frac{1}{k_m}
\]

We can now do a linear regression

```matlab
>> ones3=1./S.^3;
>> onev0=1./v0;
>> [p stat]=polyfit(ones3,onev0,1)
```

\[
p = 1.6453e+004 4.1400e+004
\]

\[
\text{stat} = R: [2x2 \text{ double}]
\]
df: 7

normr: 2.4371e+003

>> km=1/p(2)

km =

2.4155e-005

>> K=km*p(1)

K =

3.9741e-001

So that the fit is

\[ v_0 = \frac{2.4155 \times 10^{-5} S^3}{0.39741 + S^3} \]

This fit, however, does not minimize the sum of the squares of the error in the original data. To do that, we use optimization.

We can create a function that calculates the sum of the squares in an M-file

function f = fSSR(a,Sm,v0m)

v0p = a(1)*Sm.^3./(a(2)+Sm.^3);

f = sum((v0m-v0p).^2);

Or we can do that with an anonymous function.

Normerr=@(k) norm(k(1)*S.^3./(k(2)+S.^3)-v0)

>> kmin=fminsearch(normerr,[2e-5 1])

kmin =

2.4310e-005 3.9976e-001

So that we get slightly different coefficients.
>> normerr(kmin)
ans =
  4.6197e-009

>> ktran=[km K]
ktran =
   2.4155e-005   3.9741e-001

>> normerr(ktran)
ans =
  3.2015e-007

So there is substantial difference.