Bad Honnef School

Peter Young

Problem 3, Solution

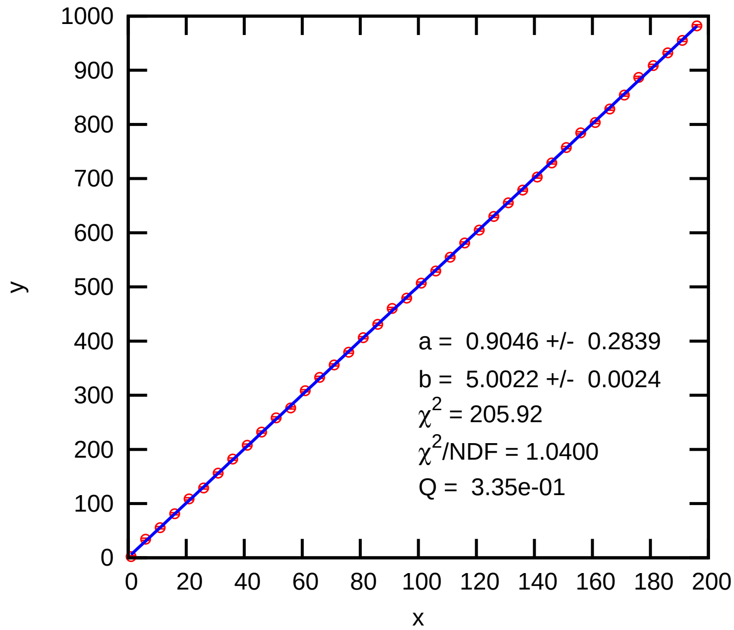
Least squares fit.

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

```
OUTPUT

slope     =      5.0022 +/-   0.0024
intercept =      0.9046 +/-   0.2839

chi squared / DOF =  1.0400

Note: The data actually corresponds to y = 1 + 5x (plus some random noise).
Hence the fit obtained the correct answer within the error bars.

The chi squared per degree of freedom is close to 1, as expected.
```

b. See the figure below which includes the fits.

The gnuplot script to do the fit (and procduce a figure) is at the bottom. Note how I present the fit parameters with error bars, the χ^2 and the goodness of fit parameter Q, *on the figure*. This is useful. All the fit parameters agree with what was found in part (a). Note too that I had to divide gnuplot's error bars by `FIT_STDFIT` $\equiv \sqrt{\chi^2/N_{\mathrm{DOF}}}$. The reason for this is discussed in the handout. The figure only shows every fifth point for clarity.



A python script, using fitting functions from the scipy package, is also shown at the bottom.

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

```
#!/usr/bin/perl

$n = 0;
while(<>)
{
    @line = split;
    $x[$n] = $line[0];
    $y[$n] = $line[1];
    $err[$n] = $line[2];
    $err2 = $err[$n]**2;
    $s += 1 / $err2;
    $sumx  += $x[$n] / $err2 ;
    $sumy  += $y[$n] / $err2 ;
    $sumxx += $x[$n]*$x[$n] / $err2 ;
    $sumxy += $x[$n]*$y[$n] / $err2 ;
    $n++;
}

$delta = $s * $sumxx - $sumx * $sumx ;
$c = ($sumy * $sumxx - $sumx * $sumxy) / $delta ;
$m = ($s  *  $sumxy - $sumx * $sumy) / $delta ;
$errm = sqrt($s / $delta) ;
$errc = sqrt($sumxx / $delta) ;

printf ("slope     = %10.4f +/- %7.4f \n", $m, $errm);
printf ("intercept = %10.4f +/- %7.4f \n\n", $c, $errc);

$NDF = $n - 2; # the no. of degrees of freedom is n-no. of fit params
$chisq = 0;
for ($i = 0; $i < $n; $i++)
{
    $chisq += (($y[$i] - $m*$x[$i] - $c)/$err[$i])**2;
}
$chisq /= $NDF;
printf ("chi squared / NDF = %7.4lf \n", $chisq);

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

#include "math.h";
#include "stdio.h";

main()
{
    double m, sumx, sumy, sumxx, sumxy, s, x[200], y[200], err[200], err2;
    double errc, errm, c, delta, chisq;
    int i, n;

    s = 0; sumx=0; sumy = 0; sumxx = 0; sumxy = 0;
    n = 200;

    for (i = 0; i < n; i++)
    {
```

```
        scanf("%lf %lf %lf", &x[i], &y[i], &err[i]);
        err2 = err[i]*err[i] ;
        s     += 1 / err2 ;
        sumx  += x[i] / err2 ;
        sumy  += y[i] / err2 ;
        sumxx += x[i]*x[i] / err2 ;
        sumxy += x[i]*y[i] / err2 ;
    }

    delta = s * sumxx - sumx * sumx ;
    c = (sumy * sumxx - sumx * sumxy) / delta ;
    m = (s  *  sumxy - sumx * sumy) / delta ;
    errm = sqrt(s / delta) ;
    errc = sqrt(sumxx / delta) ;

    printf ("slope     = %10.4f +/- %7.4f \n", m, errm);
    printf ("intercept = %10.4f +/- %7.4f \n\n", c, errc);

    chisq = 0;
    for (i = 0; i < n; i++)
    {
        chisq += pow((y[i] - m*x[i] - c)/err[i], 2);
    }
    chisq /= (n - 2); // the no. of degrees of freedom is n - no. of fit params
    printf ("chi squared / DOF = %7.4lf \n", chisq);

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

import fileinput
from math import *

x = []
y = []
err = []
s = sumx = sumy = sumxx = sumxy = 0.

for line in fileinput.input():
    line = line.split()
    x_i = float(line[0]);   x.append(x_i)
    y_i = float(line[1]);   y.append(y_i)
    err_i = float(line[2]); err.append(err_i)
    err2 = err_i**2
    s += 1 / err2
    sumx    += x_i / err2
    sumy    += y_i / err2
    sumxx   += x_i*x_i / err2
    sumxy   += x_i*y_i / err2

n = len(x)
delta = s * sumxx - sumx * sumx
c = (sumy * sumxx - sumx * sumxy) / delta
```

```
m = (s * sumxy - sumx * sumy) / delta
errm = sqrt(s / delta)
errc = sqrt(sumxx / delta)

print "slope     = %10.4f +/- %7.4f " % (m, errm)
print "intercept = %10.4f +/- %7.4f \n" % (c, errc)

NDF = n - 2
chisq = 0.

for i in xrange(n):
    chisq += ((y[i] - m*x[i] - c)/err[i])**2;

chisq /= NDF
print "chi squared / NDF = %7.4lf " % chisq

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

set size 1.0, 0.6
set terminal postscript portrait enhanced font 'Helvetica,16'
set output "HW3b.eps"
set fit errorvariables
f(x) = a + b * x
fit f(x) "data.HW3" using 1:2:3 via a, b
set xlabel "x"
set ylabel "y"
ndf = FIT_NDF
chisq = FIT_STDFIT**2 * ndf
Q = 1 - igamma(0.5 * ndf, 0.5 * chisq)
set label sprintf("a = %7.4f +/- %7.4f", a, a_err/FIT_STDFIT) at 100, 400
set label sprintf("b = %7.4f +/- %7.4f", b, b_err/FIT_STDFIT) at 100, 330
set label sprintf("{/Symbol c}^2 = %6.2f", chisq) at 100, 270
set label sprintf("{/Symbol c}^2/NDF = %6.4f", FIT_STDFIT**2) at 100, 200
set label sprintf("Q = %9.2e", Q) at 100, 130
plot "data.HW3" using 1:2:3 every 5 with errorbars notitle pt 6 lc rgb "red" lw 2, \
f(x) notitle lc rgb "blue" lw 4 lt 1

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

from pylab import *
from scipy.optimize import leastsq

fname = sys.argv[1] if len(sys.argv) > 1 else 'data.txt'
x, y, err = np.loadtxt(fname, unpack=True)
n = len(x)

resids = lambda p: (p[0] + p[1]*x - y) / err
p0 = [5., 0.1]  # initial values of parameters

p, covm = leastsq(resids, p0, full_output=True)[:2]
```

```
c, m = p
errc, errm = sqrt(diag(covm))
chisq = sum(resids(p)**2)
chisq /= n - 2

print "slope     = %10.4f +/- %7.4f " % (m, errm)
print "intercept = %10.4f +/- %7.4f \n" % (c, errc)
print "chi squared / NDF = %7.4lf " % chisq
```