**Bad Honnef School**

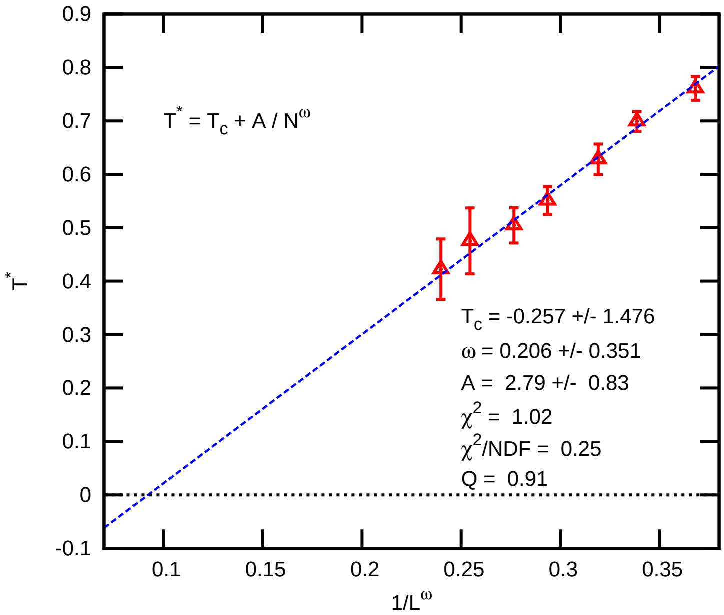
Peter Young

**Problem 4, Solution**

Least squares fit to a non-linear model.

a. Source code is given for gnuplot and python at the bottom. (Thanks to Matt Wittmann for doing the python scripts.)

A figure, produced by gnuplot, which includes all the fit parameters, is below. Note how the value of $Q$ is calculated, and how the fit parameters, including error bars, are displayed on the figure. As discussed in the handout, in the gnuplot script I divided the error bars by $\texttt{FIT\_STDFIT} \equiv \sqrt{\chi^2/N_{\rm DOF}}$.




The error bar for $\omega$ does not make sense because it includes negative values for which the fit blows up as $L \to \infty$. The error bar for $T_c$ also does not make sense because, on the positive side, it includes temperatures higher than any of the values for $T_L^\star$, which cannot be true because $T_L^\star$ *decreases* with increasing $L$.

b. I modified the gnuplot script to fix the value of $T_c$ (called $a$ in the script). The script is given below.

I ran the script for several values of $T_c$ and noted the value of $\chi^2$ in each case. I plotted the resulting value of $\chi^2$ against $T_c$ below. A common confidence limit is the region where $\Delta\chi^2 \le 1$, which corresponds to a 68% probability if the noise is Gaussian.

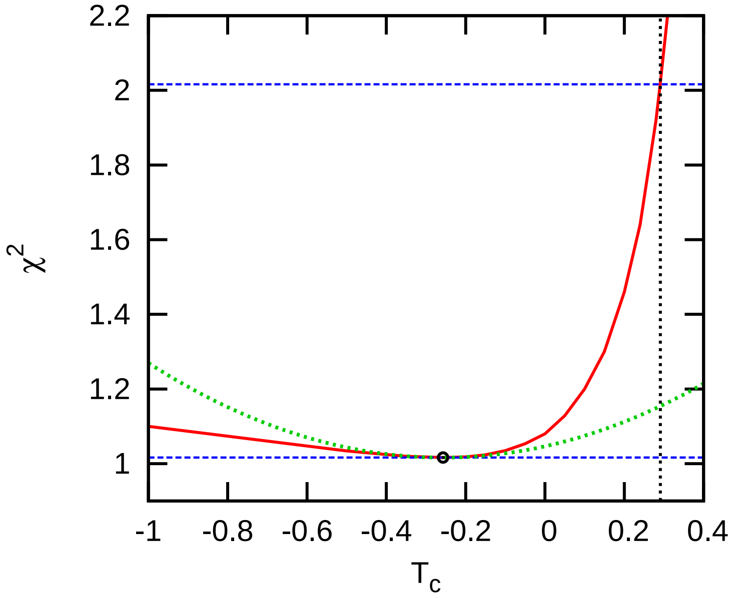
The curve is very asymmetric. On the negative side the range is *unbounded*. A fit with arbitrarily large negative $T_c$ still has $\Delta\chi^2 < 1$, and goes with a very small (probably unphysically small) value of the exponent $\omega$. This is an example of a strong *correlation* between values of different fit parameters.

On the positive side the confidence region extends up to $T_c = 0.29$, shown by the vertical (black) dotted line. Hence we can say that $T_c \le 0.29$, but can't give a lower bound.

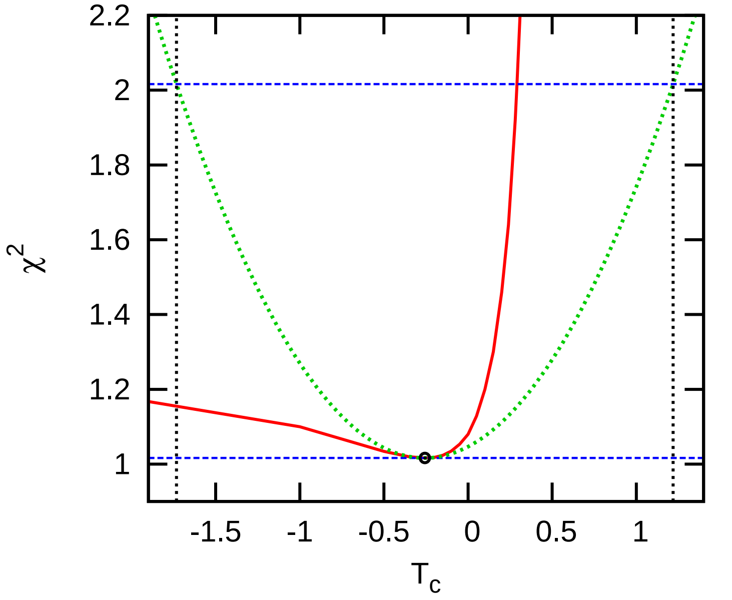
The dotted (green) line in the figure shows the (quadratic) variation of $\Delta\chi^2$ determined from the curvature at the minimum, i.e.

$$\Delta\chi^2 = \left(\delta T_c\right)^2 / \sigma_{T_c}^2 \, , \tag{1}$$

where $\sigma_{T_c}$ is the error bar quoted by the fitting program in part (a). Clearly, however, estimating $\Delta\chi^2$ from the curvature is inaccurate (recall this is *non-linear* model): considerably underestimating it on the positive side, and overestimating it on the negative side.




I also show below a global view, in which the end of the error bars determined from the fit, $\sigma_{T_c} = 1.476$, are shown by vertical (black) dotted lines. These are where $\Delta\chi^2 = 1$ if one computes $\Delta\chi^2$ according to Eq. (1) (the green, dotted line).




I also include a python script at the bottom, which gives the same results for $\chi^2$ as a function of $T_c$ as gnuplot.

```
==========================
Gnuplot Code for part (a):
==========================

set size 1.0, 0.6
set terminal postscript portrait enhanced
```

```
set output "HW4a.eps"
set fit errorvariables
f(x) = a + b / x**c
set xlabel "1/L^{/Symbol w}"
set ylabel "T^*"
set label "T^* = T_c + A / N^{/Symbol w}" at 0.1, 0.7
a = 0.3
b = 1
c = 0.2
fit f(x) "data.HW4" using 1:2:3 via a, b, c
set xrange [0.07:0.38]
g(x) = a + b * x
h(x) = 0 + 0 * x
ndf = FIT_NDF
chisq = FIT_STDFIT**2 * ndf
Q = 1 - igamma(0.5 * ndf, 0.5 * chisq)
set label sprintf("T_c = %5.3f +/- %5.3f",a, a_err/FIT_STDFIT) at 0.25, 0.33
set label sprintf("{/Symbol w} = %5.3f +/- %5.3f",c, c_err/FIT_STDFIT) at 0.25, 0.27
set label sprintf("A = %5.2f +/- %5.2f",b, b_err/FIT_STDFIT) at 0.25, 0.21
set label sprintf("{/Symbol c}^2 = %5.2f", chisq) at 0.25, 0.15
set label sprintf("{/Symbol c}^2/NDF = %5.2f", FIT_STDFIT**2) at 0.25, 0.09
set label sprintf("Q = %5.2f", Q) at 0.25, 0.03
plot "data.HW4" using (1/$1**c):2:3 with errorbars notitle ls 1, \
g(x) notitle ls 13, \
h(x) notitle lt 3 lw 4

==========================
Python Code for part (a):
==========================

from pylab import *
from scipy.optimize import leastsq
from scipy.stats import chi2

fname = sys.argv[1] if len(sys.argv) > 1 else 'data.txt'
L, Ts, Tserr = np.loadtxt(fname, unpack=True)
n = len(L)

def resids(p):
    Tc, w, A = p
    return (Tc + A/L**w - Ts) / Tserr

p0 = [-0.25, 0.2, 2.8]

p, covm = leastsq(resids, p0, full_output=True)[:2]
Tc, w, A = p
Tcerr, werr, Aerr = sqrt(diag(covm))
chisq = sum(resids(p)**2)
ndf = n - len(p)
Q = 1. - chi2.cdf(chisq, ndf)

print 'Tc = %10.4f +/- %7.4f' % (Tc, Tcerr)
print 'A  = %10.4f +/- %7.4f' % (A, Aerr)
print 'w  = %10.4f +/- %7.4f' % (w, werr)
```

```
print 'chi squared / NDF = %7.4lf' % (chisq / ndf)
print 'Q  = %10.4f' % Q


==========================
Gnuplot Code for part (b):
==========================

set size 1.0, 0.6
set terminal postscript portrait enhanced
set output "HW4b.eps"
set fit errorvariables
f(x) = a + b / x**c
set xlabel "1/L^{/Symbol w}"
set ylabel "T^*"
set label "T^* = T_c + A / N^{/Symbol w}" at 0.1, 0.7
a = 0.29
b = 1
c = 0.2
fit f(x) "data.HW4" using 1:2:3 via b, c
#set xrange [0.7:0.38]
g(x) = a + b * x
h(x) = 0 + 0 * x
ndf = FIT_NDF
chisq = FIT_STDFIT**2 * ndf
Q = 1 - igamma(0.5 * ndf, 0.5 * chisq)
set label sprintf("T_c = %5.3f",a) at 0.25, 0.33
set label sprintf("{/Symbol w} = %5.3f +/- %5.3f",c, c_err/FIT_STDFIT) at 0.25, 0.27
set label sprintf("A = %5.2f +/- %5.2f",b, b_err/FIT_STDFIT) at 0.25, 0.21
set label sprintf("{/Symbol c}^2 = %5.2f", chisq) at 0.25, 0.15
set label sprintf("{/Symbol c}^2/NDF = %5.2f", FIT_STDFIT**2) at 0.25, 0.09
set label sprintf("Q = %5.2f", Q) at 0.25, 0.03
plot "data.HW4" using (1/$1**c):2:3 with errorbars notitle ls 1, \
g(x) notitle ls 13, \
h(x) notitle lt 3 lw 4

==========================
Python Code for part (a):
==========================

from pylab import *
from scipy.optimize import leastsq
from scipy.stats import chi2

fname = sys.argv[1] if len(sys.argv) > 1 else 'data.txt'
L, Ts, Tserr = np.loadtxt(fname, unpack=True)
n = len(L)

Tc = 0.2

def resids(p):
    w, A = p
    return (Tc + A/L**w - Ts) / Tserr
```

```
p0 = [0.2, 2.8]

p, covm = leastsq(resids, p0, full_output=True)[:2]
w, A = p
werr, Aerr = sqrt(diag(covm))
chisq = sum(resids(p)**2)
ndf = n - len(p)
Q = 1. - chi2.cdf(chisq, ndf)

print 'A  = %10.4f +/- %7.4f' % (A, Aerr)
print 'w  = %10.4f +/- %7.4f' % (w, werr)
print 'chi squared / NDF = %7.4lf' % (chisq / ndf)
print 'Q  = %10.4f' % Q
```