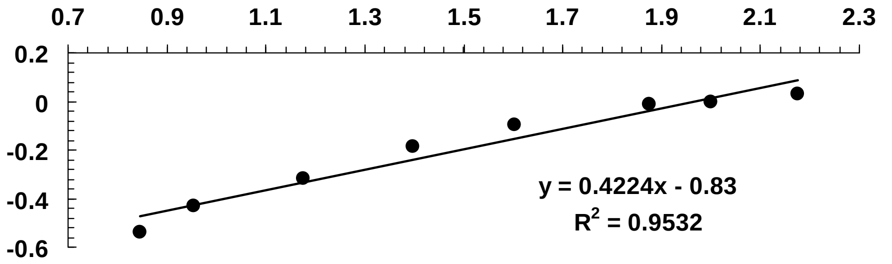
# CHAPTER 20

**20.1** A plot of $\log_{10}k$ versus $\log_{10}f$ can be developed as



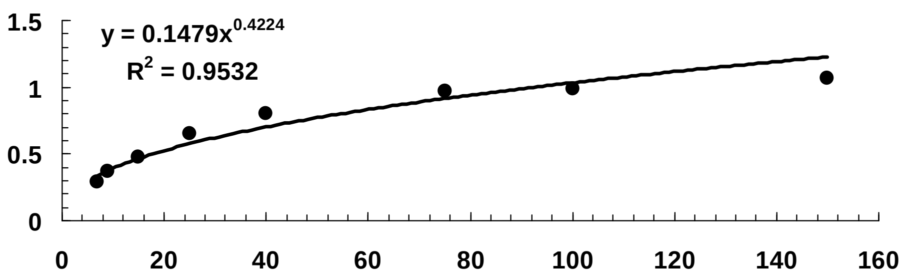

As shown, the best fit line is

$$\log_{10} f = 0.422363 \log_{10} k - 0.83$$

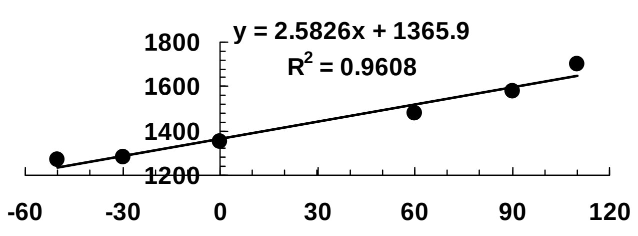
Therefore, $\alpha_2 = 10^{-0.83} = 0.147913$ and $\beta_2 = 0.422363$, and the power model is

$$y = 0.147913x^{0.422363}$$

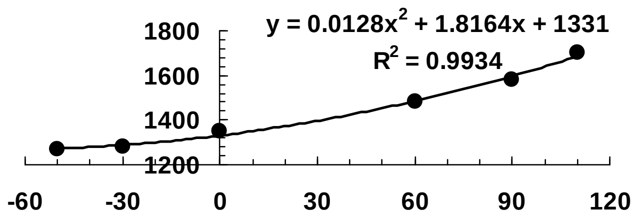
The model and the data can be plotted as




**20.2** We can first try a linear fit

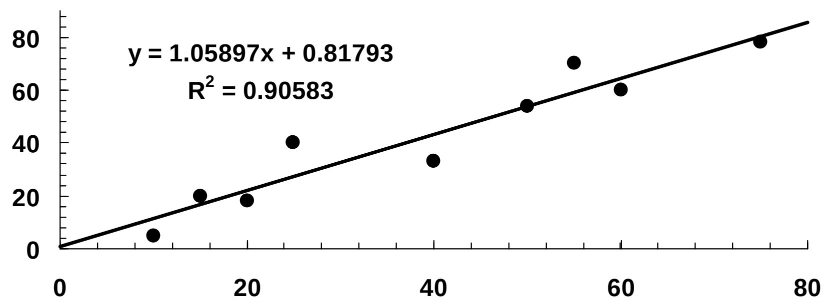



As shown, the fit line is somewhat lacking. Therefore, we can use polynomial regression to fit a parabola




This fit seems adequate in that it captures the general trend of the data. Note that a slightly better fit can be attained with a cubic polynomial, but the improvement is marginal.

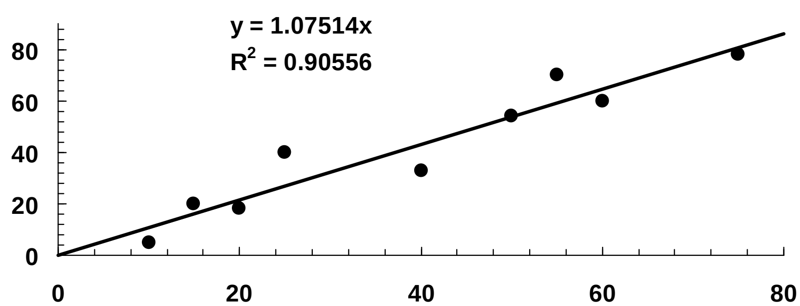
**20.3 (a)** The linear fit is




The tensile strength at $t$ = 32 can be computed as

$$y = 1.05897(32) + 0.81793 = 34.7048913$$

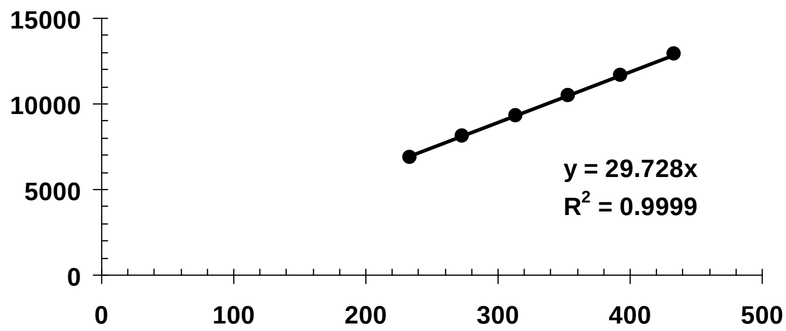
**(b)** A straight line with zero intercept can be fit as




For this case, the tensile strength at $t$ = 32 can be computed as

$$y = 1.07514(32) = 34.40452$$

**20.4** Linear regression with a zero intercept gives [note that $T$(K) = $T$($^o$C) + 273.15].




Thus, the fit is

$$p = 29.728T$$

Using the ideal gas law

$$R = \left(\frac{p}{T}\right)\frac{V}{n}$$

For our fit

$$\frac{p}{T} = 29.728$$

For nitrogen,

$$n = \frac{1 \text{ kg}}{28 \text{ g/mole}}$$

Therefore,

$$R = 29.728\left(\frac{10}{10^3 / 28}\right) = 8.324$$

This is close to the standard value of 8.314 J/gmole.

**20.5** This problem is ideally suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$x_0 = 740$ $f(x_0) = 0.1406$
$x_1 = 760$ $f(x_1) = 0.15509$
$x_2 = 720$ $f(x_2) = 0.12184$
$x_3 = 780$ $f(x_3) = 0.16643$
$x_4 = 700$ $f(x_4) = 0.0977$

The results of applying Newton's polynomial at $T = 750$ are

| Order | *f*(*x*) | Error |
|---|---|---|
| 0 | 0.14060 | 0.007245 |
| 1 | 0.14785 | 0.000534 |

| | | |
|---|---|---|
| 2 | 0.14838 | -7E-05 |
| 3 | 0.14831 | 0.00000 |
| 4 | 0.14831 | |

Note that the third-order polynomial yields an exact result, and so we conclude that the interpolation is 0.14831.

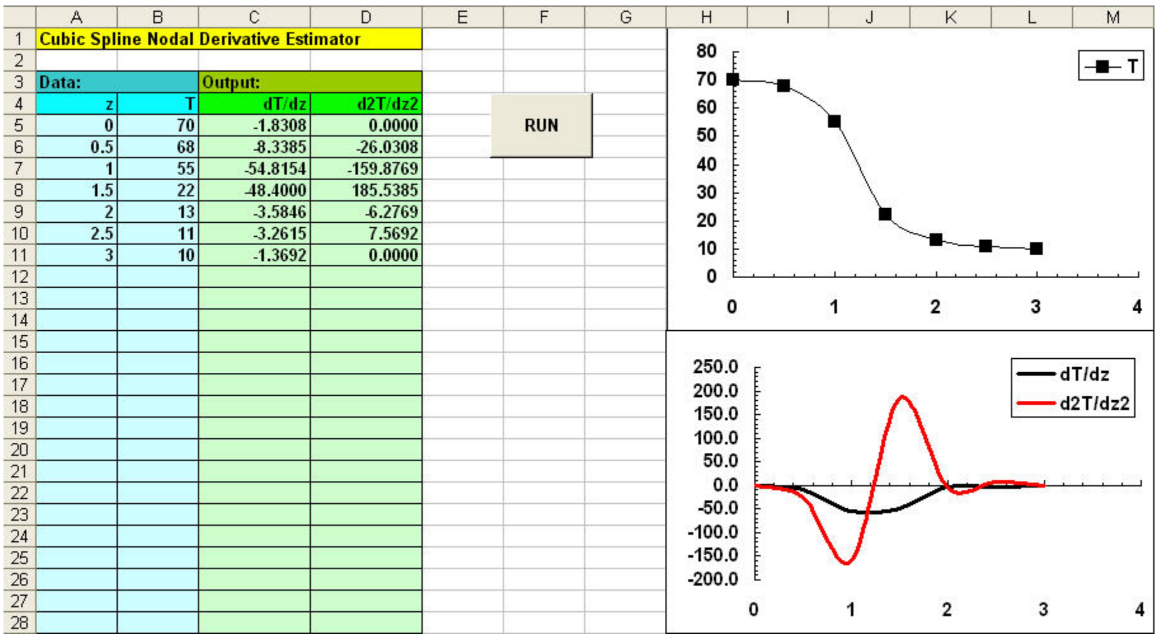
**20.6** A program can be written to fit a natural cubic spline to this data and also generate the first and second derivatives at each knot.

```
Option Explicit

Sub Splines()
Dim i As Integer, n As Integer
Dim x(100) As Double, y(100) As Double, xu As Double, yu As Double
Dim dy As Double, d2y As Double
Dim resp As Variant
Range("a5").Select
n = ActiveCell.Row
Selection.End(xlDown).Select
n = ActiveCell.Row - n
Range("a5").Select
For i = 0 To n
  x(i) = ActiveCell.Value
  ActiveCell.Offset(0, 1).Select
  y(i) = ActiveCell.Value
  ActiveCell.Offset(1, -1).Select
Next i
Range("c5").Select
Range("c5:d1005").ClearContents
For i = 0 To n
  Call Spline(x(), y(), n, x(i), yu, dy, d2y)
  ActiveCell.Value = dy
  ActiveCell.Offset(0, 1).Select
  ActiveCell.Value = d2y
  ActiveCell.Offset(1, -1).Select
Next i
Do
  resp = MsgBox("Do you want to interpolate?", vbYesNo)
  If resp = vbNo Then Exit Do
  xu = InputBox("z = ")
  Call Spline(x(), y(), n, xu, yu, dy, d2y)
  MsgBox "For z = " & xu & Chr(13) & "T = " & yu & Chr(13) & _
                            "dT/dz = " & dy & Chr(13) & "d2T/dz2 = " & d2y
Loop
End Sub

Sub Spline(x, y, n, xu, yu, dy, d2y)
Dim e(100) As Double, f(100) As Double, g(100) As Double, r(100) As Double,
d2x(100) As Double
Call Tridiag(x, y, n, e, f, g, r)
Call Decomp(e(), f(), g(), n - 1)
Call Substit(e(), f(), g(), r(), n - 1, d2x())
Call Interpol(x, y, n, d2x(), xu, yu, dy, d2y)
End Sub

Sub Tridiag(x, y, n, e, f, g, r)
Dim i As Integer
f(1) = 2 * (x(2) - x(0))
g(1) = x(2) - x(1)
```

```
r(1) = 6 / (x(2) - x(1)) * (y(2) - y(1))
r(1) = r(1) + 6 / (x(1) - x(0)) * (y(0) - y(1))
For i = 2 To n - 2
  e(i) = x(i) - x(i - 1)
  f(i) = 2 * (x(i + 1) - x(i - 1))
  g(i) = x(i + 1) - x(i)
  r(i) = 6 / (x(i + 1) - x(i)) * (y(i + 1) - y(i))
  r(i) = r(i) + 6 / (x(i) - x(i - 1)) * (y(i - 1) - y(i))
Next i
e(n - 1) = x(n - 1) - x(n - 2)
f(n - 1) = 2 * (x(n) - x(n - 2))
r(n - 1) = 6 / (x(n) - x(n - 1)) * (y(n) - y(n - 1))
r(n - 1) = r(n - 1) + 6 / (x(n - 1) - x(n - 2)) * (y(n - 2) - y(n - 1))
End Sub

Sub Interpol(x, y, n, d2x, xu, yu, dy, d2y)
Dim i As Integer, flag As Integer
Dim c1 As Double, c2 As Double, c3 As Double, c4 As Double
Dim t1 As Double, t2 As Double, t3 As Double, t4 As Double
flag = 0
i = 1
Do
  If xu >= x(i - 1) And xu <= x(i) Then
    c1 = d2x(i - 1) / 6 / (x(i) - x(i - 1))
    c2 = d2x(i) / 6 / (x(i) - x(i - 1))
    c3 = y(i - 1) / (x(i) - x(i - 1)) - d2x(i - 1) * (x(i) - x(i - 1)) / 6
    c4 = y(i) / (x(i) - x(i - 1)) - d2x(i) * (x(i) - x(i - 1)) / 6
    t1 = c1 * (x(i) - xu) ^ 3
    t2 = c2 * (xu - x(i - 1)) ^ 3
    t3 = c3 * (x(i) - xu)
    t4 = c4 * (xu - x(i - 1))
    yu = t1 + t2 + t3 + t4
    t1 = -3 * c1 * (x(i) - xu) ^ 2
    t2 = 3 * c2 * (xu - x(i - 1)) ^ 2
    t3 = -c3
    t4 = c4
    dy = t1 + t2 + t3 + t4
    t1 = 6 * c1 * (x(i) - xu)
    t2 = 6 * c2 * (xu - x(i - 1))
    d2y = t1 + t2
    flag = 1
  Else
    i = i + 1
  End If
  If i = n + 1 Or flag = 1 Then Exit Do
Loop
If flag = 0 Then
  MsgBox "outside range"
  End
End If
End Sub

Sub Decomp(e, f, g, n)
Dim k As Integer
For k = 2 To n
  e(k) = e(k) / f(k - 1)
  f(k) = f(k) - e(k) * g(k - 1)
Next k
End Sub

Sub Substit(e, f, g, r, n, x)
Dim k As Integer
```

```
For k = 2 To n
  r(k) = r(k) - e(k) * r(k - 1)
Next k
x(n) = r(n) / f(n)
For k = n - 1 To 1 Step -1
  x(k) = (r(k) - g(k) * x(k + 1)) / f(k)
Next k
End Sub
```

Here is the output when it is applied to the data for this problem:


Cubic Spline Nodal Derivative Estimator

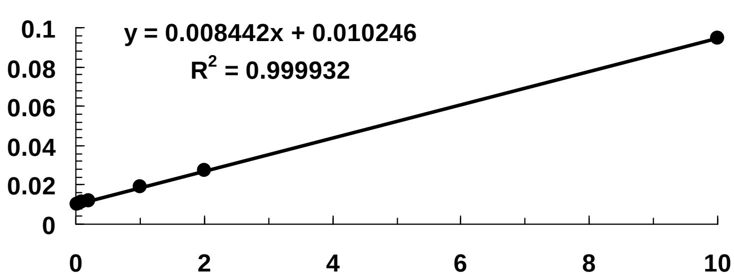
| Data: | | Output: | |
|---|---|---|---|
| z | T | dT/dz | d2T/dz2 |
| 0 | 70 | -1.8308 | 0.0000 |
| 0.5 | 68 | -8.3385 | -26.0308 |
| 1 | 55 | -54.8154 | -159.8769 |
| 1.5 | 22 | -48.4000 | 185.5385 |
| 2 | 13 | -3.5846 | -6.2769 |
| 2.5 | 11 | -3.2615 | 7.5692 |
| 3 | 10 | -1.3692 | 0.0000 |



The plot suggests a zero second derivative at a little above $z$ = 1.2 m. The program is set up to allow you to evaluate the spline prediction and the associated derivatives for as many cases as you desire. This can be done by trial-and-error to determine that the zero second derivative occurs at about $z$ = 1.2315 m. At this point, the first derivative is –73.315 °C/m. This can be substituted into Fourier's law to compute the flux as

$$J = -0.02 \frac{\text{cal}}{\text{s cm °C}} \left( -73.315 \frac{\text{°C}}{\text{m}} \right) \times \frac{\text{m}}{100 \text{ cm}} = 0.01466 \frac{\text{cal}}{\text{cm}^2 \text{ s}}$$

**20.7** This is an example of the saturated-growth-rate model. We can plot 1/[F] versus 1/[B] and fit a straight line as shown below.



The model coefficients can then be computed as

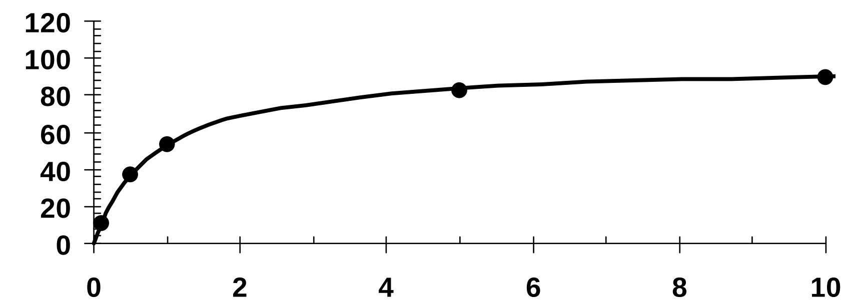
$$\alpha_3 = \frac{1}{0.010246} = 97.60128$$

$$\beta_3 = 97.60128(0.008442) = 0.823968$$

Therefore, the best-fit model is

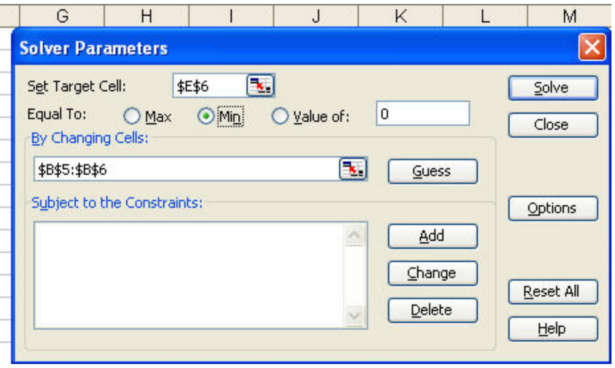
$$[B] = 97.60128 \frac{[F]}{0.823968 + [F]}$$

A plot of the original data along with the best-fit curve can be developed as




**20.8** Nonlinear regression can be used to estimate the model parameters. This can be done using the Excel Solver

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Prob 20.12 | | | | |
| 2 | | | | | |
| 3 | R | 0.00198 | | | |
| 4 | | | | | |
| 5 | k01 | 7526.842 | | | |
| 6 | E1 | 4.478896 | | SSR | 1547.576 |
| 7 | | | | | |
| 8 | -dA/dt data | A | T | -dA/dt calc | SR |
| 9 | 460 | 200 | 280 | 466.7210343 | 45.1723 |
| 10 | 960 | 150 | 320 | 960.9301461 | 0.865172 |
| 11 | 2485 | 50 | 450 | 2468.669749 | 266.6771 |
| 12 | 1600 | 20 | 500 | 1632.431512 | 1051.803 |
| 13 | 1245 | 10 | 550 | 1231.470094 | 183.0584 |
| 14 | | | | | |
| 15 | | | | | |



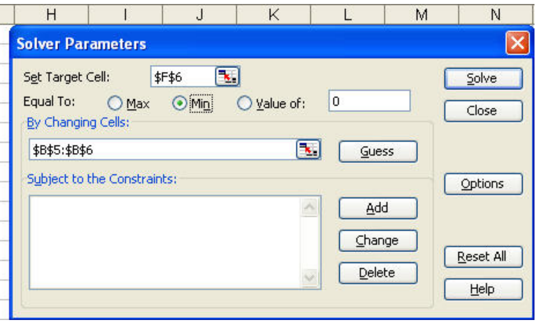

Here are the formulas:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Prob 20.12 | | | | |
| 2 | | | | | |
| 3 | R | 0.00198 | | | |
| 4 | | | | | |
| 5 | k01 | 7526.84240252 | | | |
| 6 | E1 | 4.47889589058 | | SSR | =SUM(E9:E16) |
| 7 | | | | | |
| 8 | -dA/dt data | A | T | -dA/dt calc | SR |
| 9 | 460 | 200 | 280 | =k01_*EXP(-E1_/(R_*C9))*B9 | =(A9-D9)^2 |
| 10 | 960 | 150 | 320 | =k01_*EXP(-E1_/(R_*C10))*B10 | =(A10-D10)^2 |
| 11 | 2485 | 50 | 450 | =k01_*EXP(-E1_/(R_*C11))*B11 | =(A11-D11)^2 |
| 12 | 1600 | 20 | 500 | =k01_*EXP(-E1_/(R_*C12))*B12 | =(A12-D12)^2 |
| 13 | 1245 | 10 | 550 | =k01_*EXP(-E1_/(R_*C13))*B13 | =(A13-D13)^2 |

Therefore, we estimate $k_{01}$ = 7,526.842 and $E_1$ = 4.478896.

**20.9** Nonlinear regression can be used to estimate the model parameters. This can be done using the Excel Solver. To do this, we set up columns holding each side of the equation: $PV/(RT)$ and $1 + A_1/V + A_2/V$. We then form the square of the difference between the two sides and minimize this difference by adjusting the parameters.

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Prob. 20.13 | | | | | | |
| 2 | | | | | | | |
| 3 | R | 82.05 | | | | | |
| 4 | | | | | | | |
| 5 | A1 | -237.531 | | | | | |
| 6 | A2 | 1.192283 | | | SSR | 2.09474E-08 | |
| 7 | | | | | | | |
| 8 | P (atm) | T(K) | Vol(mL) | PV/(RT) | 1 + A1/V +A2/V^2 | SR | |
| 9 | 0.985 | 303 | 25000 | 0.990501 | 0.990498757 | 6.18312E-12 | |
| 10 | 1.108 | 303 | 22200 | 0.989399 | 0.989300403 | 9.74565E-09 | |
| 11 | 1.363 | 303 | 18000 | 0.986841 | 0.986803831 | 1.37521E-09 | |
| 12 | 1.631 | 303 | 15000 | 0.984065 | 0.984164598 | 9.82039E-09 | |
| 13 | | | | | | | |
| 14 | | | | | | | |
| 15 | | | | | | | |




Here are the formulas that underly the worksheet cells:

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Prob. 20.13 | | | | | |
| 2 | | | | | | |
| 3 | R | 82.05 | | | | |
| 4 | | | | | | |
| 5 | A1 | -237.53111469 | | | | |
| 6 | A2 | 1.19228253686 | | | SSR | =SUM(F9:F12) |
| 7 | | | | | | |
| 8 | P (atm) | T(K) | Vol(mL) | PV/(RT) | 1 + A1/V +A2/V^2 | SR |
| 9 | 0.985 | 303 | 25000 | =A9*C9/(R_*B9) | =1+A1_/C9+A2_/C9^2 | =(D9-E9)^2 |
| 10 | 1.108 | 303 | 22200 | =A10*C10/(R_*B10) | =1+A1_/C10+A2_/C10^2 | =(D10-E10)^2 |
| 11 | 1.363 | 303 | 18000 | =A11*C11/(R_*B11) | =1+A1_/C11+A2_/C11^2 | =(D11-E11)^2 |
| 12 | 1.631 | 303 | 15000 | =A12*C12/(R_*B12) | =1+A1_/C12+A2_/C12^2 | =(D12-E12)^2 |

Therefore, we estimate $A_1$ = -237.531 and $A_2$ = 1.192283.

**20.10** The standard errors can be computed via Eq. 17.9

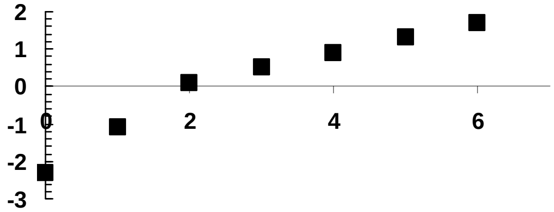
$$s_{y/x} = \sqrt{\frac{S_r}{n-p}}$$

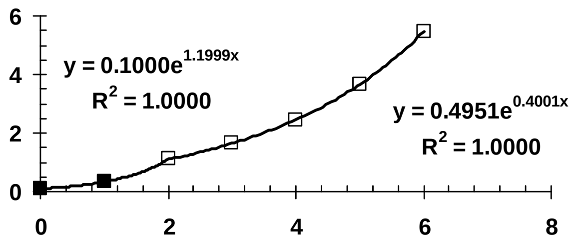
$n$ = 15

| | Model A | Model B | Model C |
|---|---|---|---|
| $S_r$ | 135 | 105 | 100 |
| Number of model parameters fit ($p$) | 2 | 3 | 5 |
| $s_{y/x}$ | 3.222517 | 2.95804 | 3.162278 |

Thus, Model B seems best because its standard error is lower.

**20.11** A plot of the natural log of cells versus time indicates two straight lines with a sharp break at 2. The Excel Trendline tool can be used to fit each range separately with the exponential model as shown in the second plot.

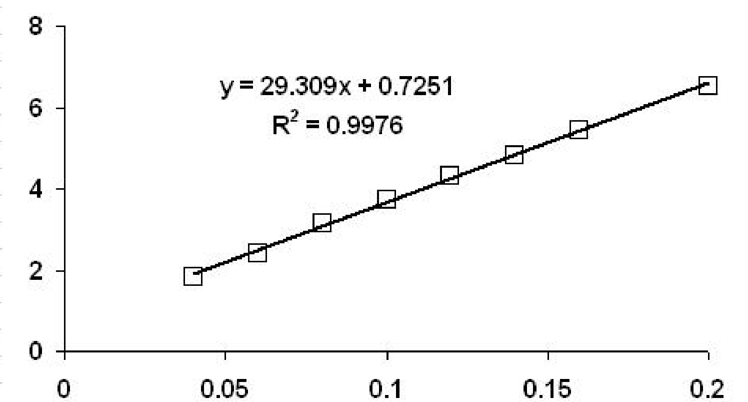



**20.12** This problem can be solved with Excel,

B17 =(INTERCEPT(C8:C15,A8:A15)/K)^(1/a)

| | A | B | C |
|---|---|---|---|
| 1 | Problem 20.16 | | |
| 2 | | | |
| 3 | K | 3.90E-04 | |
| 4 | a | 0.58 | |
| 5 | | | |
| 6 | c | t | (t/t0-1)/c |
| 7 | 0 | 83 | |
| 8 | 0.04 | 89 | 1.807229 |
| 9 | 0.06 | 95 | 2.409639 |
| 10 | 0.08 | 104 | 3.162651 |
| 11 | 0.1 | 114 | 3.73494 |
| 12 | 0.12 | 126 | 4.317269 |
| 13 | 0.14 | 139 | 4.819277 |
| 14 | 0.16 | 155 | 5.421687 |
| 15 | 0.2 | 191 | 6.506024 |
| 16 | | | |
| 17 | M | 433273.1 | |




**20.13** This problem can be solved with a power model,

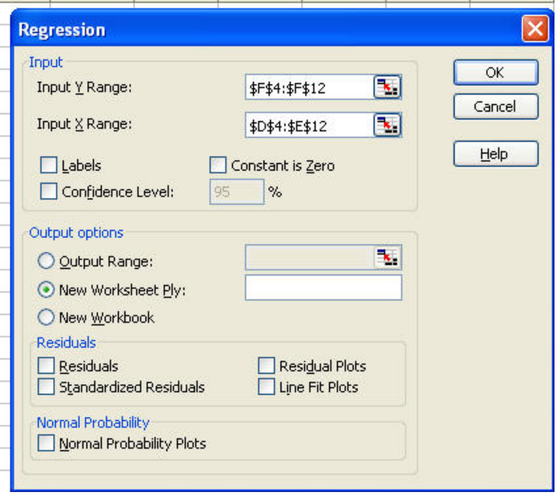
$$A = a_0 H^{a_1} W^{a_2}$$

This equation can be linearized by taking the logarithm

$$\log_{10} A = \log_{10} a_0 + a_1 \log_{10} H + a_2 \log_{10} W$$

This model can be fit with the Excel Data Analysis Regression tool,

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Problem 20.17 | | | | | |
| 2 | | | | | | |
| 3 | Height (cm) | Weight (kg) | Area (m2) | log(H) | log(W) | log(A) |
| 4 | 182 | 74 | 1.92 | 2.260071 | 1.869232 | 0.283301 |
| 5 | 180 | 88 | 2.11 | 2.255273 | 1.944483 | 0.324282 |
| 6 | 179 | 94 | 2.15 | 2.252853 | 1.973128 | 0.332438 |
| 7 | 187 | 78 | 2.02 | 2.271842 | 1.892095 | 0.305351 |
| 8 | 189 | 84 | 2.09 | 2.276462 | 1.924279 | 0.320146 |
| 9 | 194 | 98 | 2.31 | 2.287802 | 1.991226 | 0.363612 |
| 10 | 195 | 76 | 2.02 | 2.290035 | 1.880814 | 0.305351 |
| 11 | 193 | 86 | 2.16 | 2.285557 | 1.934498 | 0.334454 |
| 12 | 200 | 96 | 2.31 | 2.30103 | 1.982271 | 0.363612 |



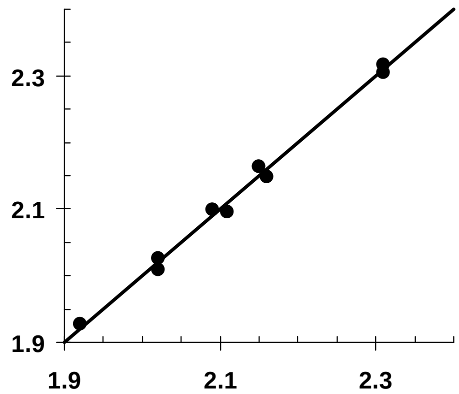

The results are

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SUMMARY OUTPUT | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | *Regression Statistics* | | | | | | | | |
| 4 | Multiple R | 0.996755676 | | | | | | | |
| 5 | R Square | 0.993521878 | | | | | | | |
| 6 | Adjusted R Square | 0.991362504 | | | | | | | |
| 7 | Standard Error | 0.002470988 | | | | | | | |
| 8 | Observations | 9 | | | | | | | |
| 9 | | | | | | | | | |
| 10 | ANOVA | | | | | | | | |
| 11 | | *df* | *SS* | *MS* | *F* | *Significance F* | | | |
| 12 | Regression | 2 | 0.005618506 | 0.002809 | 460.0972 | 2.71861E-07 | | | |
| 13 | Residual | 6 | 3.66347E-05 | 6.11E-06 | | | | | |
| 14 | Total | 8 | 0.005655141 | | | | | | |
| 15 | | | | | | | | | |
| 16 | | *Coefficients* | *Standard Error* | *t Stat* | *P-value* | *Lower 95%* | *Upper 95%* | *ower 95.0%* | *Ipper 95.0%* |
| 17 | Intercept | -1.864582132 | 0.118395673 | -15.7487 | 4.16E-06 | -2.154285906 | -1.57488 | -2.15429 | -1.57488 |
| 18 | X Variable 1 | 0.521803441 | 0.052706914 | 9.900095 | 6.13E-05 | 0.392834268 | 0.650773 | 0.392834 | 0.650773 |
| 19 | X Variable 2 | 0.519017838 | 0.019877438 | 26.1109 | 2.08E-07 | 0.4703795 | 0.567656 | 0.470379 | 0.567656 |

Therefore the best-fit coefficients are $a_0 = 10^{-1.86458} = 0.013659$, $a_1 = 0.5218$, and $a_2 = 0.5190$. Therefore, the model is

$$A = 0.013659 H^{0.5218} W^{0.5190}$$

We can assess the fit by plotting the model predictions versus the data. We have included the perfect 1:1 line for comparison.

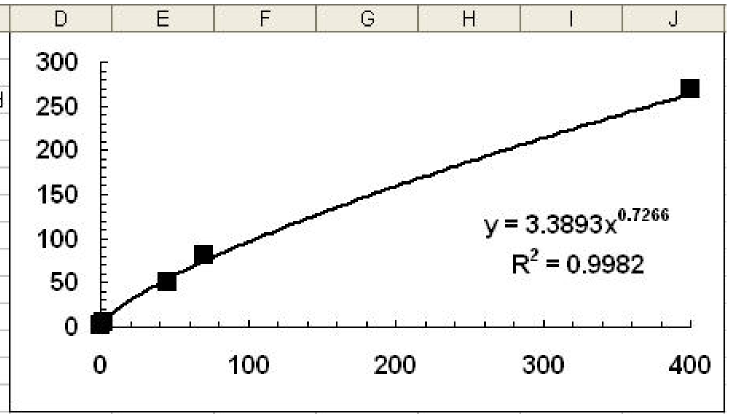



The prediction at $H$ = 187 cm and $W$ = 78 kg can be computed as

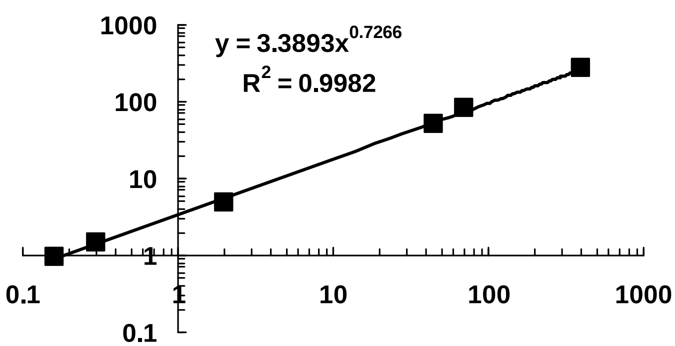
$$A = 0.013659(187)^{0.5218}(78)^{0.5190} = 2.008646$$

**20.14** The Excel Trend Line tool can be used to fit a power law to the data:

| | A | B | C |
|---|---|---|---|
| 1 | Problem 20.18 | | |
| 2 | | | |
| 3 | | Mass, kg | Metabolism, kcal/d |
| 4 | Cow | 400 | 270 |
| 5 | Human | 70 | 82 |
| 6 | Sheep | 45 | 50 |
| 7 | Hen | 2 | 4.8 |
| 8 | Rat | 0.3 | 1.45 |
| 9 | Dove | 0.16 | 0.97 |
| 10 | | | |
| 11 | | | |
| 12 | | | |
| 13 | | | |
| 14 | | | |




Note that although the model seems to represent a good fit, the performance of the lower-mass animals is not evident because of the scale. A log-log plot provides a better perspective to assess the fit:




**20.15** For the Casson Region, linear regression yields

$$\sqrt{\tau} = 0.065818 + 0.180922\sqrt{\dot{\gamma}} \qquad r^2 = 0.99985$$

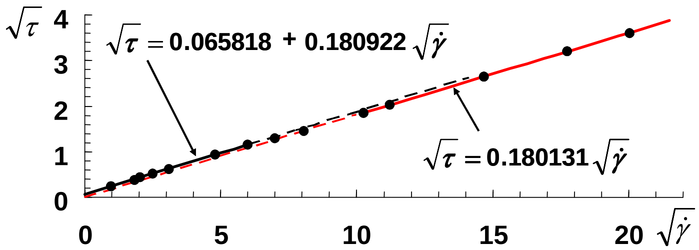
For the Newton Region, linear regression with zero intercept gives

$$\tau = 0.032447\dot{\gamma} \qquad r^2 = 0.99992$$

On a Casson plot, this function becomes

$$\sqrt{\tau} = 0.180131\sqrt{\dot{\gamma}}$$

We can plot both functions along with the data on a Casson plot




**20.16** Recall from Sec. 4.1.3, that finite divided differences can be used to estimate the derivatives. For the first point, we can use a forward difference (Eq. 4.17)

$$\frac{d\sigma}{d\varepsilon} = \frac{96.6 - 87.8}{204 - 153} = 0.1725$$

For the intermediate points, we can use centered differences. For example, for the second point

$$\frac{d\sigma}{d\varepsilon} = \frac{176 - 87.8}{255 - 153} = 0.8647$$

For the last point, we can use a backward difference

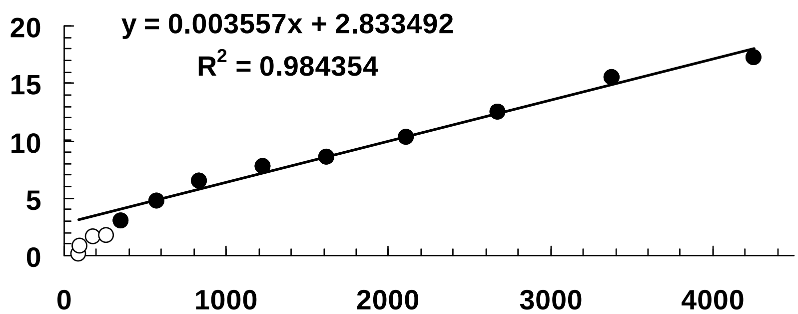
$$\frac{d\sigma}{d\varepsilon} = \frac{4258 - 3380}{765 - 714} = 17.2157$$

All the values can be tabulated as

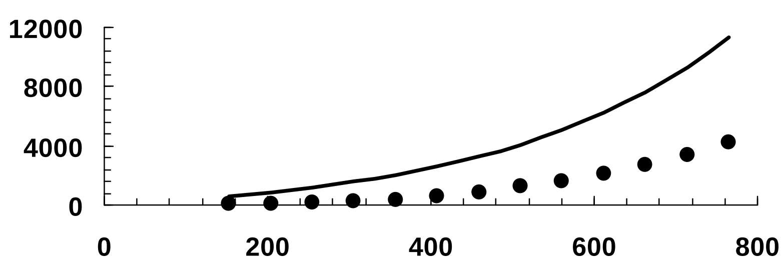
| $\sigma$ | $\varepsilon$ | $d\sigma/d\varepsilon$ |
|---|---|---|
| 87.8 | 153 | 0.1725 |
| 96.6 | 204 | 0.8647 |
| 176 | 255 | 1.6314 |
| 263 | 306 | 1.7157 |
| 351 | 357 | 3.0196 |
| 571 | 408 | 4.7353 |
| 834 | 459 | 6.4510 |
| 1229 | 510 | 7.7451 |
| 1624 | 561 | 8.6078 |
| 2107 | 612 | 10.3333 |
| 2678 | 663 | 12.4804 |
| 3380 | 714 | 15.4902 |

| | | |
|---|---|---|
| 4258 | 765 | 17.2157 |

We can plot these results and after discarding the first few points, we can fit a straight line as shown (note that the discarded points are displayed as open circles).




Therefore, the parameter estimates are $E_o$ = 2.833492 and $a$ = 0.003557. The data along with the first equation can be plotted as



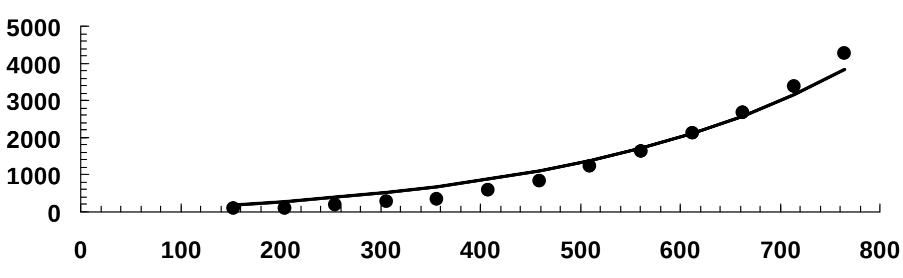

As described in the problem statement, the fit is not very good. We therefore pick a point near the midpoint of the data interval

$(\bar{\varepsilon} = 612, \bar{\sigma} = 2107)$

We can then plot the second model

$$\sigma = \left(\frac{2107}{e^{0.003557(612)} - 1}\right)(e^{0.003557\varepsilon} - 1) = 269.5(e^{0.003557\varepsilon} - 1)$$

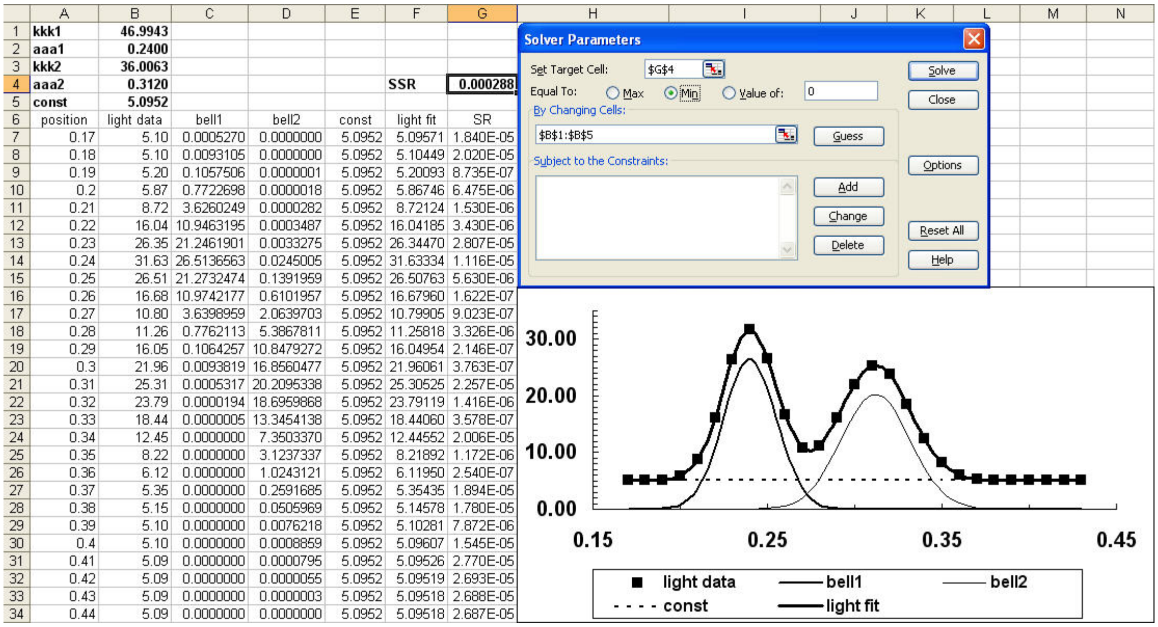
The result is a much better fit



**20.17** The problem is set up as the following Excel Solver application. Notice that we have assumed that the model consists of a constant plus two bell-shaped curves:

$$f(x) = c + \frac{k_1 e^{-k_1^2 (x-a_1)^2}}{\sqrt{\pi}} + \frac{k_2 e^{-k_2^2 (x-a_2)^2}}{\sqrt{\pi}}$$

The resulting solution is



| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | **kkk1** | **46.9943** | | | | | |
| 2 | **aaa1** | **0.2400** | | | | | |
| 3 | **kkk2** | **36.0063** | | | | | |
| 4 | **aaa2** | **0.3120** | | | | **SSR** | **0.000288** |
| 5 | **const** | **5.0952** | | | | | |
| 6 | position | light data | bell1 | bell2 | const | light fit | SR |
| 7 | 0.17 | 5.10 | 0.0005270 | 0.0000000 | 5.0952 | 5.09571 | 1.840E-05 |
| 8 | 0.18 | 5.10 | 0.0093105 | 0.0000000 | 5.0952 | 5.10449 | 2.020E-05 |
| 9 | 0.19 | 5.20 | 0.1057506 | 0.0000001 | 5.0952 | 5.20093 | 8.735E-07 |
| 10 | 0.2 | 5.87 | 0.7722698 | 0.0000018 | 5.0952 | 5.86746 | 6.475E-06 |
| 11 | 0.21 | 8.72 | 3.6260249 | 0.0000282 | 5.0952 | 8.72124 | 1.530E-06 |
| 12 | 0.22 | 16.04 | 10.9463195 | 0.0003487 | 5.0952 | 16.04185 | 3.430E-06 |
| 13 | 0.23 | 26.35 | 21.2461901 | 0.0033275 | 5.0952 | 26.34470 | 2.807E-05 |
| 14 | 0.24 | 31.63 | 26.5136563 | 0.0245005 | 5.0952 | 31.63334 | 1.116E-05 |
| 15 | 0.25 | 26.51 | 21.2732474 | 0.1391959 | 5.0952 | 26.50763 | 5.630E-06 |
| 16 | 0.26 | 16.68 | 10.9742177 | 0.6101957 | 5.0952 | 16.67960 | 1.622E-07 |
| 17 | 0.27 | 10.80 | 3.6398959 | 2.0639703 | 5.0952 | 10.79905 | 9.023E-07 |
| 18 | 0.28 | 11.26 | 0.7762113 | 5.3867811 | 5.0952 | 11.25818 | 3.326E-06 |
| 19 | 0.29 | 16.05 | 0.1064257 | 10.8479272 | 5.0952 | 16.04954 | 2.146E-07 |
| 20 | 0.3 | 21.96 | 0.0093819 | 16.8560477 | 5.0952 | 21.96061 | 3.763E-07 |
| 21 | 0.31 | 25.31 | 0.0005317 | 20.2095338 | 5.0952 | 25.30525 | 2.257E-05 |
| 22 | 0.32 | 23.79 | 0.0000194 | 18.6959868 | 5.0952 | 23.79119 | 1.416E-06 |
| 23 | 0.33 | 18.44 | 0.0000005 | 13.3454138 | 5.0952 | 18.44060 | 3.578E-07 |
| 24 | 0.34 | 12.45 | 0.0000000 | 7.3503370 | 5.0952 | 12.44552 | 2.006E-05 |
| 25 | 0.35 | 8.22 | 0.0000000 | 3.1237337 | 5.0952 | 8.21892 | 1.172E-06 |
| 26 | 0.36 | 6.12 | 0.0000000 | 1.0243121 | 5.0952 | 6.11950 | 2.540E-07 |
| 27 | 0.37 | 5.35 | 0.0000000 | 0.2591685 | 5.0952 | 5.35435 | 1.894E-05 |
| 28 | 0.38 | 5.15 | 0.0000000 | 0.0505969 | 5.0952 | 5.14578 | 1.780E-05 |
| 29 | 0.39 | 5.10 | 0.0000000 | 0.0076218 | 5.0952 | 5.10281 | 7.872E-06 |
| 30 | 0.4 | 5.10 | 0.0000000 | 0.0008859 | 5.0952 | 5.09607 | 1.545E-05 |
| 31 | 0.41 | 5.09 | 0.0000000 | 0.0000795 | 5.0952 | 5.09526 | 2.770E-05 |
| 32 | 0.42 | 5.09 | 0.0000000 | 0.0000055 | 5.0952 | 5.09519 | 2.693E-05 |
| 33 | 0.43 | 5.09 | 0.0000000 | 0.0000003 | 5.0952 | 5.09518 | 2.688E-05 |
| 34 | 0.44 | 5.09 | 0.0000000 | 0.0000000 | 5.0952 | 5.09518 | 2.687E-05 |



Thus, the retina thickness is estimated as 0.312 – 0.24 = 0.072.

**20.18** The equation can be linearized by inverting it

$$\frac{1}{c} = \frac{1}{c_0} + kt$$

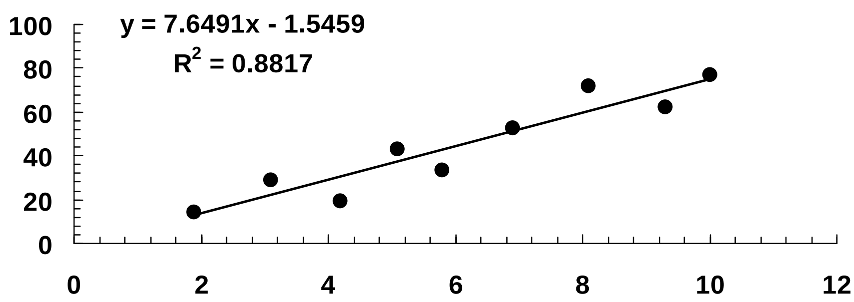
Thus, $1/c$ versus $t$ should yield a straight line with a slope of $k$ and an intercept of $1/c_0$.

| | x | | y | x*y | x^2 |
|---|---|---|---|---|---|
| | t, min | c, mole/L | 1/c | t*(1/c) | t^2 |
| | 0 | 0.381 | 2.624672 | 0 | 0 |
| | 20 | 0.264 | 3.787879 | 75.75758 | 400 |
| | 50 | 0.18 | 5.555556 | 277.7778 | 2500 |
| | 65 | 0.151 | 6.622517 | 430.4636 | 4225 |
| | 150 | 0.086 | 11.62791 | 1744.186 | 22500 |
| sum | 285 | | 30.21853 | 2528.185 | 29625 |
| n | 5 | | | | |
| slope | 0.060219 | | | | |
| int | 2.611229 | | | | |

Therefore, $k = 0.0602$ and $c_0 = 1/26112 = 0.383$.

$$c = \frac{0.383}{1+0.0602(0.383)160} = 0.08166$$

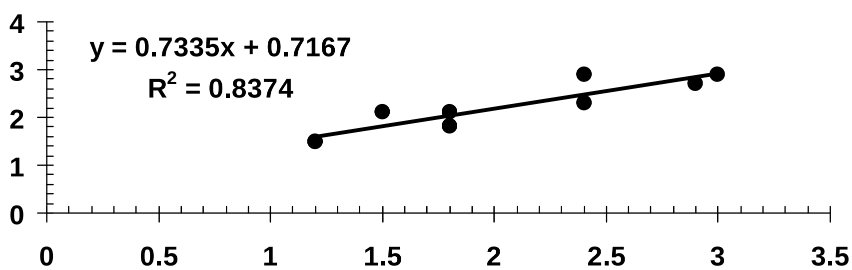
**20.19** Simple linear regression can be applied to yield the following fit




The shear stress at the depth of 4.5 m can be computed as

$$\sigma = 7.6491(4.5) - 1.5459 = 32.875$$

**20.20 (a) and (b)** Simple linear regression can be applied to yield the following fit




**(c)** The minimum lane width corresponding to a bike-car distance of 2 m can be computed as

$$y = 0.7335(2) + 0.7167 = 2.1837 \text{ m}$$

**20.21** This problem is ideally suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$x_0 = 20$ $f(x_0) = 8.17$
$x_1 = 15$ $f(x_1) = 9.03$
$x_2 = 25$ $f(x_2) = 7.46$
$x_3 = 10$ $f(x_3) = 10.1$
$x_4 = 30$ $f(x_4) = 6.85$
$x_5 = 5$ $f(x_5) = 11.3$
$x_6 = 0$ $f(x_6) = 12.9$

The results of applying Newton's polynomial at $T$ = 18 are

| Order | *f*(*x*) | Error |
|---|---|---|
| 0 | 8.17000 | 0.344 |

| | | |
|---|---|---|
| 1 | 8.51400 | -0.018 |
| 2 | 8.49600 | -0.00336 |
| 3 | 8.49264 | 0.00022 |
| 4 | 8.49286 | -0.00161 |
| 5 | 8.49125 | -0.00298 |
| 6 | 8.48827 | |

The minimum error occurs for the third-order version so we conclude that the interpolation is 8.4926.

**20.22** We can use MATLAB to solve this problem,

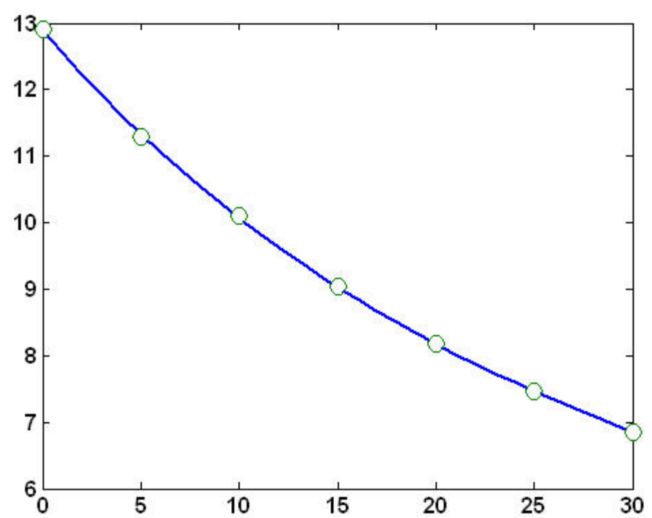
```
>> format long
>> T = [0 5 10 15 20 25 30];
>> c = [12.9 11.3 10.1 9.03 8.17 7.46 6.85];
>> p = polyfit(T,c,3)
p =
  -0.00006222222222   0.00652380952381  -0.34111111111111
12.88785714285715
```

Thus, the best-fit cubic would be

$$c = 12.8878571 - 0.34111111T + 0.00652381T^2 - 0.000062222T^3$$

We can generate a plot of the data along with the best-fit line as

```
>> Tp=[0:30];
>> cp=polyval(p,Tp);
>> plot(Tp,cp,T,c,'o')
```




We can use the best-fit equation to generate a prediction at $T = 8$ as

```
>> y=polyval(p,8)
y =
  10.54463428571429
```

**20.23** The multiple linear regression model to evaluate is

$o = a_0 + a_1 T + a_2 c$

As described in Section 17.1, we can use general linear least squares to generate the best-fit equation. The [$Z$] and $y$ matrices can be set up using MATLAB commands as

```
>> format long
>> t = [0 5 10 15 20 25 30];
>> T = [t t t]';
>> c = [zeros(size(t)) 10*ones(size(t)) 20*ones(size(t))]';
>> Z = [ones(size(T)) T c];
>> y = [14.6 12.8 11.3 10.1 9.09 8.26 7.56 12.9 11.3 10.1 9.03 8.17 7.46 6.85
11.4 10.3 8.96 8.08 7.35 6.73 6.2]';
```

The coefficients can be evaluated as

```
>> a=inv(Z'*Z)*(Z'*y)
a =
  13.52214285714285
  -0.20123809523809
  -0.10492857142857
```

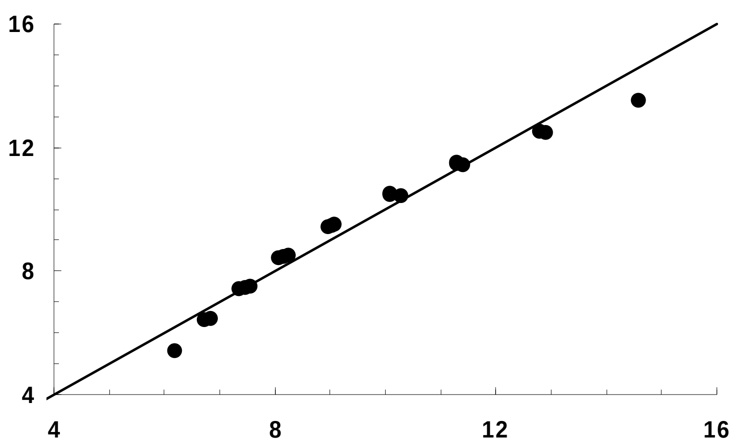
Thus, the best-fit multiple regression model is

$$o = 13.52214285714285 - 0.20123809523809T - 0.10492857142857c$$

We can evaluate the prediction at $T$ = 17 and $c$ = 5 and evaluate the percent relative error as

```
>> op = a(1)+a(2)*17+a(3)*5
op =
   9.57645238095238
```

We can also assess the fit by plotting the model predictions versus the data. A one-to-one line is included to show how the predictions diverge from a perfect fit.



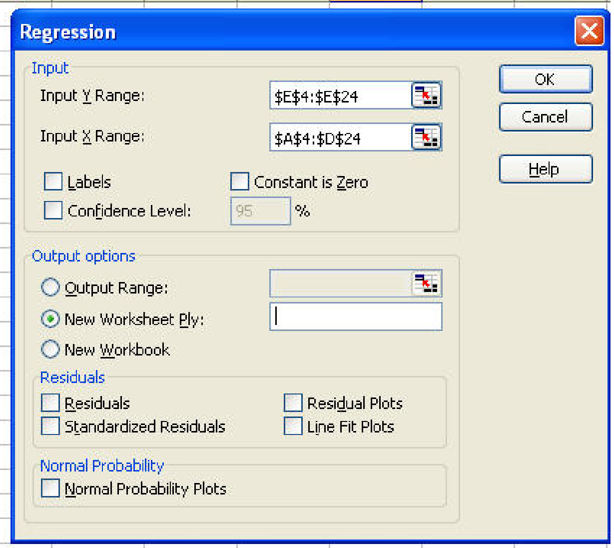

The cause for the discrepancy is because the dependence of oxygen concentration on the unknowns is significantly nonlinear. It should be noted that this is particularly the case for the dependency on temperature.

**20.24** The multiple linear regression model to evaluate is

$$o = a_0 + a_1 T + a_2 T^2 + a_3 T^3 + a_4 c$$

The Excel Data Analysis toolpack provides a convenient tool to fit this model. We can set up a worksheet and implement the Data Analysis Regression tool as

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Problem 20.6 | | | | |
| 2 | | | | | |
| 3 | T, oC | t^2 | T^3 | c | o |
| 4 | 0 | 0 | 0 | 0 | 14.6 |
| 5 | 5 | 25 | 125 | 0 | 12.8 |
| 6 | 10 | 100 | 1000 | 0 | 11.3 |
| 7 | 15 | 225 | 3375 | 0 | 10.1 |
| 8 | 20 | 400 | 8000 | 0 | 9.09 |
| 9 | 25 | 625 | 15625 | 0 | 8.26 |
| 10 | 30 | 900 | 27000 | 0 | 7.56 |
| 11 | 0 | 0 | 0 | 10 | 12.9 |
| 12 | 5 | 25 | 125 | 10 | 11.3 |
| 13 | 10 | 100 | 1000 | 10 | 10.1 |
| 14 | 15 | 225 | 3375 | 10 | 9.03 |
| 15 | 20 | 400 | 8000 | 10 | 8.17 |
| 16 | 25 | 625 | 15625 | 10 | 7.46 |
| 17 | 30 | 900 | 27000 | 10 | 6.85 |
| 18 | 0 | 0 | 0 | 20 | 11.4 |
| 19 | 5 | 25 | 125 | 20 | 10.3 |
| 20 | 10 | 100 | 1000 | 20 | 8.96 |
| 21 | 15 | 225 | 3375 | 20 | 8.08 |
| 22 | 20 | 400 | 8000 | 20 | 7.35 |
| 23 | 25 | 625 | 15625 | 20 | 6.73 |
| 24 | 30 | 900 | 27000 | 20 | 6.2 |



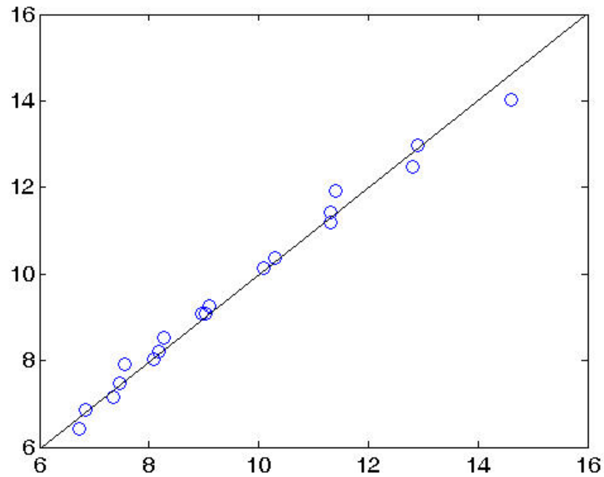

When the tool is run, the following worksheet is generated

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SUMMARY OUTPUT | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | *Regression Statistics* | | | | | | | | |
| 4 | Multiple R | 0.993835141 | | | | | | | |
| 5 | R Square | 0.987708288 | | | | | | | |
| 6 | Adjusted R Square | 0.98463536 | | | | | | | |
| 7 | Standard Error | 0.282662722 | | | | | | | |
| 8 | Observations | 21 | | | | | | | |
| 9 | | | | | | | | | |
| 10 | ANOVA | | | | | | | | |
| 11 | | *df* | *SS* | *MS* | *F* | *Significance F* | | | |
| 12 | Regression | 4 | 102.7243429 | 25.68109 | 321.4225 | 4.63842E-15 | | | |
| 13 | Residual | 16 | 1.278371429 | 0.079898 | | | | | |
| 14 | Total | 20 | 104.0027143 | | | | | | |
| 15 | | | | | | | | | |
| 16 | | *Coefficients* | *Standard Error* | *t Stat* | *P-value* | *Lower 95%* | *Upper 95%* | *ower 95.0%* | *pper 95.0%* |
| 17 | Intercept | 14.02714286 | 0.17446322 | 80.40172 | 2.71E-22 | 13.65729736 | 14.39699 | 13.6573 | 14.39699 |
| 18 | X Variable 1 | -0.33642328 | 0.049672965 | -6.77276 | 4.48E-06 | -0.441725261 | -0.23112 | -0.44173 | -0.23112 |
| 19 | X Variable 2 | 0.005744444 | 0.00406041 | 1.414745 | 0.17631 | -0.002863241 | 0.014352 | -0.00286 | 0.014352 |
| 20 | X Variable 3 | -4.37037E-05 | 8.88323E-05 | -0.49198 | 0.629415 | -0.00023202 | 0.000145 | -0.00023 | 0.000145 |
| 21 | X Variable 4 | -0.104928571 | 0.007554479 | -13.8896 | 2.41E-10 | -0.120943351 | -0.08891 | -0.12094 | -0.08891 |

Thus, the best-fit model is

$$o_s = 14.027143 - 0.336423T + 0.00574444T^2 - 0.000043704T^3 - 0.10492857c$$

The model can then be used to predict values of oxygen at the same values as the data. These predictions can be plotted against the data to depict the goodness of fit.

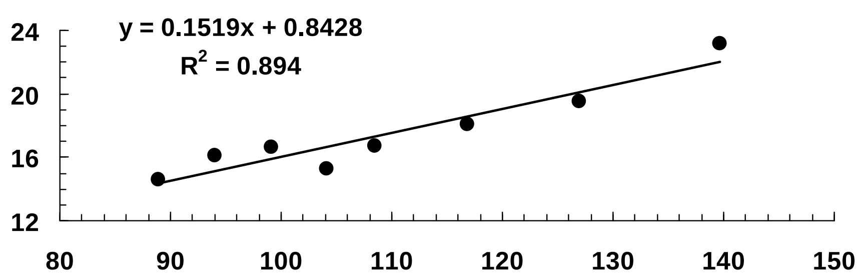



Thus, although there are some discrepancies, the fit is generally adequate. Finally, the prediction can be made at $T = 20$ and $c = 10$,

$$o_s = 14.027143 - 0.336423(20) + 0.00574444(20)^2 - 0.000043704(20)^3 - 0.10492857(10) = 8.19754$$

which compares favorably with the true value of 8.17 mg/L.

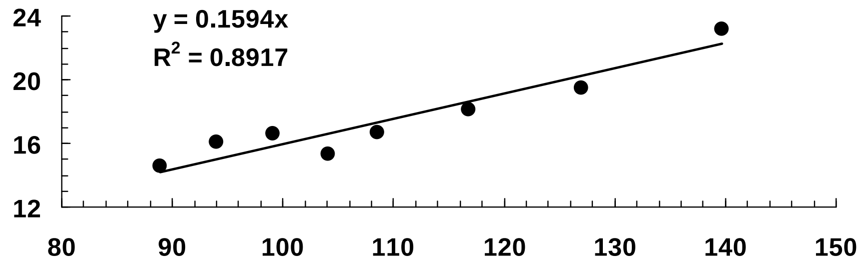
**20.25 (a) and (b)** Simple linear regression can be applied to yield the following fit




**(c)** The flow corresponding to the precipitation of 120 cm can be computed as

$$Q = 0.1519(120) + 0.8428 = 19.067$$

**(d)** We can redo the regression, but with a zero intercept




Thus, the model is

$$Q = 0.1594P$$

where $Q$ = flow and $P$ = precipitation. Now, if there are no water losses, the maximum flow, $Q_m$, that could occur for a level of precipitation should be equal to the product of the annual precipitation and the drainage area. This is expressed by the following equation.

$$Q_m = A(\text{km}^2)P\left(\frac{\text{cm}}{\text{yr}}\right)$$

For an area of 1100 km$^2$ and applying conversions so that the flow has units of m$^3$/s

$$Q_m = 1{,}100\ \text{km}^2 P\left(\frac{\text{cm}}{\text{yr}}\right)\frac{10^6\,\text{m}^2}{\text{km}^2}\frac{1\text{ m}}{100\text{ cm}}\frac{\text{d}}{86{,}400\text{ s}}\frac{\text{yr}}{365\text{ d}}$$

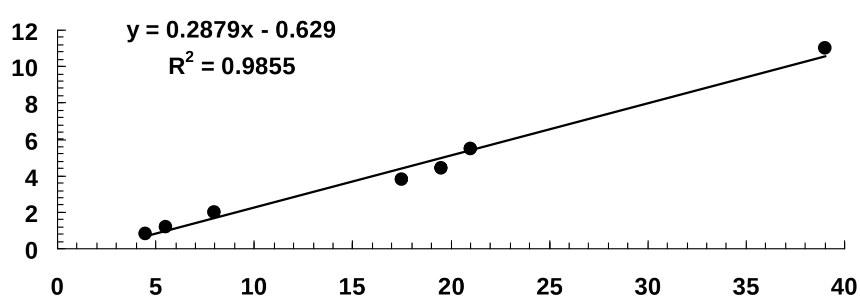
Collecting terms gives

$$Q_m = 0.348808P$$

Using the slope from the linear regression with zero intercept, we can compute the fraction of the total flow that is lost to evaporation and other consumptive uses can be computed as

$$F = \frac{0.348808 - 0.1594}{0.348808} = 0.543$$

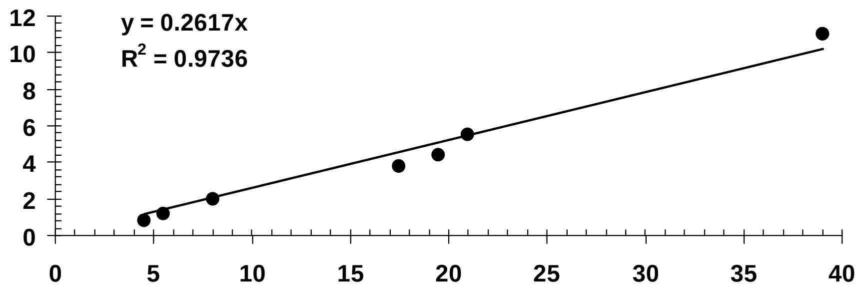
**20.26** The data can be computed in a number of ways. First, we can use linear regression.



For this model, the chlorophyll level in western Lake Erie corresponding to a phosphorus concentration of 10 mg/m$^3$ is

$$c = 0.2879(10) - 0.629 = 2.2495$$

One problem with this result is that the model yields a physically unrealistic negative intercept. In fact, because chlorophyll cannot exist without phosphorus, a fit with a zero intercept would be preferable. Such a fit can be developed as

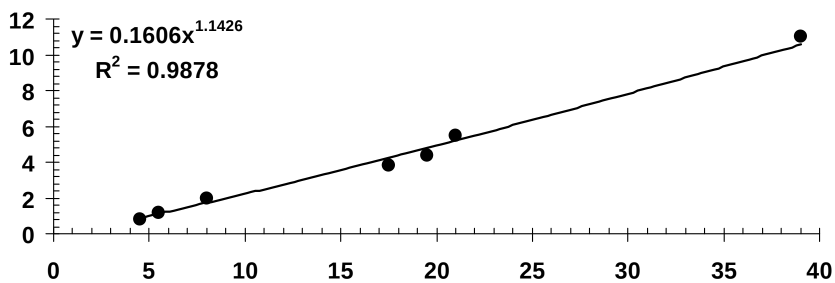



For this model, the chlorophyll level in western Lake Erie corresponding to a phosphorus concentration of 10 mg/m$^3$ is

$$c = 0.2617(10) = 2.617$$

Thus, as expected, the result differs from that obtained with a nonzero intercept.

Finally, it should be noted that in practice, such data is usually fit with a power model. This model is often adopted because it has a zero intercept while not constraining the model to be linear. When this is done, the result is




For this model, the chlorophyll level in western Lake Erie corresponding to a phosphorus concentration of 10 mg/m$^3$ is

$$c = 0.1606(10)^{1.1426} = 2.2302$$

**20.27**

$$m = \frac{4.6}{10} = 0.46 \qquad n = \frac{14}{10} = 1.4$$

Use the following values

$x_0 = 0.3$ $f(x_0) = 0.08561$
$x_1 = 0.4$ $f(x_1) = 0.10941$
$x_2 = 0.5$ $f(x_2) = 0.13003$
$x_3 = 0.6$ $f(x_3) = 0.14749$

MATLAB can be used to perform the interpolation

```
>> format long
>> x=[0.3 0.4 0.5 0.6];
>> y=[0.08561 0.10941 0.13003 0.14749];
>> a=polyfit(x,y,3)
a =
  Columns 1 through 3
   0.00333333333331  -0.16299999999997   0.35086666666665
  Column 4
  -0.00507000000000
>> polyval(a,0.46)
ans =
   0.12216232000000
```

This result can be used to compute the vertical stress

$$q = \frac{100}{4.6(14)} = 1.552795$$

$$\sigma_z = 1.552795(0.1221623) = 0.189693$$

**20.28** This is an ideal problem for general linear least squares. The problem can be solved with MATLAB:

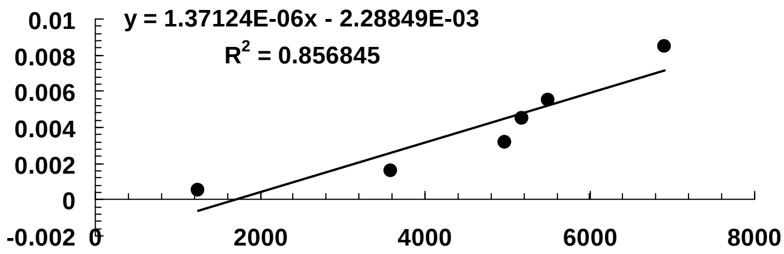
```
>> t=[0.5 1 2 3 4 5 6 7 9]';
>> p=[6 4.4 3.2 2.7 2.2 1.9 1.7 1.4 1.1]';
>> Z=[exp(-1.5*t) exp(-0.3*t) exp(-0.05*t)];
>> a=inv(Z'*Z)*(Z'*p)
a =
    4.0046
    2.9213
    1.5647
```

Therefore, $A$ = 4.0046, $B$ = 2.9213, and $C$ = 1.5647.

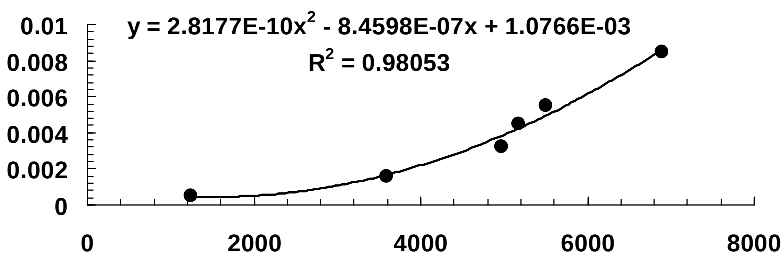
**20.29** First, we can determine the stress

$$\sigma = \frac{25000}{10.65} = 2,347.418$$

We can then try to fit the data to obtain a mathematical relationship between strain and stress. First, we can try linear regression:




This is not a particularly good fit as the $r^2$ is relatively low. We therefore try a best-fit parabola,



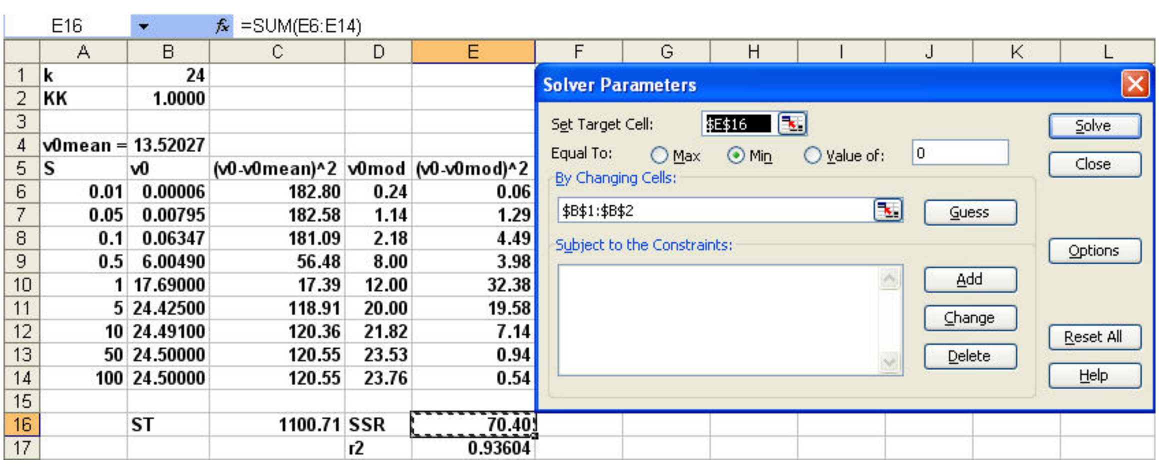

We can use this model to compute the strain as

$$\varepsilon = 2.8177\times10^{-10}(2347.4178)^2 - 8.4598\times10^{-7}(2347.4178) + 1.0766\times10^{-3} = 6.4341\times10^{-4}$$

The deflection can be computed as

$$\Delta L = 6.4341\times10^{-4}(9) = 0.0057907 \text{ m}$$

**20.30** Clearly the linear model is not adequate. The second model can be fit with the Excel Solver:

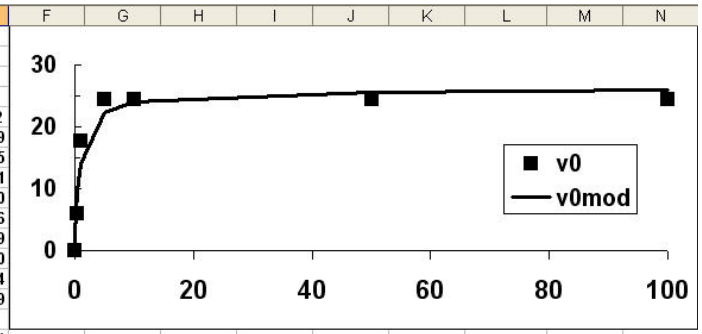

E16 =SUM(E6:E14)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | k | 24 | | | |
| 2 | KK | 1.0000 | | | |
| 3 | | | | | |
| 4 | v0mean = | 13.52027 | | | |
| 5 | S | v0 | (v0-v0mean)^2 | v0mod | (v0-v0mod)^2 |
| 6 | 0.01 | 0.00006 | 182.80 | 0.24 | 0.06 |
| 7 | 0.05 | 0.00795 | 182.58 | 1.14 | 1.29 |
| 8 | 0.1 | 0.06347 | 181.09 | 2.18 | 4.49 |
| 9 | 0.5 | 6.00490 | 56.48 | 8.00 | 3.98 |
| 10 | 1 | 17.69000 | 17.39 | 12.00 | 32.38 |
| 11 | 5 | 24.42500 | 118.91 | 20.00 | 19.58 |
| 12 | 10 | 24.49100 | 120.36 | 21.82 | 7.14 |
| 13 | 50 | 24.50000 | 120.55 | 23.53 | 0.94 |
| 14 | 100 | 24.50000 | 120.55 | 23.76 | 0.54 |
| 15 | | | | | |
| 16 | | ST | 1100.71 | SSR | 70.40 |
| 17 | | | | r2 | 0.93604 |

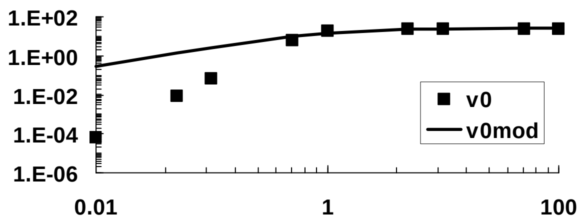


Notice that we have reexpressed the initial rates by multiplying them by $1\times10^6$. We did this so that the sum of the squares of the residuals would not be miniscule. Sometimes this will lead the Solver to conclude that it is at the minimum, even though the fit is poor. The solution is:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | k | 26.097 | | | |
| 2 | KK | 0.8558 | | | |
| 3 | | | | | |
| 4 | v0mean = | 13.52027 | | | |
| 5 | S | v0 | (v0-v0mean)^2 | v0mod | (v0-v0mod)^2 |
| 6 | 0.01 | 0.00006 | 182.80 | 0.30 | 0.09 |
| 7 | 0.05 | 0.00795 | 182.58 | 1.44 | 2.05 |
| 8 | 0.1 | 0.06347 | 181.09 | 2.73 | 7.11 |
| 9 | 0.5 | 6.00490 | 56.48 | 9.62 | 13.10 |
| 10 | 1 | 17.69000 | 17.39 | 14.06 | 13.16 |
| 11 | 5 | 24.42500 | 118.91 | 22.28 | 4.59 |
| 12 | 10 | 24.49100 | 120.36 | 24.04 | 0.20 |
| 13 | 50 | 24.50000 | 120.55 | 25.66 | 1.34 |
| 14 | 100 | 24.50000 | 120.55 | 25.88 | 1.89 |
| 15 | | | | | |
| 16 | | ST | 1100.71 | SSR | 43.54 |
| 17 | | | | r2 | 0.96044 |



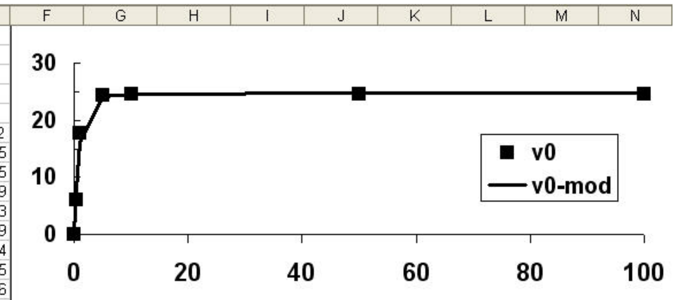

Although the fit might appear to be OK, it is biased in that it underestimates the low values and overestimates the high ones. The poorness of the fit is really obvious if we display the results as a log-log plot:

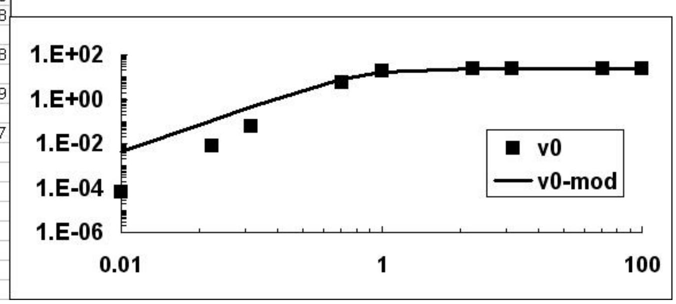



Notice that this view illustrates that the model actually overpredicts the very lowest values.

The third and fourth models provide a means to rectify this problem. Because they raise [*S*] to powers, they have more degrees of freedom to follow the underlying pattern of the data. For example, the third model gives:

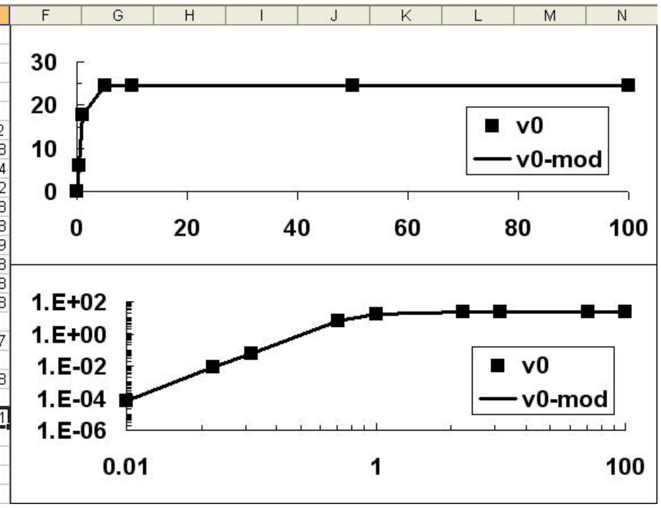
| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | k | 24.77759718 | | | |
| 2 | KK | 0.5475 | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | mean | 13.52027 | | | |
| 6 | S | v0 | (v0 - v0mean)^2 | v0-mod | (v0 - v0mod)^2 |
| 7 | 0.01 | 0.00006 | 182.7958528 | 0.004525 | 1.99039E-05 |
| 8 | 0.05 | 0.00795 | 182.5826105 | 0.112632 | 0.010957845 |
| 9 | 0.1 | 0.06347 | 181.0852857 | 0.444466 | 0.145156179 |
| 10 | 0.5 | 6.00490 | 56.48071549 | 7.767571 | 3.107008233 |
| 11 | 1 | 17.69000 | 17.38668753 | 16.01169 | 2.816728109 |
| 12 | 5 | 24.42500 | 118.913239 | 24.24663 | 0.03181734 |
| 13 | 10 | 24.49100 | 120.35702 | 24.64269 | 0.023008655 |
| 14 | 50 | 24.50000 | 120.5545742 | 24.77217 | 0.074077806 |
| 15 | 100 | 24.50000 | 120.5545742 | 24.77624 | 0.076308958 |
| 16 | | | | | |
| 17 | | ST | 1100.71056 | SSR | 6.285083028 |
| 18 | | | | | |
| 19 | | | | log(SSR) | 0.798311019 |
| 20 | | | | | |
| 21 | | | | r2 | 0.994289977 |
| 22 | | | | | |
| 23 | | | | | |
| 24 | | | | | |
| 25 | | | | | |
| 26 | | | | | |
| 27 | | | | | |
| 28 | | | | | |
| 29 | | | | | |




Finally, the cubic model results in a perfect fit:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | k | 24.50019166 | | | |
| 2 | KK | 0.3850 | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | mean | 13.52027 | | | |
| 6 | S | v0 | (v0 - v0mean)^2 | v0-mod | (v0 - v0mod)^2 |
| 7 | 0.01 | 0.00006 | 182.7958528 | 6.36387E-05 | 7.34596E-18 |
| 8 | 0.05 | 0.00795 | 182.5826105 | 0.007952277 | 7.69871E-14 |
| 9 | 0.1 | 0.06347 | 181.0852857 | 0.063474003 | 4.01155E-12 |
| 10 | 0.5 | 6.00490 | 56.48071549 | 6.005092415 | 3.70235E-08 |
| 11 | 1 | 17.69000 | 17.38668753 | 17.6898247 | 3.07288E-08 |
| 12 | 5 | 24.42500 | 118.913239 | 24.42496515 | 1.21484E-09 |
| 13 | 10 | 24.49100 | 120.35702 | 24.49076301 | 5.61635E-08 |
| 14 | 50 | 24.50000 | 120.5545742 | 24.5001162 | 1.35022E-08 |
| 15 | 100 | 24.50000 | 120.5545742 | 24.50018222 | 3.32059E-08 |
| 16 | | | | | |
| 17 | | ST | 1100.71056 | SSR | 1.71843E-07 |
| 18 | | | | | |
| 19 | | | | log(SSR) | -6.7648688 |
| 20 | | | | | |
| 21 | | | | r2 | 1 |
| 22 | | | | | |
| 23 | | | | | |
| 24 | | | | | |
| 25 | | | | | |




Thus, the best fit is

$$v_0 = \frac{2.45 \times 10^{-5}[S]^3}{0.385 + [S]^3}$$

**20.31** As described in Section 17.1, we can use general linear least squares to generate the best-fit equation. We can use a number of different software tools to do this. For example, the [*Z*] and *y* matrices can be set up using MATLAB commands as

```
>> format long
>> x = [273.15 283.15 293.15 303.15 313.15]';
>> Kw = [1.164e-15 2.950e-15 6.846e-15 1.467e-14 2.929e-14]';
>> y=-log10(Kw);
>> Z = [(1./x) log10(x) x ones(size(x))];
```

The coefficients can be evaluated as

```
>> a=inv(Z'*Z)*(Z'*y)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 6.457873e-020.
a =
  1.0e+003 *
   5.18067187500000
   0.01342456054688
   0.00000562858582
  -0.03827636718750
```

Note the warning that the results are ill-conditioned. According to this calculation, the best-fit model is

$$-\log_{10} K_w = \frac{5180.67}{T_a} + 13.42456 \log_{10} T_a + 0.00562859 T_a - 38.276367$$

We can check the results by using the model to make predictions at the values of the original data

```
>> yp=10.^-(a(1)./x+a(2)*log10(x)+a(3)*x+a(4))
yp =
  1.0e-013 *
   0.01161193308242
```

```
0.02943235714551
0.06828461729494
0.14636330575049
0.29218444886852
```

These results agree to about 2 or 3 significant digits with the original data.

**20.32** The equation can be linearized by inverting it to yield

$$\frac{1}{k}=\frac{c_s}{k_{\max}}\frac{1}{c^2}+\frac{1}{k_{\max}}$$

Consequently, a plot of $1/k$ versus $1/c$ should yield a straight line with an intercept of $1/k_{\max}$ and a slope of $c_s/k_{\max}$

| ***c*, mg/L** | ***k*, /d** | **1/*c*²** | **1/*k*** | **1/*c*²×1/*k*** | **(1/*c*²)²** |
|---|---|---|---|---|---|
| 0.5 | 1.1 | 4.000000 | 0.909091 | 3.636364 | 16.000000 |
| 0.8 | 2.4 | 1.562500 | 0.416667 | 0.651042 | 2.441406 |
| 1.5 | 5.3 | 0.444444 | 0.188679 | 0.083857 | 0.197531 |
| 2.5 | 7.6 | 0.160000 | 0.131579 | 0.021053 | 0.025600 |
| 4 | 8.9 | 0.062500 | 0.112360 | 0.007022 | 0.003906 |
| | **Sum →** | **6.229444** | **1.758375** | **4.399338** | **18.66844** |

The slope and the intercept can be computed as

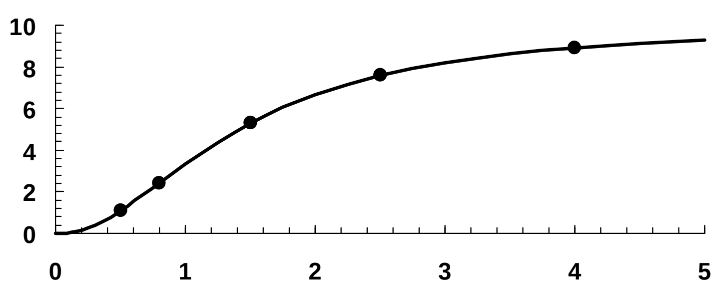
$$a_1=\frac{5(4.399338)-6.229444(1.758375)}{5(18.66844)-(6.229444)^2}=0.202489$$

$$a_0=\frac{1.758375}{5}-0.202489\frac{6.229444}{5}=0.099396$$

Therefore, $k_{\max}$ = 1/0.099396 = 10.06074 and $c_s$ = 10.06074(0.202489) = 2.037189, and the fit is

$$k=\frac{10.06074c^2}{2.037189+c^2}$$

This equation can be plotted together with the data:




The equation can be used to compute

$$k=\frac{10.06074(2)^2}{2.037189+(2)^2}=6.666$$

**20.33** The equation can be linearized by taking its logarithm,

$$\log k = \log k_{20} + \log \theta (T - 20)$$

Consequently, a plot of $k$ versus $T$ – 20 should yield a straight line with an intercept of log $k_{20}$ and a slope of log $\theta$.

| *T* | *k* | *T* – 20 | log *k* |
|---|---|---|---|
| 6 | 0.14 | -14 | -0.85387 |
| 12 | 0.2 | -8 | -0.69897 |
| 18 | 0.31 | -2 | -0.50864 |
| 24 | 0.46 | 4 | -0.33724 |
| 30 | 0.69 | 10 | -0.16115 |

Regression can be used to develop the line of best fit as

$$\log k = -0.45374 + 0.029119(T - 20)$$

which can be used to compute

$$k_{20} = 10^{-0.45374} = 0.35174$$
$$\theta = 10^{0.029119} = 1.069349$$

The reaction rate at $T$ = 17 can then be computed as

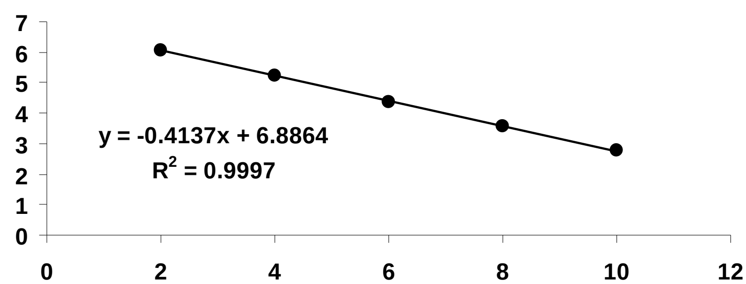
$$k = 0.35177(1.069349)^{17-20} = 0.287677$$

**20.34** Bacterial die-off often follows an exponential model (Eq. 17.12),

$$c = c_0 e^{-kt}$$

If this is the case, a plot of ln $c$ versus $t$ should yield a straight line with a slope of $-k$ and an intercept of ln $c_0$.

| *t* | ln *c* |
|---|---|
| 2 | 6.063785 |
| 4 | 5.247024 |
| 6 | 4.382027 |
| 8 | 3.555348 |
| 10 | 2.772589 |




Therefore, the exponential model holds, and

$$k = 0.4137$$
$$c_0 = e^{6.8864} = 978.85$$

and the final model is

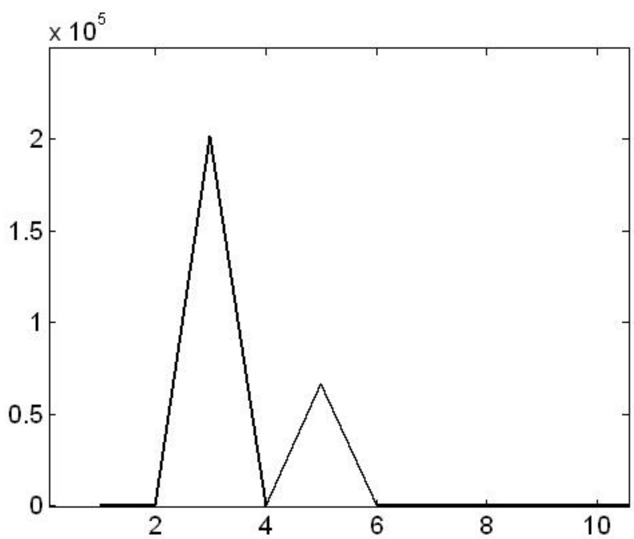
$$c = 978.85e^{-0.4137t}$$

The time that it will be safe to drink the water can be computed by substituting $c$ = 5 and solving for

$$t = \frac{\ln\left(\frac{5}{978.85}\right)}{-0.4137} = 12.75536 \text{ d}$$

**20.35** Using MATLAB:

```
>> t=0:2*pi/128:2*pi;
>> f=4*cos(5*t)-7*sin(3*t)+6;
>> y=fft(f);
>> y(1)=[ ];
>> n=length(y);
>> power=abs(y(1:n/2)).^2;
>> nyquist=1/2;
>> freq=n*(1:n/2)/(n/2)*nyquist;
>> plot(freq,power);
```




**20.36** Since we do not know the proper order of the interpolating polynomial, this problem is suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$x_0$ = 1.25$f(x_0)$ = 0.7
$x_1$ = 0.75$f(x_1)$ = –0.6
$x_2$ = 1.5  $f(x_2)$ = 1.88
$x_3$ = 0.25$f(x_3)$ = –0.45
$x_4$ = 2     $f(x_4)$ = 6

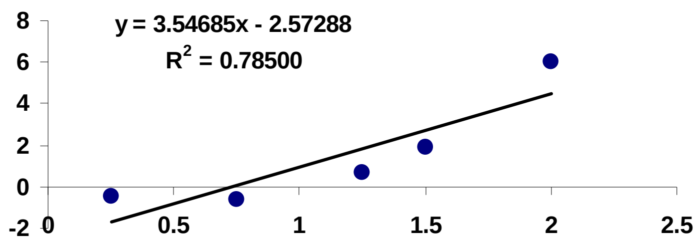
The results of applying Newton's polynomial at $i$ = 1.15 are

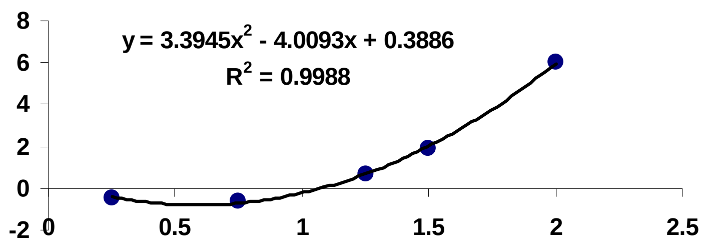
| **Order** | ***f*(*x*)** | **Error** |
|---|---|---|

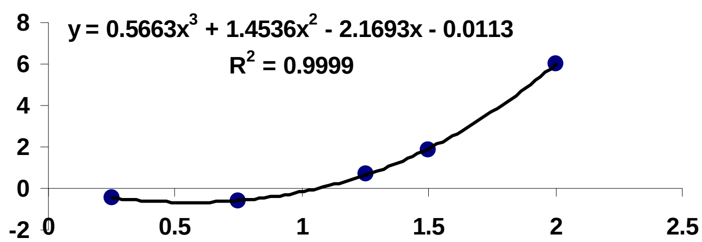
| | | |
|---|---|---|
| 0 | 0.7 | -0.26 |
| 1 | 0.44 | -0.11307 |
| 2 | 0.326933 | -0.00082 |
| 3 | 0.326112 | 0.011174 |
| 4 | 0.337286 | |

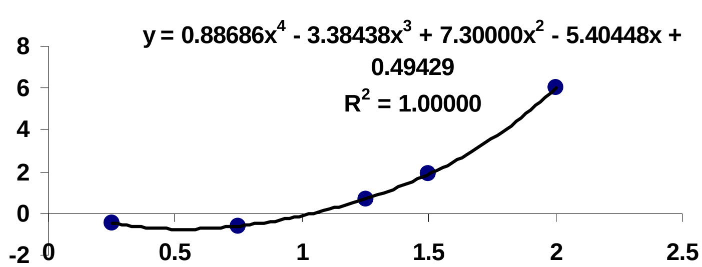
The minimum error occurs for the second-order version so we conclude that the interpolation is 0.3269.

**20.37** Here are the results of first through fourth-order regression:




The 2nd through 4th-order polynomials all seem to capture the general trend of the data. Each of the polynomials can be used to make the prediction at $i$ = 1.15 with the results tabulated below:

| Order | Prediction |
|---|---|
| 1 | 1.5060 |
| 2 | 0.2670 |
| 3 | 0.2776 |
| 4 | 0.3373 |

Thus, although the 2nd through 4th-order polynomials all seem to follow a similar trend, they yield quite different predictions. The results are so sensitive because there are few data points.

**20.38** Since we do not know the proper order of the interpolating polynomial, this problem is suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

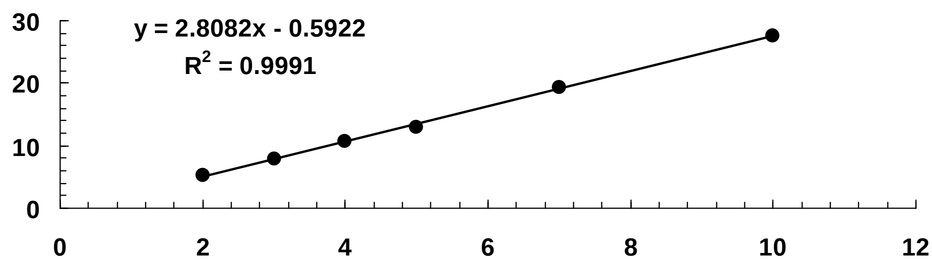
$x_0 = 0.25 f(x_0) = 7.75$
$x_1 = 0.125 \quad f(x_1) = 6.24$
$x_2 = 0.375 \quad f(x_2) = 4.85$
$x_3 = 0 \quad f(x_3) = 0$
$x_4 = 0.5 \quad f(x_4) = 0$

The results of applying Newton's polynomial at $t = 0.23$ are

| Order | $f(x)$ | Error |
|---|---|---|
| 0 | 7.75 | -0.2416 |
| 1 | 7.5084 | 0.296352 |
| 2 | 7.804752 | 0.008315 |
| 3 | 7.813067 | 0.025579 |
| 4 | 7.838646 | |

The minimum error occurs for the second-order version so we conclude that the interpolation is 7.805.

**20.39 (a)** The linear fit is

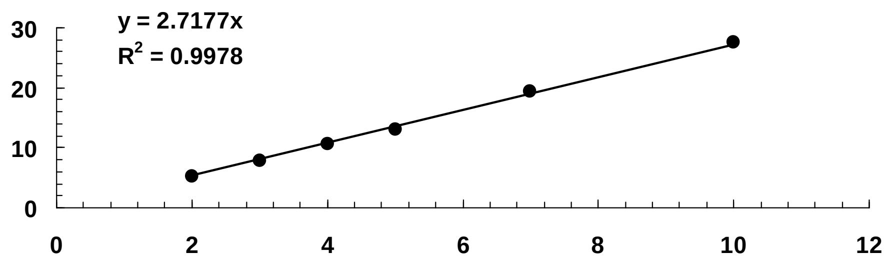



The current for a voltage of 3.5 V can be computed as

$$y = 2.8082(3.5) - 0.5922 = 9.2364$$

Both the graph and the $r^2$ indicate that the fit is good.

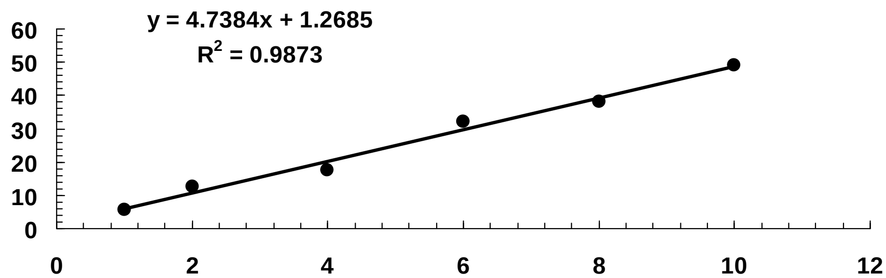
**(b)** A straight line with zero intercept can be fit as



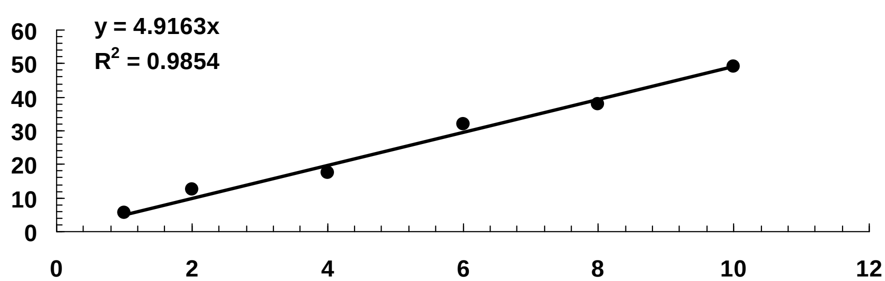

For this case, the current at $V = 3.5$ can be computed as

$$y = 2.7177(3.5) = 9.512$$

**20.40** The linear fit is




Therefore, an estimate for $L$ is 4.7384. However, because there is a non-zero intercept, a better approach would be to fit the data with a linear model with a zero intercept




This fit is almost as good as the first case, but has the advantage that it has the physically more realistic zero intercept. Thus, a superior estimate of $L$ is 4.9163.

**20.41** Since we do not know the proper order of the interpolating polynomial, this problem is suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$x_0 = 0.5$ $f(x_0) = 20.5$
$x_1 = -0.5$ $f(x_1) = -20.5$
$x_2 = 1$ $f(x_2) = 96.5$
$x_3 = -1$ $f(x_3) = -96.5$
$x_4 = 2$ $f(x_4) = 637$

$x_5 = -2 \quad f(x_5) = -637$

The results of applying Newton's polynomial at $i = 0.1$ are

| Order | f(x) | Error |
|---|---|---|
| 0 | 20.5 | -16.4 |
| 1 | 4.1 | -17.76 |
| 2 | -13.66 | 15.984 |
| 3 | 2.324 | 0 |
| 4 | 2.324 | 0 |
| 5 | 2.324 | |

Thus, we can see that the data was generated with a cubic polynomial.

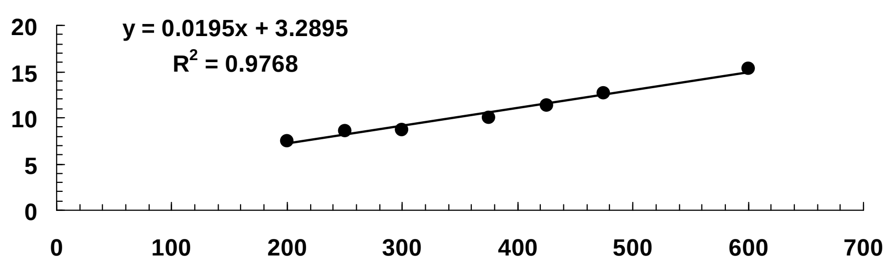
**20.42** Because there are 6 points, we can fit a 5$^{th}$-order polynomial. This can be done with Eq. 18.26 or with a software package like Excel or MATLAB that is capable of evaluating the coefficients. For example, using MATLAB,

```
>> x=[-2 -1 -0.5 0.5 1 2];
>> y=[-637 -96.5 -20.5 20.5 96.5 637];
>> a=polyfit(x,y,5)
a =
    0.0000    0.0000   74.0000   -0.0000   22.5000    0.0000
```

Thus, we see that the polynomial is

$$V = 74i^3 + 22.5i$$

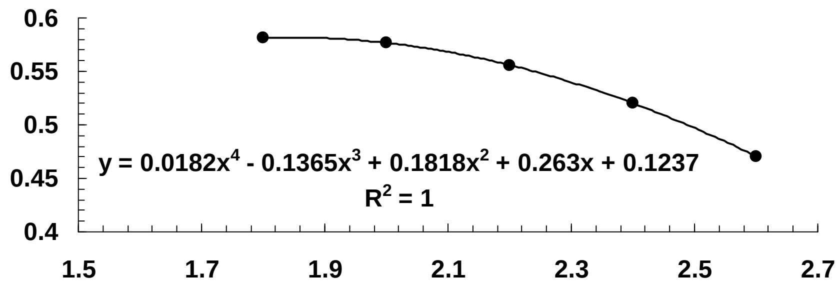
**20.43** Linear regression yields




The percent elongation for a temperature of 400 can be computed as

$$\%\ \text{elongation} = 0.0195(400) + 3.2895 = 11.072$$

**20.44 (a)** A 4$^{th}$-order interpolating polynomial can be generated as




The polynomial can be used to compute

$$J_1(2.1) = 0.018229(2.1)^4 - 0.13646(2.1)^3 + 0.181771(2.1)^2 + 0.262958(2.1) + 0.1237 = 0.568304$$

The relative error is

$$\varepsilon_t = \left|\frac{0.568292 - 0.568304}{0.568292}\right| \times 100\% = 0.0021\%$$

Thus, the interpolating polynomial yields an excellent result.

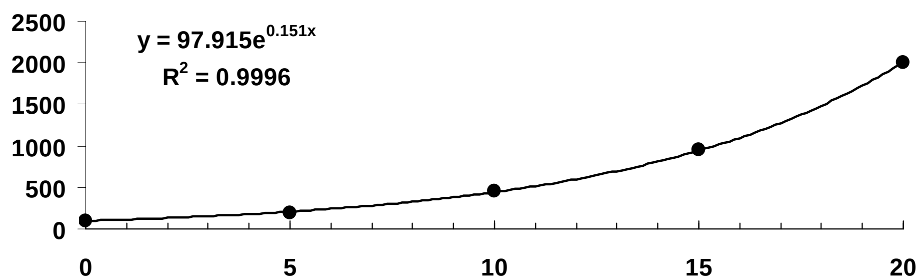
**(b)** A program can be developed to fit natural cubic splines through data based on Fig. 18.18. If this program is run with the data for this problem, the interpolation at 2.1 is 0.56846 which has a relative error of $\varepsilon_t$ = 0.0295%.

A spline can also be fit with MATLAB. It should be noted that MATLAB does not use a natural spline. Rather, it uses a so-called “not-a-knot” spline. Thus, as shown below, although it also yields a very good prediction, the result differs from the one generated with the natural spline,

```
>> format long
>> x=[1.8 2 2.2 2.4 2.6];
>> y=[0.5815 0.5767 0.556 0.5202 0.4708];
>> spline(x,y,2.1)
ans =
   0.56829843750000
```

This result has a relative error of $\varepsilon_t$ = 0.0011%.

**20.45** The fit of the exponential model is




The model can be used to predict the population 5 years in the future as

$$p = 97.91484e^{0.150992(25)} = 4268$$

**20.46** The prediction using the model from Sec. 20.4 is

$$Q = 55.9(1.23)^{2.62}(0.001)^{0.54} = 2.30655$$

The linear prediction is

$x_0 = 1$ $f(x_0) = 1.4$
$x_1 = 2$ $f(x_1) = 8.3$

$$Q = 1.4 + \frac{8.3 - 1.4}{2 - 1}(1.23 - 1) = 2.987$$

The quadratic prediction is

$x_0 = 1$ $f(x_0) = 1.4$
$x_1 = 2$ $f(x_1) = 8.3$
$x_2 = 3$ $f(x_2) = 24.2$

$$Q = 2.987 + \frac{\dfrac{24.2 - 8.3}{3 - 2} - \dfrac{8.3 - 1.4}{2 - 1}}{3 - 1}(1.23 - 1)(1.23 - 2) = 2.19005$$

**20.47** The model to be fit is

$$S = b_0 D^{b_1} Q^{b_2}$$

Taking the common logarithm gives

$$\log_{10} S = \log_{10} b_0 + b_1 \log_{10} D + b_2 \log_{10} Q$$

We can use a number of different approaches to fit this model. The following shows how it can be done with MATLAB,

```
>> D=[1 2 3 1 2 3 1 2 3]';
>> S=[0.001 0.001 0.001 0.01 0.01 0.01 0.05 0.05 0.05]';
>> Q=[1.4 8.3 24.2 4.7 28.9 84 11.1 69 200]';
>> Z=[ones(size(S)) log10(D) log10(Q)];
>> y=log10(S);
>> a=inv(Z'*Z)*(Z'*y)
a =
   -3.2563
   -4.8726
    1.8627
```

Therefore, the result is

$$\log_{10} S = -3.2563 - 4.8726 \log_{10} D + 1.862733 \log_{10} Q$$

or in untransformed format

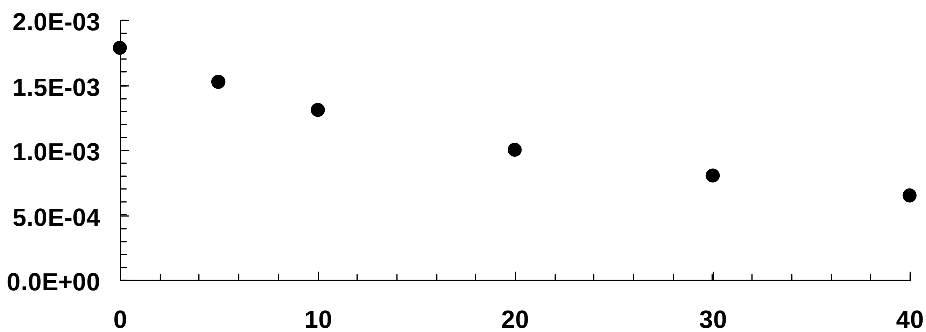
$$S = 0.000554D^{-4.8726}Q^{1.862733} \quad (1)$$

We can compare this equation with the original model developed in Sec. 20.4 by solving Eq. 1 for $Q$,

$$Q = 55.9897D^{2.6158}S^{0.5368}$$

This is very close to the original model.

**20.48 (a)** The data can be plotted as




**(b)** This part of the problem is well-suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

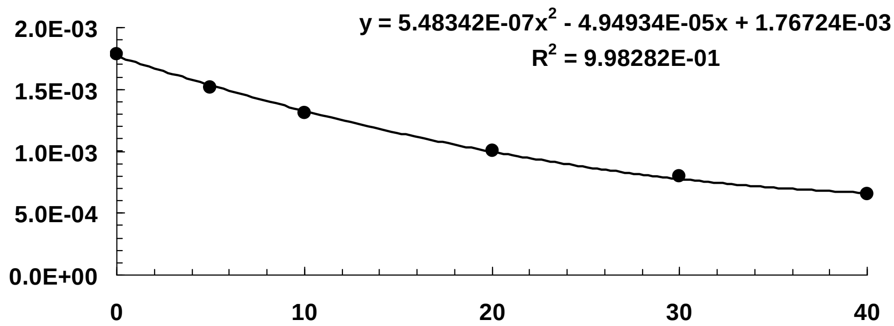
$x_0 = 5 \quad f(x_0) = 1.519\times10^{-3}$
$x_1 = 10 \quad f(x_1) = 1.307\times10^{-3}$
$x_2 = 0 \quad f(x_2) = 1.787\times10^{-3}$
$x_3 = 20 \quad f(x_3) = 1.002\times10^{-3}$
$x_4 = 30 \quad f(x_4) = 0.7975\times10^{-3}$
$x_5 = 40 \quad f(x_5) = 0.6529\times10^{-3}$

The results of applying Newton's polynomial at $T = 7.5$ are

| Order | *f*(*x*) | Error |
|---|---|---|
| 0 | 1.51900E-03 | -0.00011 |
| 1 | 1.41300E-03 | -7E-06 |
| 2 | 1.40600E-03 | 7.66E-07 |
| 3 | 1.40677E-03 | 9.18E-08 |
| 4 | 1.40686E-03 | 5.81E-09 |
| 5 | 1.40686E-03 | |

The minimum error occurs for the fourth-order version so we conclude that the interpolation is $1.40686\times10^{-3}$.

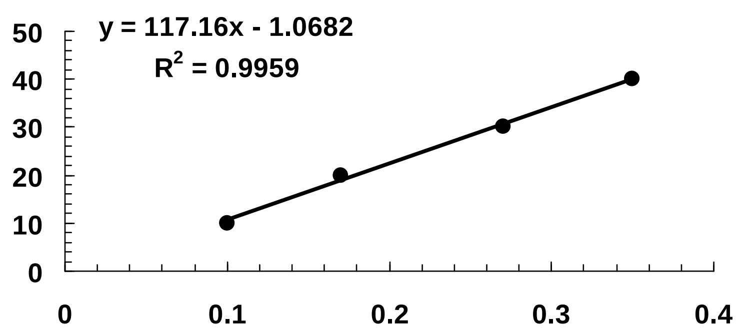
**(b)** Polynomial regression yields



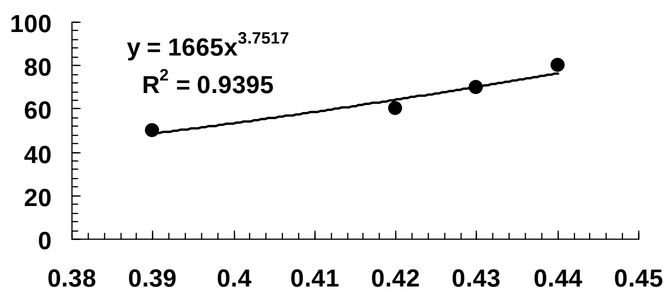

This leads to a prediction of

$$\mu = 5.48342\times10^{-7}(7.5)^2 - 4.94934\times10^{-5}(7.5) + 1.76724\times10^{-3} = 1.4269\times10^{-3}$$

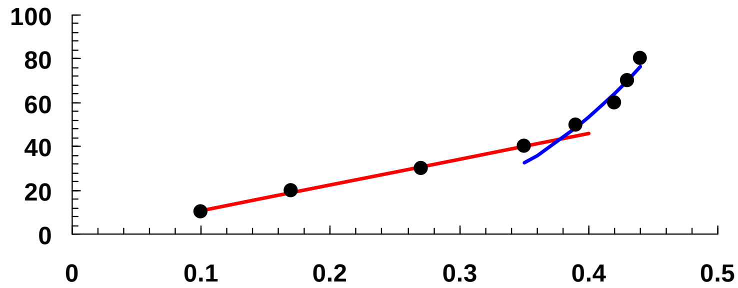
**20.49** For the first four points, linear regression yields



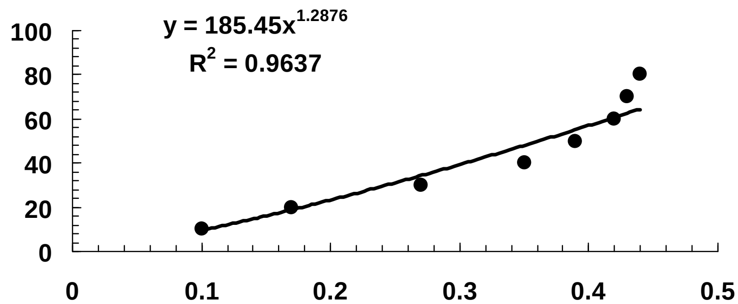

We can fit a power equation to the last four points.




Both curves can be plotted along with the data.

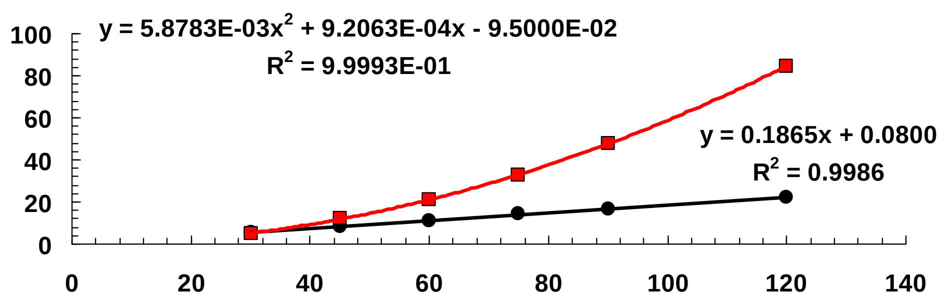



**20.50** A power fit to all the data can be implemented as in the following plot.




Although the coefficient of determination is relatively high ($r^2 = 0.9637$), this fit is not very acceptable. In contrast, the piecewise fit from Prob. 20.45 does a much better job of tracking on the trend of the data.

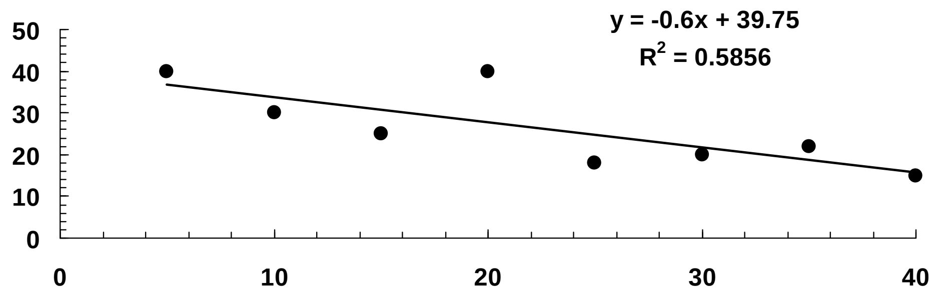
**20.51** The "Thinking" curve can be fit with a linear model whereas the "Braking" curve can be fit with a quadratic model as in the following plot.




A prediction of the total stopping distance for a car traveling at 110 km/hr can be computed as

$$d = 0.1865(110) + 0.0800 + 5.8783\times10^{-3}(110)^2 + 9.2063\times10^{-4}(110) - 9.5000\times10^{-2} = 91.726 \text{ m}$$

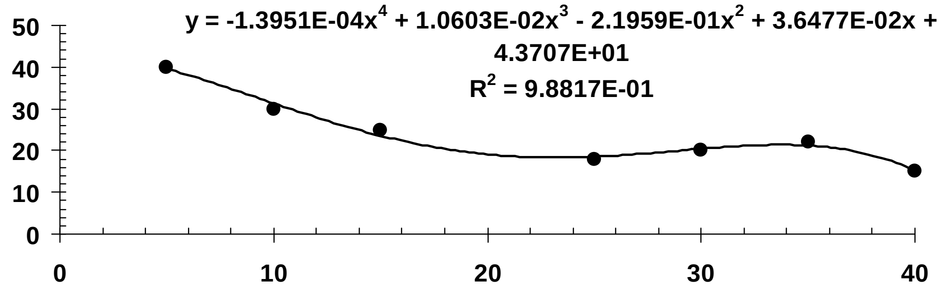
**20.52** Using linear regression gives




A prediction of the fracture time for an applied stress of 20 can be computed as

$$t = -0.6(20) + 39.75 = 27.75$$

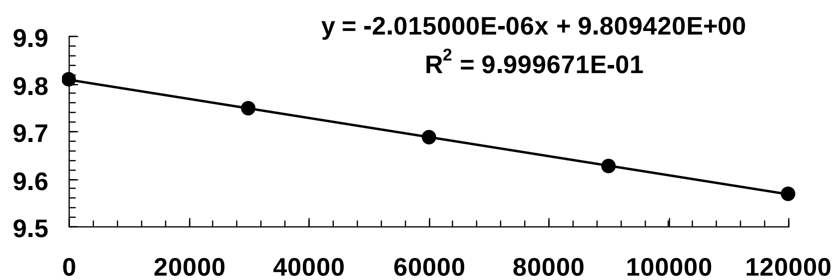
Some students might consider the point at (20, 40) as an outlier and discard it. If this is done, 4th-order polynomial regression gives the following fit




A prediction of the fracture time for an applied stress of 20 can be computed as

$$t = -0.00013951(20)^4 + 0.010603(20)^3 - 0.21959(20)^2 + 0.036477(20) + 43.707 = 19.103$$

**20.53** Using linear regression gives




A prediction of $g$ at 55,000 m can be made as

$$g(55,000) = -2.015 \times 10^{-6}(55,000) + 9.80942 = 9.6986$$

Note that we can also use linear interpolation to give

$$g_1(55,000) = 9.698033$$

Quadratic interpolation yields

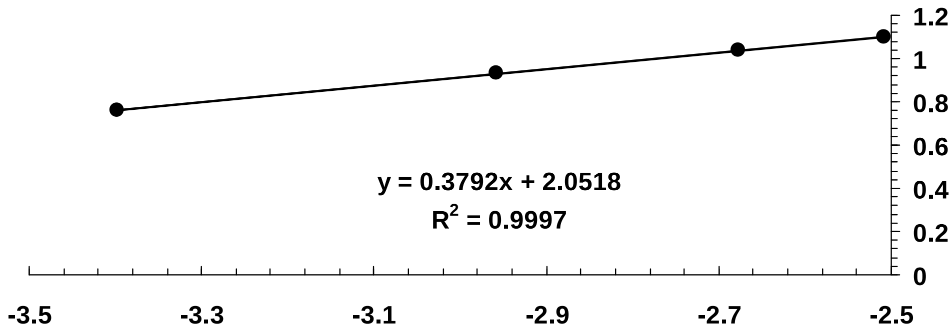
$g_2(55,000) = 9.697985$

Cubic interpolation gives

$g_3(55,000) = 9.69799$

Based on all these estimates, we can be confident that the result to 3 significant digits is 9.698.

**20.54** A plot of $\log_{10} \dot{\varepsilon}$ versus $\log_{10}\sigma$ can be developed as



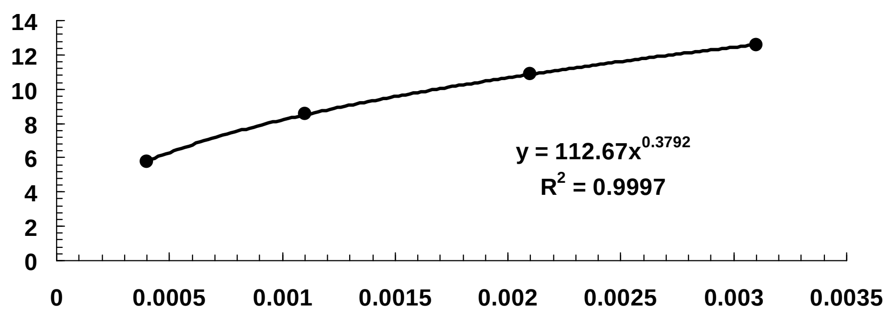

As shown, the best fit line is

$$\log_{10} \dot{\varepsilon} = 0.3792 \log_{10} \sigma + 2.0518$$

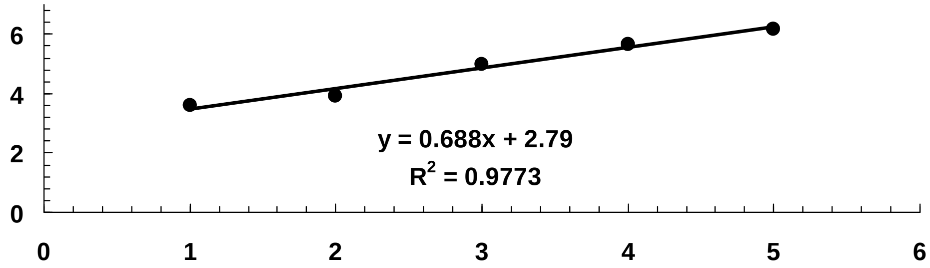
Therefore, $B = 10^{2.0518} = 112.67$ and $m = 0.3792$, and the power model is

$$y = 112.67 x^{0.3792}$$

The model and the data can be plotted as

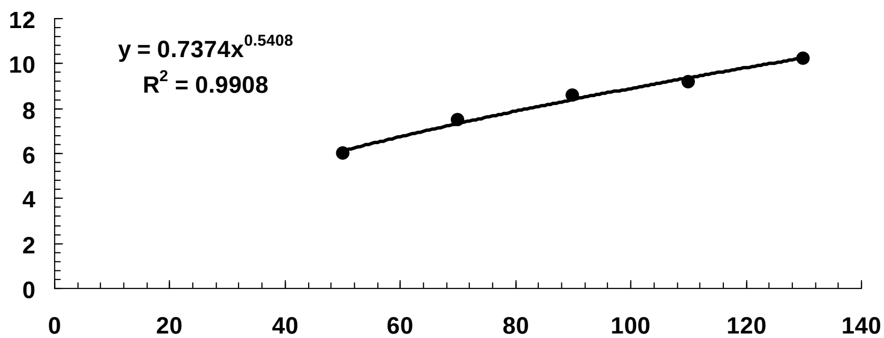



**20.55** Using linear regression gives



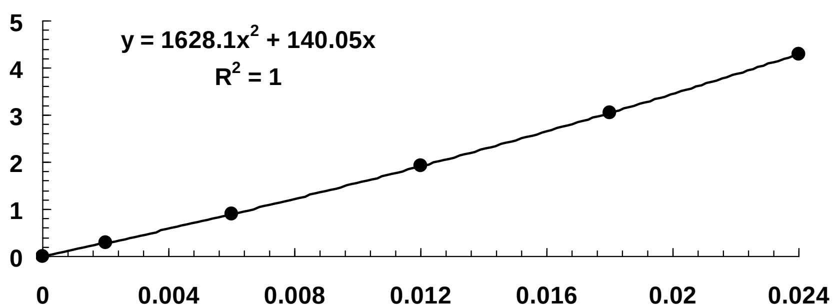

Therefore, $\tau_y = 2.79$ and $\mu = 0.688$.

**20.56** A power fit can be developed as




Therefore, $\mu = 0.7374$ and $n = 0.5408$.

**20.57** We fit a number of curves to this data and obtained the best fit with a second-order polynomial with zero intercept




Therefore, the best-fit curve is

$$u = 1628.1y^2 + 140.05y$$

We can differentiate this function

$$\frac{du}{dy} = 3256.2y + 140.05$$

Therefore, the derivative at the surface is 140.05 and the shear stress can be computed as $1.8\times10^{-5}(140.05) = 0.002521$ N/m$^2$.

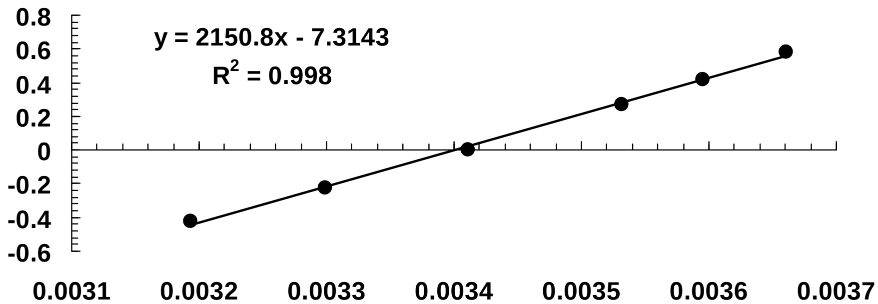
**20.58** We can use transformations to linearize the model as

$$\ln \mu = \ln D + B\frac{1}{T_a}$$

Thus, we can plot the natural log of $\mu$ versus $1/T_a$ and use linear regression to determine the parameters. Here is the data showing the transformations.

| $T$ | $\mu$ | $T_a$ | $1/T_a$ | $\ln \mu$ |
|---|---|---|---|---|
| 0 | 1.787 | 273.15 | 0.003661 | 0.580538 |
| 5 | 1.519 | 278.15 | 0.003595 | 0.418052 |
| 10 | 1.307 | 283.15 | 0.003532 | 0.267734 |
| 20 | 1.002 | 293.15 | 0.003411 | 0.001998 |
| 30 | 0.7975 | 303.15 | 0.003299 | -0.22627 |
| 40 | 0.6529 | 313.15 | 0.003193 | -0.42633 |

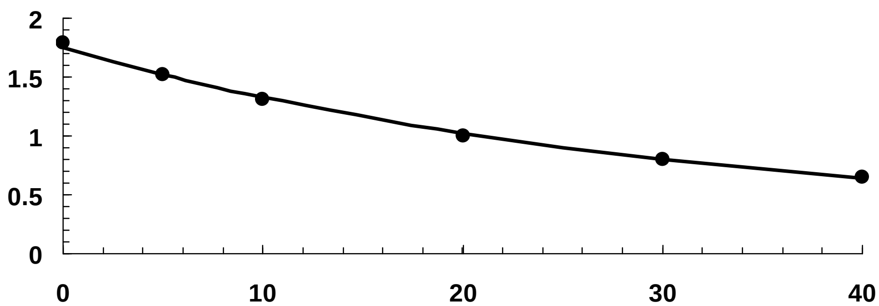
Here is the fit:




Thus, the parameters are estimated as $D = e^{-7.3143} = 6.65941\times10^{-4}$ and $B = 2150.8$, and the Andrade equation is

$$\mu = 6.65941\times10^{-4} e^{2150.8/Ta}$$

This equation can be plotted along with the data

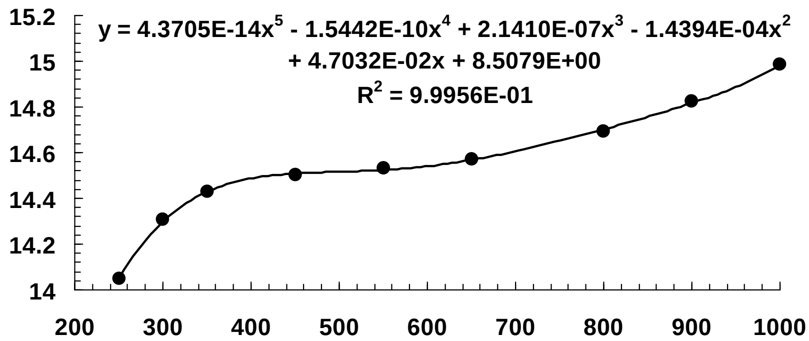



Note that this model can also be fit with nonlinear regression. If this is done, the result is

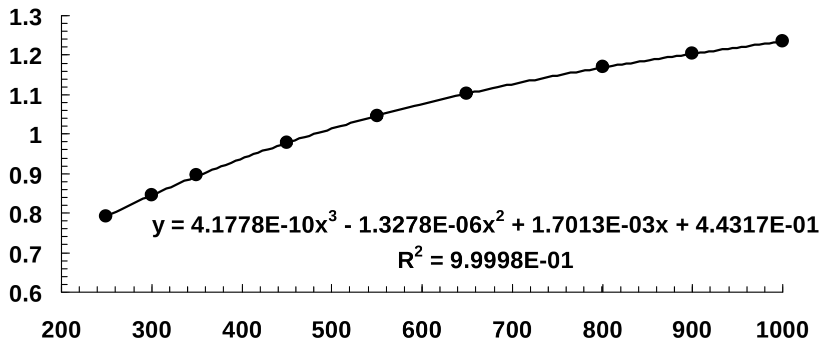
$$\mu = 5.39872\times10^{-4} e^{2210.66/Ta}$$

Although it is difficult to discern graphically, this fit is slightly superior ($r^2 = 0.99816$) to that obtained with the transformed model ($r^2 = 0.99757$).

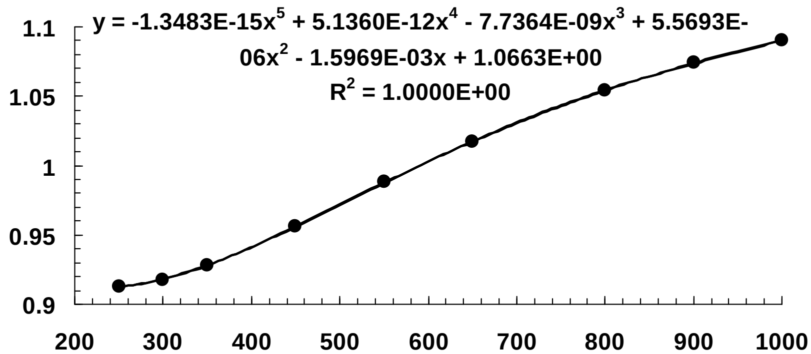
**20.59** Hydrogen: Fifth-order polynomial regression provides a good fit to the data,



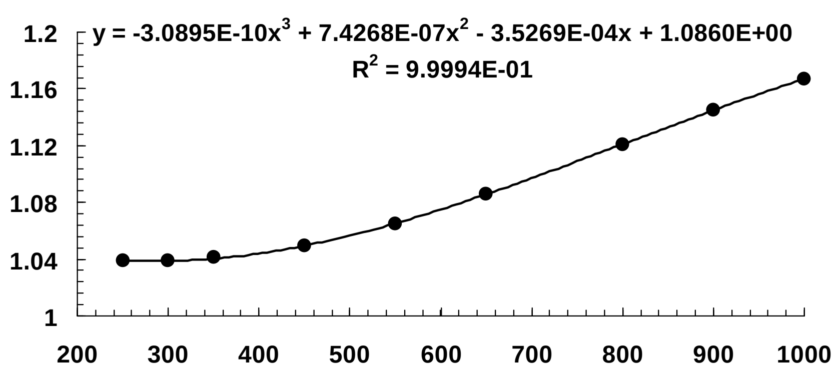

Carbon dioxide: Third-order polynomial regression provides a good fit to the data,




Oxygen: Fifth-order polynomial regression provides a good fit to the data,




Nitrogen: Third-order polynomial regression provides a good fit to the data,



**20.60** This part of the problem is well-suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible, for $x = 4$.

$y_0 = 4$ $f(y_0) = 38.43$
$y_1 = 2$ $f(y_1) = 53.5$
$y_2 = 6$ $f(y_2) = 30.39$
$y_3 = 0$ $f(y_3) = 80$
$y_4 = 8$ $f(y_4) = 30$

The results of applying Newton's polynomial at $y = 3.2$ are

| Order | *f*(*y*) | Error |
|---|---|---|
| 0 | 38.43 | 6.028 |
| 1 | 44.458 | -0.8436 |
| 2 | 43.6144 | -0.2464 |
| 3 | 43.368 | 0.112448 |
| 4 | 43.48045 | |

The minimum error occurs for the third-order version so we conclude that the interpolation is 43.368.

**(b)** This is an example of two-dimensional interpolation. One way to approach it is to use cubic interpolation along the *y* dimension for values at specific values of *x* that bracket the unknown. For example, we can utilize the following points at $x = 2$.

$y_0 = 0$ $f(y_0) = 90$
$y_1 = 2$ $f(y_1) = 64.49$
$y_2 = 4$ $f(y_2) = 48.9$
$y_3 = 6$ $f(y_3) = 38.78$

$T(x = 2, y = 2.7) = 58.13288438$

All the values can be tabulated as

$T(x = 2, y = 2.7) = 58.13288438$
$T(x = 4, y = 2.7) = 47.1505625$
$T(x = 6, y = 2.7) = 42.74770188$
$T(x = 8, y = 2.7) = 46.5$

These values can then be used to interpolate at $x = 4.3$ to yield

$T(x = 4.3, y = 2.7) = 46.03218664$

Note that some software packages allow you to perform such multi-dimensional interpolations very efficiently. For example, MATLAB has a function interp2 that provides numerous options for how the interpolation is implemented. Here is an example of how it can be implemented using linear interpolation,

```
>> Z=[100 90 80 70 60;
85 64.49 53.5 48.15 50;
70 48.9 38.43 35.03 40;
55 38.78 30.39 27.07 30;
40 35 30 25 20];
>> X=[0 2 4 6 8];
>> Y=[0 2 4 6 8];
>> T=interp2(X,Y,Z,4.3,2.7)
T =
```

```
   47.5254
```

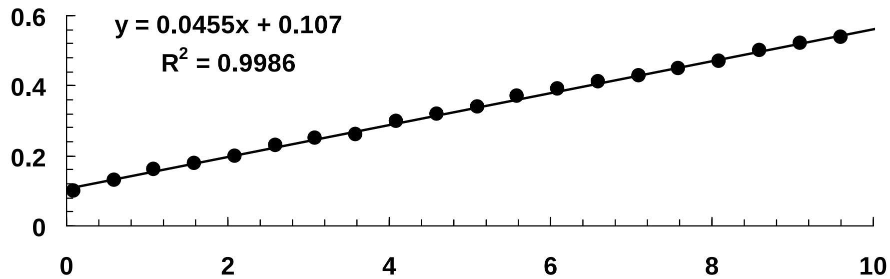
It can also perform the same interpolation but using bicubic interpolation,

```
>> T=interp2(X,Y,Z,4.3,2.7,'cubic')
T =
   46.0062
```

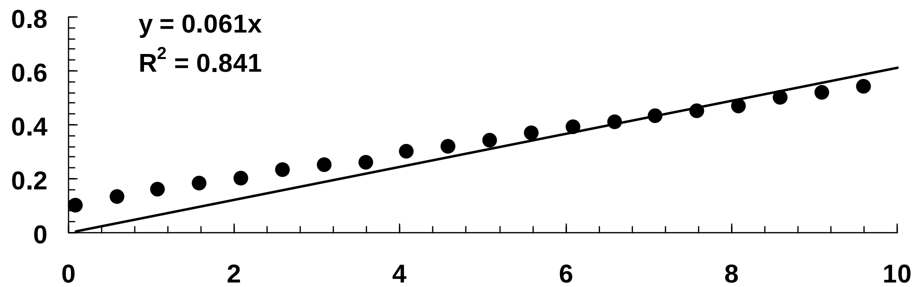
Finally, the interpolation can be implemented using splines,

```
>> T=interp2(X,Y,Z,4.3,2.7,'spline')
T =
   46.1507
```

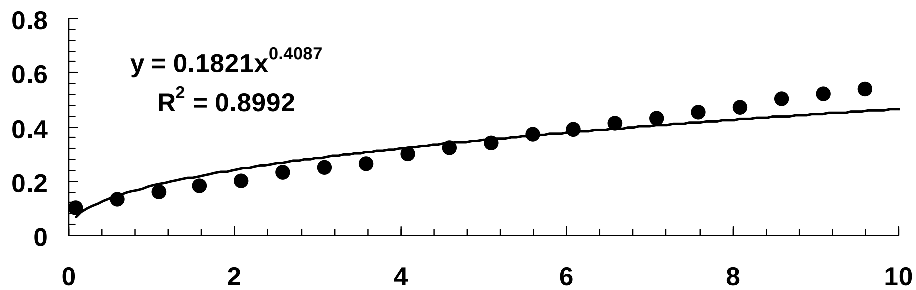
**20.61** This problem was solved using an Excel spreadsheet and TrendLine. Linear regression gives




Forcing a zero intercept yields




One alternative that would force a zero intercept is a power fit




However, this seems to represent a poor compromise since it misses the linear trend in the data. An alternative approach would to assume that the physically-unrealistic non-zero intercept is an artifact of the measurement method. Therefore, if the linear slope is valid, we might try $y = 0.0455x$.