LeastSquares01.wxmx

TABLE OF CONTENTS

1 Preface

In LeastSquares01.wxmx we concentrate on using least squares methods to fit data to a model. Our examples include both linear and nonlinear models. We emphasize the calculation of error estimates for derived parameters and using the value of X^2, where appropriate, to validate the trustworthiness of any particular fit. We discuss both ordinary least squares (OLS) and weighted least squares (WLS).

Edwin L. (Ted) Woollett
https://home.csulb.edu/~woollett/
Nov. 27, 2024

2 References

R. J. Barlow, Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences (Manchester Physics Series), 1993, Wiley

[HH] Ifan Hughes, Thomas Hase, Uncertainties: A Practical guide to Modern Error Analysis, 2010, Oxford Univ Press.

Fred Senese, Symbolic Mathematics for Chemists, A Guide for Maxima Users, Wiley, 2019

Luca Lista, 'Statistical Methods for Data Analysis in Particle Physics', Lecture Notes in Physics 909, 2016, Springer-Verlag,

Chi Square Distribution (χ2) and Least Squares Fitting
K. K. Gan
https://www.asc.ohio-state.edu/gan.1/teaching/spring04/Chapter6.pdf

Fitting Experimental Data to Straight Lines (Including Error Analysis)
by Annette D. Shine, Chemical Engineering, Univ. Delaware, August 2006
https://cbe.udel.edu/wp-content/uploads/2019/03/FittingData.pdf

Lecture 12, Fitting a Straight Line to Data Part 2: Uncertainties in Both Parameters
ASTR 288A, Practical Astrostatistics
Cole Miller, Dept. of Astronomy, Univ. Maryland at College Park
https://www.astro.umd.edu/~miller/teaching/astr288a/lecture12.pdf

Gerhard Bohm, Günter Zech,
Introduction to Statistics and Data Analysis for Physicists, 3rd revised edition, 2017,
available from CERN webpage:
https://s3.cern.ch/inspire-prod-files-d/da9d786a06bf64d703e5c6665929ca01

Frederick James, 'Statistical Methods in Experimental Physics', 2nd ed., 2006, World Scientific.

Louis Lyons, Statistics for Nuclear and Particle Physics, 1986, Cambridge Univ. Press,

Glen Cowan , Statistical Data Analysis, Clarendon Press, Oxford, 1998,

```
(%i5) /*  load(draw)$
set_draw_defaults(line_width=2,   point_type = filled_circle,
        head_type = 'nofilled, head_angle = 20, head_length = 0.5,
        background_color = light_gray, draw_realpart=false)$   */

load (descriptive)$  load (distrib)$
load (lsquares)$   fpprintprec : 5$ ratprint : false$
```

Homemade functions fll, head, tail, Lsum are useful for looking at long lists and summing the elements.

```
(%i9) fll ( aL) := [ first (aL), last (aL), length (aL) ]$
      head(L) := if listp (L) then rest (L, - (length (L) - 3) ) else
           error("Input to 'head' must be a list of expressions ")$
      tail (L) := if listp (L) then rest (L, length (L) - 3 )  else
            error("Input to 'tail' must be a list of expressions ")$
      Lsum (aList) := apply ("+", aList)$
```

3 *Guidelines for plotting data*

Quoting [HH], Ch. 5,

"The graphical representation of data is the most efficient method of reporting experimental measurements in the physical sciences. Graphs are a very effective visual aid, and we make use of them to (i) highlight trends in, and relationships among, experimental data, (ii) test theories, (iii) enable comparisons between data sets to be made, (iv) look for evidence of systematic errors and (v) extract additional parameters which characterise the data set."

Quoting [HH], Sec. 5.1:

"(1) Plot the independent variable on the horizontal axis, and the dependent variable on the vertical axis.

(2) Consider linearising the data to generate a straight-line plot.

(3) Use appropriate scales for the axes such that most of the area of the graph is utilised.

(4) Label each axis with the name and units of the variable being plotted.

(5) Add data points and error bars, ensuring that they are clear, with different data sets being distinguishable.

(6) Add a fit or trend line—either a straight line, a smooth curve to capture the trend of the data set, or a suitable theoretical model."

3.1 Linearising the data

Quoting [HH], Sec. 5.1:

"For clarity of the graphical representation of data, one should always attempt to show a linear relationship between the dependent and independent variables."

"This is because it is much easier to (i) see deviations from a straight line, (ii) fit linear relations and express the relationship between the experimental quantities and those predicted by theory."

"In particular there exist analytic expressions for the slope and intercept and their uncertainties for a straight-line fit. For example, it is known that the period, T, of a simple pendulum depends on the length L via the relation T = 2π √ (L/g), where g is the acceleration due to gravity. Typically one sets the length of the pendulum (the independent variable) and makes multiple measurements of the period (the dependent variable) to give T ± σT . Thus by plotting T^2 on the y-axis versus L on the x-axis we should get a straight line through the origin, and can extract a value for g and its error from the slope. Note that the conventional terminology is to call this ‘a graph of T^2 against L’. "

3.1.1 Nonlinear Models Transformed

Following Fred Senese, p. 300, here are some models which can be linearized.

```
(%i10) matrix ( ["Model", "Transformation", "Linear Fit Eqn"],
                ["y = exp ( m x  + c)", "Y = ln y", "Y  = m x + c"],
                ["y = (m x + c)^n", "Y = y^(1/n)",  "Y = m x + c"],
                ["y = 1/(m x + c)", "Y = 1/y", "Y = m x + c"],
                [" y = m ln x + c",  "X = ln x",  "y = m X + c"],
                [" y = b x^m ", "Y = ln y, X = ln x, C = ln b", "Y = m X + C"] );
```

(%o10)

Model	Transformation	Linear Fit Eqn
y = exp (m x + c)	Y = ln y	Y = m x + c
y = (m x + c)^n	Y = y^(1/n)	Y = m x + c
y = 1/(m x + c)	Y = 1/y	Y = m x + c
y = m ln x + c	X = ln x	y = m X + c
y = b x^m	Y = ln y, X = ln x, C = ln b	Y = m X + C

3.2 Error Bars

"The coordinates of a data point on a graph are our best estimate (i.e. the mean) of the independent and dependent variables. There is an uncertainty associated with these mean values and a graph needs to reflect this. Error bars are added to points on a graph to indicate the 68% confidence limits."

In Uncertainties02B.wxmx we discussed the calculation of the uncertainty of the mean σE, and we quote the best estimate of the mean of x as <x> +/- σx, in which σx is the uncertainty of the mean for the quantity x (the 68% confidence interval).

In Uncertainties03.wxmx we discussed the progation of errors when a final value is a function of various measured quantities, each with their own uncertainty of the mean.

"In general," when plotting y = f(x), "the fractional errors on the dependent variable y should be larger than those of the independent variable x."

3.3 Maxima Error Bars

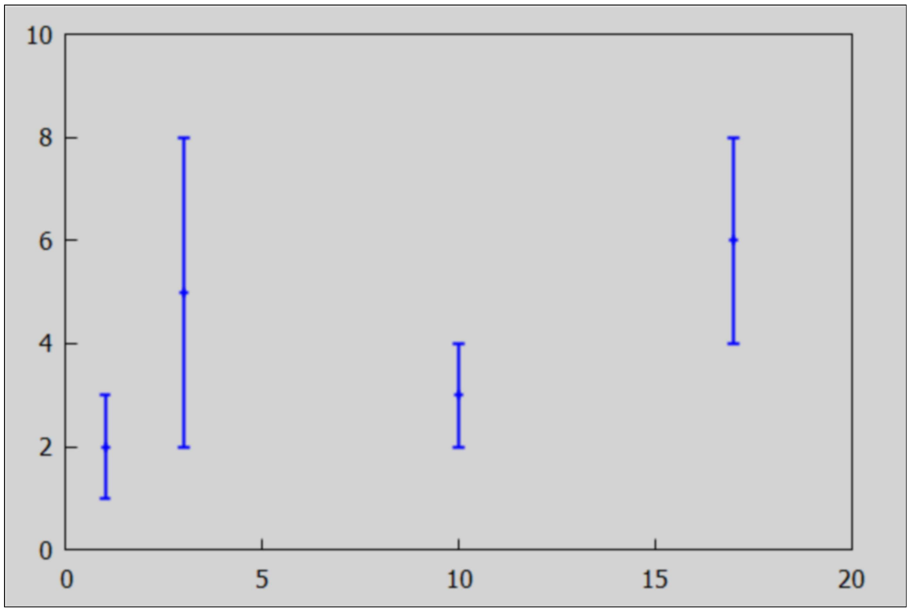
We can use the errors function in the draw2d package for error bars. The argument to the errors function is a list in which there is one sublist for each data point.

The general idea if we want both x and y error bars is the following. If the error bars are symmetric, each sublist should have the form [x, y, σx, σy].
If the error bars are not symmetric, each sublist should have the form:
 [x, y, xlo, xhi, ylo, yhi]. The results are sensitive to the value of line_width.

Here is an example from the Maxima manual of vertical error bars:

```
(%i11) wxdraw2d( xrange = [0,20], yrange = [0,10],
           error_type = 'y, background_color = light_gray, line_width = 2,
           errors([[1,2,1], [3,5,3], [10,3,1], [17,6,2]]))$
```

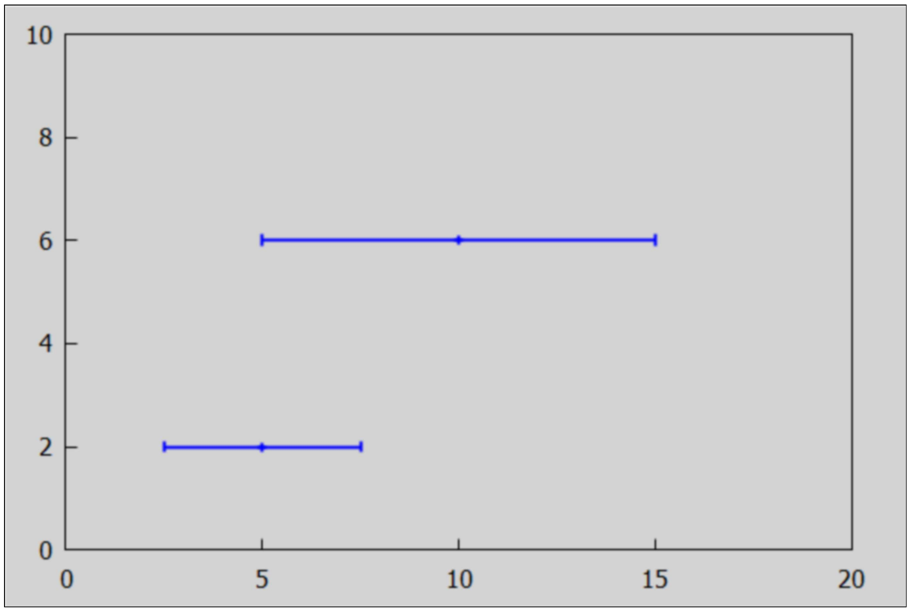
(%t11)



Here is an example of horizontal error bars:

```
(%i12) wxdraw2d( xrange = [0,20], yrange = [0,10],
        error_type = 'x, background_color = light_gray, line_width = 2,
        errors( [ [5,2,2.5], [10,6,5] ] ))$
```

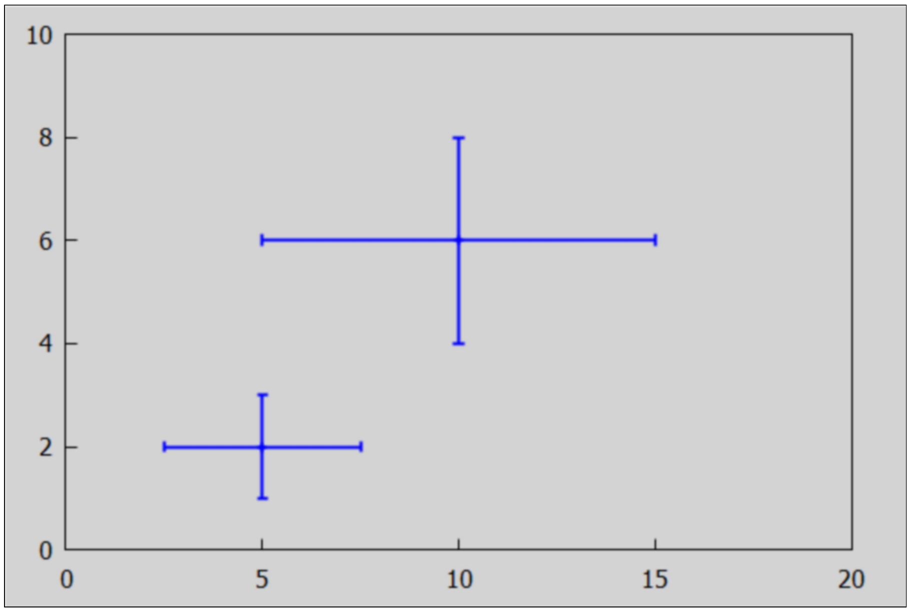
(%t12)



Here is an example of both x and y error bars.

```
(%i13) wxdraw2d( xrange = [0,20], yrange = [0,10],
        error_type = 'xy, background_color = light_gray, line_width = 2,
        errors( [ [5, 2, 2.5, 1], [10, 6, 5, 2] ] ))$
```

(%t13)



3.4 Two or More Dependent Variables

"There are circumstances in which two dependent variables are measured for each value of the independent variable. In such a situation it is possible to draw a graph with two y-axes, provided a sufficient level of clarity is maintained."

"It is crucial to indicate clearly which data set is associated with which axis. A smooth curve can be added to emphasise the trend in the data where there is no particular theoretical model, with an explanation in the caption that the curve is a guide to the eye."

3.5 Adding a fit or trend line

"There are three types of fit, or trend line, which we can add to a graph.

(1) If a visual inspection of the graph confirms a linear dependence, we add a linear trend line.

(2) There are occasions when the theoretical model cannot be linearised. For such a case the appropriate nonlinear function can be added to the graph. The relevant parameters are initially chosen, by eye, to capture the trend of the data.

(3) In circumstances where there is no theoretical prediction of the relationship between the dependent and independent variable, one can add a smooth curve to guide the eye. The caption should clearly state that the smooth curve is a guide to the eye, and not a theoretical prediction."

"Experiments are often designed such that the dependent variable exhibits a linear dependence on the independent variable. There are simple analytic results for calculating the best-fit straight line for a data set with constant error bars on the vertical axis (we will discuss in more detail in the next section ... what exactly we mean by 'best fit')."
As previously mentioned, "the error bars on the y-axis of a graph represent the 68% confidence limit for the value of the ordinate. If we assume that the best-fit straight line is a good description of the data, we would therefore expect that the line intersects two-thirds of the measurements within their error bar."
In other words, "for a good fit we expect two-thirds of the data points to be within one standard error bar of the trend line."
"There are two obvious reasons why the fraction of points which are consistent with the line of best fit may be significantly different from this: (i) the error bars have been overestimated; and (ii) the assumed linearity of the best fit is not valid."

"If approximately two-thirds of the data points intersect the linear trend line, the data are well modelled by a straight line

$$y = m\,x + c,$$

and it is appropriate to extract four important parameters—these are the gradient, m, and intercept, c, and their associated errors.......the correlation coefficient should also be reported for a straight-line fit."
"Using a process known as the method of least squares ... one can derive analytic expressions for the gradient m and its uncertainty and the intercept c and its uncertainty."

4 Linear Correlation Coefficient r

Define the average of the set of x_i's:
<x> = (1/N)* Σx_i
and the average of the corresponding y_i's:
<y> = (1/N)* Σy_i,
likewise define
<xy> = (1/N)* Σ(x_i * y_i),
<x²> = 1/N)* Σ(x_i)²,
<y²> = 1/N)* Σ(y_i)².

Then the linear correlation coefficient r can be calculated with the formula

r = (<xy> - <x>*<y>) / sqrt ((<x^2> - <x>^2)*(<y^2> - <y>^2)).

We use the Maxima list behavior shown here:

```
(%i14) [a, b]*[A, B];
(%o14) [A a,B b]
```

The function Lsum (alist) is defined at the top of this worksheet.

```
(%i15) Lsum ([a, b, c]);
(%o15) c+b+a
```

```
(%i18) X : [3, 4, 9, 10, 15, 15];
       Y : [8, 9, 12, 11, 14, 17];
       r : (6*Lsum(X*Y) - Lsum(X) * Lsum(Y))/
          ( sqrt( 6*Lsum(X^2) - (Lsum(X))^2)*sqrt (6*Lsum(Y^2) - (Lsum(Y))^2)), numer;
(X)    [3,4,9,10,15,15]
(Y)    [8,9,12,11,14,17]
(r)    0.93952
```

So r is positive and approximately equal to 0.940, a strong linear correlation between the variables x and y.

4.1 LinCorrCoef (data-list)

We can turn this into a small Maxima function.

```
(%i19) LinCorrCoef (dataL) :=
        block ([XX, YY, nn],
          nn : length (dataL),
          XX : makelist (dataL[j][1], j, 1, nn),
          YY : makelist (dataL[j][2], j, 1, nn),
          float ( (nn*Lsum(XX*YY) - Lsum (XX)*Lsum (YY) )/
            sqrt (nn*Lsum(XX^2) - (Lsum(X))^2) /
             sqrt (nn*Lsum (YY^2) - (Lsum (Y))^2) ) )$
```

To test this function, we need to create a list of pairs (x,y).

```
(%i20) mydata : makelist ([X[j], Y[j] ], j, 1, length (X) );
(mydata) [[3,8],[4,9],[9,12],[10,11],[15,14],[15,17]]
```

```
(%i21) r : LinCorrCoef ( mydata );
(r)    0.93952
```

4.2 Coefficient of Determination r²

The coefficient of determination is calculated by first finding r, the linear correlation coefficient, and then squaring r.

```
(%i22) r^2;
(%o22) 0.88269
```

The coefficient of determination for this data set is about 0.883, which means 88.3% of the variation in y is explained by the variation in x.

100% - 88.3% = 11.7% of the variation in y is unexplained by the variation in x.

Quoting Fred Senese, p. 267,

"When doing a linear fit to a set of data with the model y = m*x + c, r and r^2 are NOT measures of how good the fit is; they only indicate the strength of the linear correlation between x and y."

"r^2 is zero when the data is completely random. r^2 can be equal to one only when all the residuals e_i = y_i - ym_i = y_i - m*x_i - c are zero, so all the data points lie exactly on the best fit line."

5 *lsquares_estimates (dataM, varL, eqn, paramL, [options])*

Maxima's lsquares package has several functions which which find solutions both symbolic and numerical: look in the manual under lsquares_estimates (which we demonstrate below), lsquares_residuals, lsquares_residual_mse (proportional to mean square error of fit), and lsquares_estimates_approximate. We can use these functions for ordinary least squares problems, but they are even more useful for nonlinear fitting problems.

You need to use load (lsquares)$ before using the functions in the package. We have done that at the top of this worksheet.

5.1 Linear Least Squares using lsquares_estimates

lsquares_estimates (dataM, varL, eqn, paramL, [options]) needs a matrix for the first argument, with each row giving a data point (x,y). We start with a data list and convert it to a matrix, although normally you would just use matrix ([0,5.9], [0.9, 5.4], etc, etc).

We will find the linear least squares fit best values of m and c, assuming the model y = m*x + c, for a dummy set of (x,y) data, using the list given:

```
(%i23) given : [ [0,5.9], [0.9,5.4],[1.8,4.4], [2.6, 4.6], [3.3,3.5], [4.4,3.7], [5.2,2.8], [6.1,2.8],
         [6.5, 2.4], [7.4, 1.5] ];
(given) [[0,5.9],[0.9,5.4],[1.8,4.4],[2.6,4.6],[3.3,3.5],[4.4,3.7],[5.2,
        2.8],[6.1,2.8],[6.5,2.4],[7.4,1.5]]

(%i24) M : apply ('matrix, given);
```

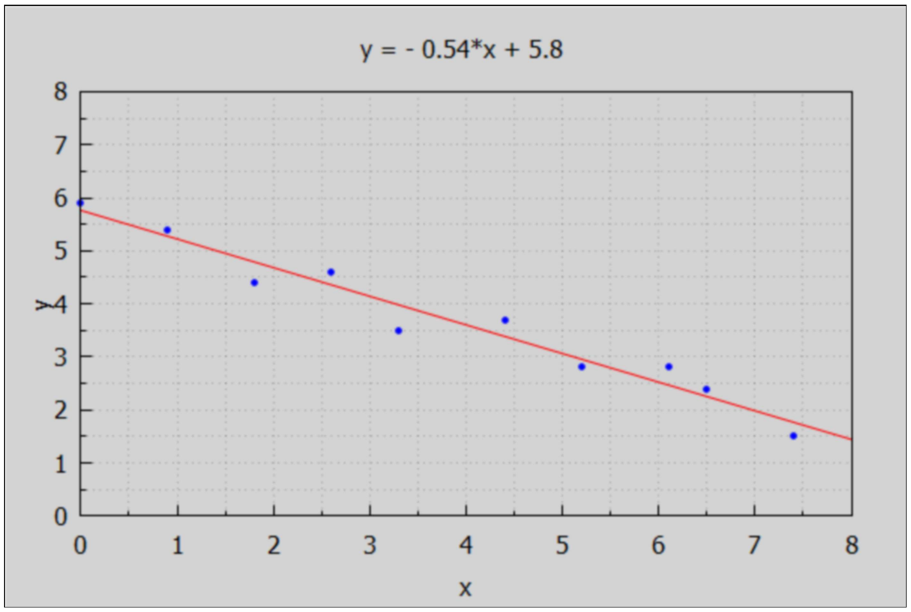
$$(M)\quad \begin{pmatrix} 0 & 5.9 \\ 0.9 & 5.4 \\ 1.8 & 4.4 \\ 2.6 & 4.6 \\ 3.3 & 3.5 \\ 4.4 & 3.7 \\ 5.2 & 2.8 \\ 6.1 & 2.8 \\ 6.5 & 2.4 \\ 7.4 & 1.5 \end{pmatrix}$$

```
(%i26) length (given);
       length (M);
(%o25) 10
(%o26) 10
```

Here the variable list has elements [x,y], with the independent variable first, (x is the label of the first column of the data matrix, y is the label of the second column), the model equation to fit the data to is y = m*x+c, and the parameter list is [m, c]. This function returns a list of possible answers for the values of the parameters m and c (in general), but here there is only one possibility so the returned list has length 1 and we need to use return-list[1] to get a list like [m = something, c = something-else].

```
(%i27) lsquares_estimates (M, [x, y], y = m*x + c, [m, c]), numer;
(%o27) [[m=−0.53958,c=5.7612]]

(%i28) solns : %[1];
(solns) [m=−0.53958,c=5.7612]

(%i29) [m, c] : map ('rhs, solns);
(%o29) [−0.53958,5.7612]

(%i31) m;
       c;
(%o30) −0.53958
(%o31) 5.7612

(%i32) wxdraw2d ( xrange = [0, 8], yrange = [0, 8], xlabel = "x",
          ylabel = "y",   title = "y = - 0.54*x + 5.8 ", point_type = filled_circle,
        background_color = light_gray, line_width = 1, grid = [2,2], point_size = 0.6,
         points (given), color = red, explicit (m*x + c, x, 0, 8) )$
```

(%t32)



5.2 Using linear_regression (M)

The linear_regression function can handle more than one independent variable, for example y_i = c + m1*x1_i + m2*x2_i.

```
(%i33) load (stats);
(%o33) C:/maxima−5.43.2/share/maxima/5.43.2/share/stats/stats.mac
```

```
(%i34) result : linear_regression (M);
```

$$\text{(result)}\quad \left| \begin{array}{c} \mathit{LINEAR\ REGRESSION\ MODEL} \\ b_{estimation}=[5.7612,-0.53958] \\ b_{statistics}=[30.404,-12.808] \\ b\_p\_values=[1.4868\ 10^{-9},1.3025\ 10^{-6}] \\ b_{distribution}=[student_t,8] \\ v_{estimation}=0.10008 \\ v\_conf\_int=[0.045662,0.36732] \\ v_{distribution}=[chi2,8] \\ adc=0.94769 \end{array} \right.$$

```
(%i35) [cl, ml] : take_inference ('b_estimation, result);
(%o35) [5.7612,−0.53958]
```

The analytic formulas we derive below give c = 5.7612, m = -0.53958 as well.

```
(%i36) items_inference(result);
```

$$\text{(\%o36)}\quad [b_{estimation},b_{covariances},b\_conf\_int,b_{statistics},b\_p\_values,b_{distribution},v_{estimation},v\_conf\_int,v_{distribution},residuals,adc,aic,bic]$$

The 95% confidence intervals for the parameters c and m require the use of the syntax take_inference ('b_conf_int, result):

```
(%i37) [cl_int, ml_int] : take_inference ('b_conf_int, result);
(%o37) [[5.3242,6.1981],[−0.63672,−0.44243]]
```

By default (which you can override), linear_regression computes the 95% confidence intervals for each of the parameters, so (high - low) = 2*(1.96)*σ, or
σ = (high - low) / 3.92.

```
(%i39) σcl : (cl_int [2] - cl_int[1])/3.92;
       σml : (ml_int[2] - ml_int[1])/3.92;
(σcl)  0.22294
(σml)  0.049563
```

The more accurate analytic formulas used by OLS2 produce σc = 0.189 ~ 0.19, and σm = 0.042.

If we use the option to assume 68.27% confidence intervals (1 σ intervals),

```
(%i40) result : linear_regression (M, 'conflevel = 0.6827);
```

$$(result)\quad \left| \begin{array}{c} \mathit{LINEAR\ REGRESSION\ MODEL} \\ b_{estimation}=[5.7612,-0.53958] \\ b_{statistics}=[30.404,-12.808] \\ b\_p\_values=[1.4868\ 10^{-9},1.3025\ 10^{-6}] \\ b_{distribution}=[student_t,8] \\ v_{estimation}=0.10008 \\ v\_conf\_int=[0.067644,0.19195] \\ v_{distribution}=[chi2,8] \\ adc=0.94769 \end{array} \right.$$

```
(%i41) [cl_int, ml_int] : take_inference ('b_conf_int, result);
(%o41) [[5.5591,5.9633],[−0.58451,−0.49465]]

(%i43) σcl : (cl_int [2] - cl_int[1])/2;
       σml : (ml_int[2] - ml_int[1])/2;
(σcl)  0.2021
(σml)  0.04493
```

which is about the same results for the parameter errors.

5.3 Residual Analysis

5.3.1 R^2 = sum of squares of residuals

Given a set of (x,y) measurement data, let x be the independent variable and y be the dependent variable. A model of the data ymod (x) = f(x, parameters) is considered and the question is what values of the free parameters give the most faithful picture of the raw data. One takes the difference between the value y_i corresponding to x_i from measurements and subtracts the model prediction f(x_i, parameters) (this difference is called the 'residual'), forms the sum of the squares of the residuals, and finds the values of the parameters which will lead to the smallest sum.

We call the sum of the squares of the residuals R^2:

$$R^2 = \Sigma (y\_i - f(x\_i, parameters)^2$$

in which the sum over index i includes all the data points.

5.3.2 Plot of Residuals

Quoting Fred Senese, p. 268:
"The residuals are a rich source of information about the quality of a fit. They can reveal outliers in the data and missing or incorrectly specified variables in the model. With precise and accurate data, they can also tell us something about the model's accuracy."

"Always present least-squares results with scatter plots of the data and the residuals. Routinely examine the following plots and include any that are particularly informative when reporting the fit."

1.) Plot the raw data y_i vs x_i (or against each x variable if there are multiple x variables.
2.) Plot residuals against time or run number.
3.) Plot residuals against ym_i (not y_i), particularly when there is more than one x variable.
4.) Plot residuals against each x variable.

For each data point (x_i, y_i), and given linear least squares fit values of slope m and y-intercept c, our model prediction is ym_i = m*x_i + c. The corresponding residual for this data point is defined as

e_i = y_i - ym_i = y_i - m*x_i - c.

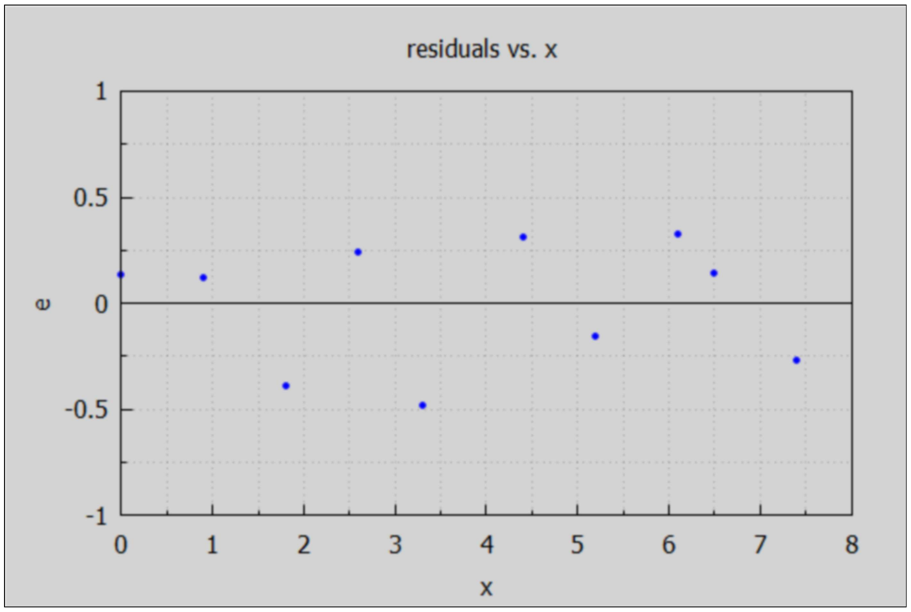
If X is a list of the x_i values and Y is a list of the corresponding y_i data values, then the list Y - m*X - c is a list of the corresponding residuals.

```
(%i48) X : makelist (given[j][1], j, 1, length (given) );
       Y : makelist (given[j][2], j, 1, length (given) );
       Ym : m*X + c;
       E : Y - Ym;
       Rsq : Lsum (E^2);
(X)    [0,0.9,1.8,2.6,3.3,4.4,5.2,6.1,6.5,7.4]
(Y)    [5.9,5.4,4.4,4.6,3.5,3.7,2.8,2.8,2.4,1.5]
(Ym)   [5.7612,5.2756,4.7899,4.3583,3.9806,3.387,2.9554,2.4698,2.2539
       ,1.7683]
(E)    [0.13881,0.12443,−0.38995,0.24172,−0.48058,0.31295,−0.15538,
       0.33024,0.14607,−0.26831]
(Rsq)  0.80066
```

$R^2 = 0.8$ is the value for the sum of the squares of the residuals.

```
(%i49) EX : makelist ([X[j], E[j]], j, 1, length (X));
(EX)   [[0,0.13881],[0.9,0.12443],[1.8,−0.38995],[2.6,0.24172],[3.3,
       −0.48058],[4.4,0.31295],[5.2,−0.15538],[6.1,0.33024],[6.5,0.14607],[
       7.4,−0.26831]]
```

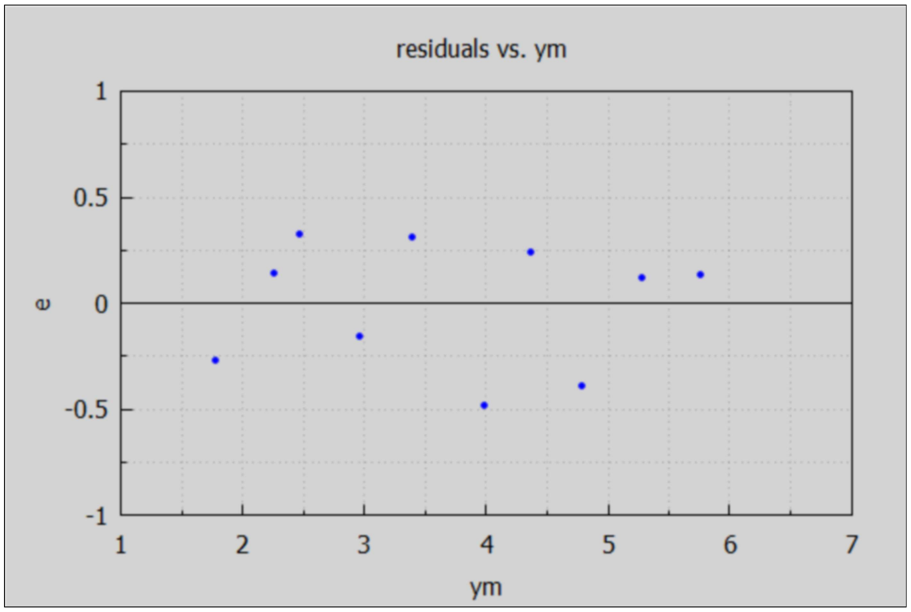
```
(%i50) wxdraw2d ( xrange = [0, 8], yrange = [-1, 1], xlabel = "x",
         ylabel = "e",   title = "residuals vs. x ", point_type = filled_circle,
       background_color = light_gray, line_width = 1, grid = [2,2], point_size = 0.6,
        points (EX), color = black, explicit (0, x, 0, 8)  )$
```

(%t50)


```
(%i51) EYm : makelist ([Ym[j], E[j]], j, 1, length (X) );
(EYm) [[5.7612,0.13881],[5.2756,0.12443],[4.7899,−0.38995],[4.3583,
0.24172],[3.9806,−0.48058],[3.387,0.31295],[2.9554,−0.15538],[
2.4698,0.33024],[2.2539,0.14607],[1.7683,−0.26831]]
```

```
(%i52) wxdraw2d ( xrange = [1, 7], yrange = [-1, 1], xlabel = "ym",
      ylabel = "e",   title = "residuals vs. ym ", point_type = filled_circle,
    background_color = light_gray, line_width = 1, grid = [2,2], point_size = 0.6,
     points (EYm), color = black, explicit (0, x, 1, 7)  )$
```

(%t52)



5.3.3 Standardizing the Residuals

Quoting Senese, p. 269:
"Standardizing the residuals (dividing them all by the standard error of the fit) helps clarify their distribution. Standardized residuals are in units of standard deviation, so roughly 2/3 should fall between +/- 1, and about 95% should fall between +/- 2."

Quoting Senese, p. 264 - 265:
"We must have some estimate of the unknown error variance σ^2 to use these equations [formulas for the variance of the parameters m and c when using the model y = m*x + c]. Remember that the residuals include both experimental error and errors due to the model's lack of fit, while the error variance σ^2 includes only experimental error. We can use the residual variance s^2 as an estimate of the error variance σ^2 if we assume that our model is correct.

s^2 = R^2 / (N - p),

where R^2 is the sum of the squared residuals, N is the number of data points, and p is the number of fitted parameters." (N-p) is also called the number of degrees of freedom (dof).

"The residual variance s^2 (the estimated variance of an ordinary least-squares fit) has two contributions: one is variance in the experimental data and the other is variance due to flaws in the model. We call these contributions the 'error variance' and the 'lack-of-fit variance, respectively."

"In 'good' models, the residual variance s^2 is close to the error variance σ^2."
"... The difference between s^2 and σ^2 is the basis for statistical tests for lack-of-fit..."

"The square root of s^2 is variously called the 'standard error of the regression', the 'residual standard error', 'standard error of the fit', and the 'standard error of estimation'.
It estimates the standard deviation of the residuals around a mean residual of zero.
If s is zero, we have a perfect fit, with all points lying exactly on the line. If s is large, at least some of the points lie far away from the [best-fit] line - and the linear model is less useful for predicting individual data points."

s is the standard error of the fit. The number of degrees of freedom (dof) is 10 - 2 = 8.

```
(%i53) s : sqrt (Rsq/8);
(s)    0.31636
```

Let Es be the list of standardized residuals.

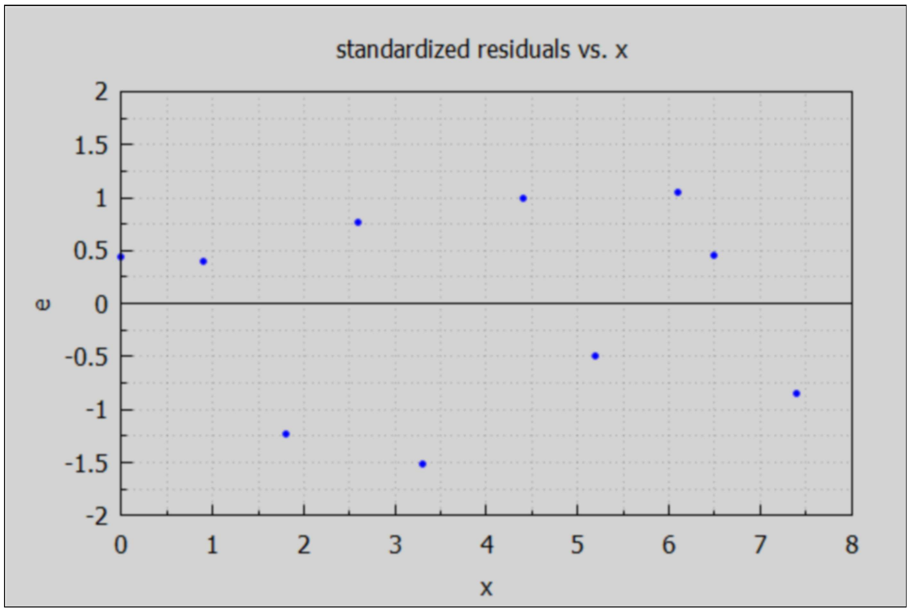
```
(%i54) Es : E / s;
(Es)   [0.43879,0.39333,−1.2326,0.76406,−1.5191,0.98924,−0.49116,
       1.0439,0.46171,−0.84813]
```

Plot the standardized residuals against x.

```
(%i55) EsX : makelist ([X[j], Es[j]], j, 1, length (X));
(EsX)  [[0,0.43879],[0.9,0.39333],[1.8,−1.2326],[2.6,0.76406],[3.3,−
       1.5191],[4.4,0.98924],[5.2,−0.49116],[6.1,1.0439],[6.5,0.46171],[7.4,
       −0.84813]]
```

```
(%i56) wxdraw2d ( xrange = [0, 8], yrange = [-2, 2], xlabel = "x",
         ylabel = "e",   title = "standardized residuals vs. x ", point_type = filled_circle,
       background_color = light_gray, line_width = 1, grid = [2,2], point_size = 0.6,
        points (EsX), color = black, explicit (0, x, 0, 8)  )$
```

(%t56)



Six out of ten of the residuals lie in the interval -1 < e < 1, roughly two-thirds.
All lie in the interval -2 < e < 2.

5.3.4 lsquares_residual_mse (dataM, varL, eqn, pvalL)

lsquares_residual_mse (dataM, varL, eqn, pvalL) in general calculates
(1/N)* Σ [lhs(eqn) - rhs(eqn)]^2,
where N = number of data points.
For our case
lsquares_residual_mse (M, [x,y], y = m*x + c, [m = -0.53958, c = 5.7612])
calculates
(1/N) * R^2, which here is:
(1/N)*Σ [y_i - mls*x_i - cls]^2 = (1/10)* Σ [y_i + 0.53958*x_i - 5.7612]^2.

We defined the data-matrix M above when using lsquares_estimates on this data set.

```
(%i57) M;
```

$$
(\%o57)\quad \begin{pmatrix} 0 & 5.9 \\ 0.9 & 5.4 \\ 1.8 & 4.4 \\ 2.6 & 4.6 \\ 3.3 & 3.5 \\ 4.4 & 3.7 \\ 5.2 & 2.8 \\ 6.1 & 2.8 \\ 6.5 & 2.4 \\ 7.4 & 1.5 \end{pmatrix}
$$

We also defined a list called solns:

```
(%i58) solns;
(%o58) [m=-0.53958,c=5.7612]
```

Let lsr_mse be the return value of using lsquares_residual_mse on this data set, which calculates (1/N) * R².

```
(%i59) lsr_mse : lsquares_residual_mse (M, [x,y], y = m*x + c, solns ), numer;
(lsr_mse) 0.080066
```

We can then calculate the value of R² for this data set using

```
(%i60) Rsq : length (M)*lsr_mse;
(Rsq)  0.80066
```

An estimated error for the dependent variable y gained from the data is found from
σy = s = sqrt (R² / (N - p)) = sqrt (N*lsr_mse / (N - p)), in which p is the number of parameters determined by the data set. Here, p = 2, since we determined m and c.

```
(%i61) σy : sqrt (Rsq / 8);
(σy)   0.31636
```

The value of χ² = R² / σy^2 can be used in linear problems to assess the plausibility of the parameter values found. If the number of degrees of freedom (N - p) is dof, then the answers are acceptable if the value of χ² lies in the region dof +/- sqrt (2*dof), that is the range [dof - sqrt (2*dof), dof + sqrt (2*dof)].

For this problem, dof = 8, and sqrt (2*dof) = sqrt (16) = 4, and the results found for m and c are acceptable if χ^2 lies in the range [4, 12].

```
(%i62) Chisq : Rsq / σy^2;
(Chisq) 8.0
```

For this problem our estimate of χ^2 is equal to the number of degrees of freedom (8), and we can trust our evaluation of m and c.

5.4 Analytic Errors of the parameter values calculated

5.4.1 Hand Calculation

We defined the symbol σy above.

```
(%i63) σy;
(%o63) 0.31636
```

Likewise the lists X and Y.

```
(%i65) X;
       Y;
(%o64) [0,0.9,1.8,2.6,3.3,4.4,5.2,6.1,6.5,7.4]
(%o65) [5.9,5.4,4.4,4.6,3.5,3.7,2.8,2.8,2.4,1.5]
```

Most of this notation, such as <xy>, has been previously introduced in writing the formula for the linear correlation coefficient above.

Define the average of the set of x_i's:
  <x> = (1/N)* Σx_i = Lsum (X)/N,
and the average of the corresponding y_i's:
  <y> = (1/N)* Σy_i = Lsum (Y)/N.
Also define the symbols
  <xy> = (1/N)* Σ(x_i * y_i) = Lsum (X*Y) / N, and
  <x²> = 1/N)* Σ(x_i)² = Lsum (X^2) / N.
Let Δ = <x^2> - <x>^2.

Then, we will prove later that
 m = [<xy> - <x><y>] / Δ, and given m one can find   c = <y> - m*<x>.
Also errors in m and c are given by:
 σm = σ / (N*Δ)^(1/2),
 σc = [<x^2)/(N*Δ)]^(1/2) * σ.

For σ we use σy = s, the 'standard error of the fit'.

```
(%i75) N : length (M);
       Xav : Lsum (X) / N;
       Yav : Lsum (Y) / N;
       XYav : Lsum (X*Y) / N;
       Xsqav : Lsum (X^2) / N;
       del : Xsqav - Xav^2;
       mm : (XYav - Xav*Yav) / del;
       cc : Yav - mm*Xav;
       σm : σy / sqrt (N*del);
       σc : σy * sqrt (Xsqav / (N*del));
(N)    10
(Xav)  3.82
(Yav)  3.7
(XYav) 11.091
(Xsqav)20.232
(del)  5.6396
(mm)   −0.53958
(cc)   5.7612
(σm)   0.042127
(σc)   0.18949
```

Which says m = -0.54 +/- 0.04 and c = 5.8 +/- 0.2 are the best-fit values of the slope m and the y-intercept c, using the model y = m*x + c with the data.

5.5 Plot of data and fit with error bars

We defined the list given of raw data above:

```
(%i76) given;
(%o76) [[0,5.9],[0.9,5.4],[1.8,4.4],[2.6,4.6],[3.3,3.5],[4.4,3.7],[5.2,
       2.8],[6.1,2.8],[6.5,2.4],[7.4,1.5]]
```

Our vertical error bars will extend over [y_i - σy, y_i + σy], and we need to construct a list having elements [x, y, σy].

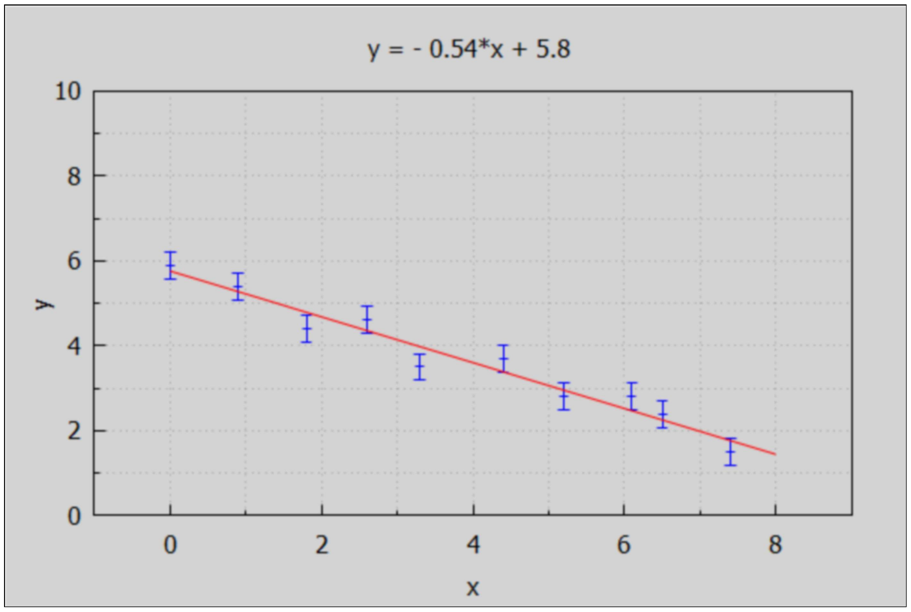
```
(%i77) errorL : makelist ([given[j][1], given[j][2], σy], j, 1, N);
(errorL) [[0,5.9,0.31636],[0.9,5.4,0.31636],[1.8,4.4,0.31636],[2.6,4.6,
       0.31636],[3.3,3.5,0.31636],[4.4,3.7,0.31636],[5.2,2.8,0.31636],[6.1,
       2.8,0.31636],[6.5,2.4,0.31636],[7.4,1.5,0.31636]]
```

We have also just defined the symbols mm and cc in our hand calculation above.0

```
(%i78) [mm, cc];
(%o78) [−0.53958,5.7612]
```

```
(%i79) wxdraw2d ( xrange = [-1,9], yrange = [0, 10], xlabel = "x",
         ylabel = "y",   title = "y = - 0.54*x + 5.8 ", error_type = 'y,
        background_color = light_gray, line_width = 1, grid = [2,2],
        errors (errorL), color = red, explicit (mm*x + cc, x, 0, 8) )$
```

(%t79)



6 Two Parameter Ordinary Least Squares Fit, OLS2 (xydata) or OLS2 (xydata, sigma-y)

Since, for the linear case, we have convenient answers in terms of the input data for both the parameters and their uncertainties, it is useful to have a specialized function for such a case which is often met. Derivations of the formulas used for the two parameters and their uncertainties are given in the next section.

For a two parameter y = m*x + c fit using OLS2 use either
OLS2 (xydata) or OLS2 (xydata, sigma-value). In both cases this Maxima function returns the list [m, σm, c, σc] and prints to the screen values of R^2 and χ^2.

The use of OLS2 (xydata) uses the standard error of the fit s for σy in the calculation of the errors in the parameters and for the calculation of X^2.

Using OLS2 (xydata, sigma-value) forces the use of the input σy in the calculation of the parameter errors and for the calculation of X^2.

```
(%i80) OLS2 ([input]) :=
         block ([usenew : false, ddata, newsig,nn, mm, cc, XX,YY, Xav, Yav,
              XYav, Xsqav, RRsq, scu, del, σmm, σcc, σyy ],
          ddata : input [1],
          if length (input) = 2 then (usenew : true, newsig : input[2]),
          nn : length (ddata),
          XX : makelist (ddata[j][1], j, 1, nn),
          YY : makelist (ddata[j][2], j, 1, nn),
          Xav : Lsum (XX) / nn,
          Yav : Lsum (YY) / nn,
          XYav : Lsum (XX*YY) / nn,
          Xsqav : Lsum (XX^2) / nn,
          del : Xsqav - Xav^2,
       mm : float ( (XYav - Xav*Yav) / del ),
       cc : float (Yav - mm*Xav),
       RRsq : float (Lsum ( (YY - mm*XX - cc)^2 )),
       print ( " R² = ", RRsq ),
          scu : sqrt ( RRsq / (nn - 2) ),
          print (" data derived estimate of σy = ", scu),
          if cc = 0 then print (sconcat (" y = ", mm, "*x "))
               else if cc > 0 then print (sconcat (" y = ",mm,"*x + ", cc))
               else  print (sconcat (" y = ",mm,"*x - ", abs (cc))),
       if usenew then σyy : newsig else σyy : scu,
       print (" we used sigma-y = " , σyy),
       σmm : float (σyy / sqrt (nn*del)),
       σcc : float (σyy * sqrt (Xsqav / (nn*del))) ,
          print (" σm = ",σmm, "  σc = ", σcc),
          print ( " χ² = ", RRsq/σyy^2 ),
          [mm, σmm, cc, σcc]     )$

(%i81) OLS2 (given);
        R^2 =  0.80066
        data derived estimate of σy =  0.31636
        y = −0.53958*x + 5.7612
        we used sigma−y =  0.31636
        σm =  0.042127    σc =  0.18949
        χ^2 =  8.0
(%o81) [−0.53958,0.042127,5.7612,0.18949]
```

which agrees with our hand calculation.

We use here methods explained in a later section: 'What the Value of X^2 Tells Us About the Quality of a Fit'.

The number of degrees of freedom is 10-2 = 8.

Two thirds of the values of X^2 should lie in the range described by mean +/- 1 standard dev. For the X^2 distribution describing a data set with n degrees of freedom, the mean of the distribution is n and one standard deviation is sqrt (2*n).

```
(%i83) mean_chi2 (8);
       std_chi2(8), numer;
(%o82) 8
(%o83) 4.0
```

```
(%i84) float ( integrate (pdf_chi2 (z, 8), z, 4, 12) );
(%o84) 0.70592
```

The probability of obtaining, by chance, a value of X^2 outside the range [4, 12] (for dof = 8) is 1 - 0.706 = 0.294. Hence the chance is about 30%.

The X^2 value of 8 lies within the range [4, 12] and is acceptable.

6.1 Derivations of Analytic Formulas for m, σm, c, and σc.

We can take the straight line model containing two adjustable parameters to be
 y = m*x + c,
in which m is the slope and c is the y-intercept (the value of y when x = 0).
Then
 $R^2 = \Sigma$ (y_i - m*x_i - c)^2
is to be minimized to find the best fit values of the two parameters m and c.

6.1.1 Formulas for m and c

We get two equations, to be solved simultaneously, by setting the first derivative of R^2 with respect to both m and c equal to zero.

$\partial R^2/\partial c = \Sigma (2)*(y_i - m*x_i - c)*(-1) = 0$, which implies

$\Sigma (y_i - m*x_i - c) = 0$,

$\Sigma y_i - m* \Sigma x_i - c*\Sigma(1) = 0$, now divide each term by N, the number of data points,

$(1/N)*\Sigma y_i - m*(1/N)* \Sigma x_i - c*(1/N)*\Sigma(1) = 0$,

$(1/N)*\Sigma y_i - m*(1/N)* \Sigma x_i - c = 0$.

Now define the average of the set of x_i's [using notation also used above in two sections]:

$<x> = (1/N)* \Sigma x_i$

and the average of the corresponding y_i's:

$<y> = (1/N)* \Sigma y_i$,

to get

$c = <y> - m*<x>$. (1)

A similar calculation starting with $\partial R^2/\partial m = 0$ then leads to the second needed equation:

$m = [<xy> - <x>*<y>] / [<x^2> - <x>^2]$, (2)

in which

$<xy> = (1/N)* \Sigma(x_i * y_i)$, and

$<x^2> = 1/N)* \Sigma(x_i)^2$.

Thus we can first use (2) to calculate the slope (gradient) m, and then use (1) to obtain c.

6.1.2 Formulas for σm and σc

Let Δ = <x^2> - <x>^2.

Then m = [<xy> - <x><y>] / Δ

= (1/Δ) * { (1/N) * Σ (x_i*y_i) - (1/N) * Σ (<x>*y_i) }

= [1/ (N*Δ)] * Σ (x_i - <x>)*y_i

= Σ α (x_i) * y_i,

where α (x_i) = (x_i - <x>) / (N*Δ).

Since errors add in quadrature, the variance of m is

V(m) = Σ [α (x_i)]^2 * (σ_i)^2 --> σ^2 * Σ [α (x_i)]^2 .

Now Σ [α (x_i)]^2 = [1/(N*Δ)^2] * Σ (x_i - <x>)^2x = [1/(N*Δ)^2] * (N*Δ) = 1/(N*Δ),

and this produces the variance

V(m) = σ^2 / (N*Δ) == σm^2, so

σm = σ / (N*Δ)^(1/2).

Inserting the solution found for m into c = <y> - m*<x>, we get

c = [<x^2>*<y> - <x>*<xy>] / Δ

= Σ β(x_i) * y_i,

where β(x_i) = [<x^2> - <x>*x_i] / (N*Δ).

Using errors add in quadrature again, the variance of c is
$V(c) = \Sigma\, [\beta(x_i)]^2 * (y_i)^2 \;\text{--->}\; \sigma^2 * \Sigma\, [\beta(x_i)]^2$.

A llittle work shows that
$\Sigma\, [\beta(x_i)]^2 = \langle x^2\rangle / (N*\Delta)$, which means the variance of c
$V(c) = [\langle x^2\rangle / (N*\Delta)] * \sigma^2 = \sigma c^2$.
Taking the square root then gives
$\sigma c = [\langle x^2)/(N*\Delta)\,]^{(1/2)} * \sigma$.

7 *What the X^2 value tells us about the quality of a fit*

Following Barlow Sec. 6.1:
"The principle of least squares can be derived from the principle of maximum likelihood (if the measurements follow a Gaussian distribution: see section 5.6). Alternatively, it can just be regarded as an obviously sensible estimator. 'Least squares' means just what it says: you minimise the (suitably weighted) squared difference between a set of measurements and their predicted values."
Your measured values are the set of data (x_i, y_i), and the 'predicted values' refer to the values returned by an assumed function $f(x_i; a)$, letting 'a' stand for the set of model parameters.
"This is done by varying the parameters you want to estimate. The predicted values are adjusted so as to be close to the measurements; squaring the differences means that greater importance is placed on removing the large deviations."

"Assume "each value y_i has been measured with some accuracy σ_i.
Take the sum over all points of the squared difference between the measurement y_i and the 'prediction' $f(x_i; a)$, scaled by the expected error σ_i. This sum is called $\chi 2$:

$$X^2 = \Sigma\, [\, (y_i - f(x_i;\; a)) / \sigma_i\,]^2$$
$$= \Sigma\, [\, (y\_i\_actual - y\_i\_ideal) / \text{expected-error}\,]^2$$

"Then choose the value(s) of a which gives the smallest X^2. If all the σ_i are the same, then σ comes out as a common factor and a lot of the algebra is much simpler. If the derivatives of f with respect to a_j are known, as they usually are, then the minimisation problem becomes one of finding a solution of the equation(s):

$$\partial X^2/\partial a_j = 0."$$

"If the function f (x_i; a) agrees well with the actual values then X^2 will be small.
If X^2 is large at the end of your minimising, it is telling you that there is something wrong with the answer. However, a very small value of X^2 is also unlikely; the errors should make the measurements deviate from their ideal values to some extent, and a very low X^2 probably means that these errors have been overestimated."
"To be more specific requires the distribution for X^2, which is
(here Γ(x) is the standard gamma function) :"
    P (X^2; n) = [χ^(n - 2) * exp (- χ2 / 2)] / [2^(n/2) * Γ (n/2)],
or, if z = χ2,
    P (z; n) = [z^(n/2 - 1) * exp (- z/2)] / [2^(n/2) * Γ (n/2)].

```
(%i85) P (zz, nn) :=
         (if zz <= 0 then 0
          else zz^(nn/2 - 1) * exp (- zz/2)  / (2^(nn/2) * gamma (nn/2) ) ) $

(%i86) P (8, 10), numer;
(%o86) 0.097683
```

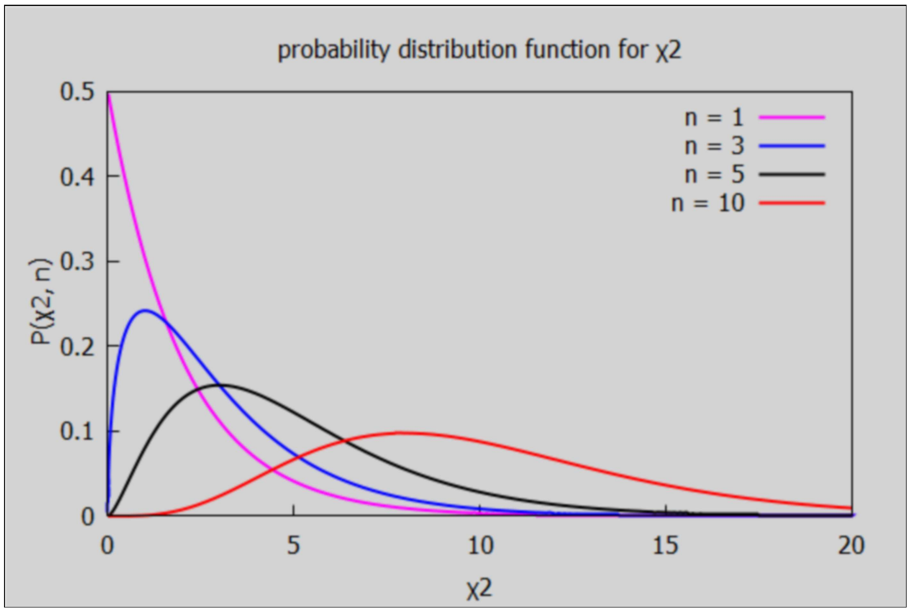
Maxima has the probability distribution function pdf_chi2 (z, n), where z is the value of X^2 and n is the number of degrees of freedom (defined in the Maxima package distrib).

```
(%i87) pdf_chi2 (8, 10), numer;
(%o87) 0.097683
```

"The distribution depends on n, which is the number of [data] points in the sum, N, minus the number of variables that have been adjusted to minimize X^2. It is called the number of degrees of freedom. Distributions are shown below for n = 1, 3, 5, and 10."

```
(%i88) wxdraw2d (xrange = [0, 20], yrange = [0, 0.5], xlabel = "χ2", ylabel =" P(χ2, n)",
          title = "probability distribution function for χ2",
          background_color = light_gray, line_width = 2,
          key = "n = 1", color = magenta, explicit (pdf_chi2 (z, 2), z, 0.01, 20),
          key = "n = 3", color = blue, explicit (pdf_chi2 (z, 3), z, 0.0001, 20),
          key = "n = 5",color = black, explicit (pdf_chi2 (z, 5), z, 0.001, 20),
          key = "n = 10", color = red, explicit (pdf_chi2 (z, 10), z, 0, 20) )$
```

(%t88)



The area (probability) under each of these curves is unity (1).

```
(%i89) float ( integrate (pdf_chi2 (z, 10), z, 0, inf) );
(%o89) 1.0
```

"The X² distribution has mean n and variance 2*n. Thus one expects a X² per degree of freedom of roughly one, [ie., X² ~ n] and there are grounds for suspecting that something is wrong if it is much bigger. However, although the central limit theorem ensures that the X² distribution tends to a Gaussian for large n, the values of n required for this are extremely large. It is better to use sqrt (2 * X²) which has a distribution that is a reasonable approximation to a Gaussian with mean sqrt (2*n - 1) and unit variance for values of n greater than about 30."

```
(%i90) float ( integrate (z * pdf_chi2 (z, 10), z, 0, inf) );
(%o90) 10.0
```

```
(%i92) mean_chi2 (10);
       std_chi2 (10), numer;
(%o91) 10
(%o92) 4.4721
```

7.0.1 Example

This is an example from K.K. Gan, Ohio State (Chap. 6, p. 4).

"What’s the probability to have X^2 >10 with the number of degrees of freedom n = 4?"

```
(%i93) float ( integrate (pdf_chi2 (z, 4), z, 10, inf) );
(%o93) 0.040428
```

"Thus P (X^2 >10, n = 4) = 0.04.
We say the probability of getting X^2 >10 with 4 degrees of freedom by chance is 4%."

7.1 One Parameter Fit to y = m*x + C, with C given

The (x,y) data is to be fitted to the straight line y = m*x + C, with the y-intercept C given.
Let X be the list of x_i values, Y be the list of y_i values.
The best value of m is the value which minimizes (assuming constant σ_i == σ)
  X^2 = (1/σ^2) * Σ (y_i - m*x_i - C)^2 = Lsum ((Y - m*X - C)^2) / σ^2.

Requiring ∂X^2/∂m = 0 produces an equation which can be solved for the value of m which produces an extremum in X^2, and using the combination of errors equation with the error estimate σ of each measured y_i leads to an estimate of σm (the error in m).

Since all the assumed error σ_i of the y_i data values are assumed approximately equal to σ, the best fit value of m is the value which minimizes
   S = Σ (y_i - m*x_i - C)^2.

∂S/∂m = 2 * Σ (y_i - m*x_i - C) * (- x_i) = 0, or Σ [x_i * y_i - m*(x_i)^2 - C*x_i] = 0, or
using notation we have used multiple times above,
<xy> - m*<x^2> - C*<x> = 0, thus

m = [<xy> - C*<x>] / <x^2> and

<x^2> * m = <xy> - C*<x>, and now define M as:
M == <x^2> * m + C*<x> = <xy>, so M is proportional to the y_i values.
Thus
M = (1/N)*Σ x_i * y_i = Σ α(x_i) * y_i, with α(x_i) = x_i / N.

Using "errors add in quadrature", we get the variance of M, assuming the individual errors σy_i are all approximately the same value σ:

V(M) = Σ [α(x_i)]^2 * (σy_i)^2 --> σ^2 * Σ [α(x_i]^2.

But Σ [α(x_i]^2 = Σ (x_i/N)^2 = <x^2> / N.
Hence
V(M) = σ^2 * <x^2> /N.

Since M == <x^2> * m + C*<x> and C is treated as a known constant,
a small change in M (dM) will be associated with a small change in m (dm), via
dM = <x^2> * dm, so (dm)^2 = (dM)^2 / (<x^2>)^2, and hence
V(m) = V(M) / (<x^2>)^2, or
V(m) = σ^2 / [N*<x^2>], and finally:

σm = σ / [N*<x^2>]^(1/2).

In the function OLS1 (data-list, C, sigma) defined below, sigma is the assumed uncertainty of the measured values of the dependent variable (y), all approximately the same.

If a value of the uncertainty of the measured values y_i is not given, we use the data derived "common uncertainty" σcu of the measured values of the y_i, which
we called the standard error of the fit above,
σcu = (R^2 / (N - 1))^(1/2), or

σcu ~ [Σ (y_i - m*x_i - C)^2 / (N - 1)]^(1/2) = [Lsum ((Y - m*X - C)^2) / (N - 1)]^(1/2),

where N is the number of data points, N-1 is the number of degrees of freedom after using the N data points to determine the least squares value of m.

7.2 OLS1 (data-list, C), OLS1 (data-list, C, sigma), with C given

This Maxima function fits the xydata to y = m*x + C, given known constant C. The third argument sigma is optional. If the form OLS1 (data-list,C) is used, the standard error of the fit σcu = s is used for the common weighting factor σ_i, in the expression X² = Σ [(y_i - m*x_i - C) / σ_i]^2 which is to be minimized by choice of m.

The input data has the form [[x1,y1], [x2,y2], ...].
If present, the input value 'sigma' is the assumed constant error in the y_i measurements.

The output is the list [m, σm], the best fit values of the slope m and the error in the slope, σm.

In addition to the output list, this function prints to the screen the calculated values of χ2 and σcu.

```
(%i94) OLS1 ([input]) :=
         block ([usenew : false, ddata, newsig,nn, mm, CC, XX,YY, Xav,
                      XYav, Xsqav, RRsq, scu,  σmm,  σyy ],
          ddata : input [1],
          CC : input [2],
          if length (input) = 3 then (usenew : true, newsig : input[3]),
          nn : length (ddata),
          XX : makelist (ddata[j][1], j, 1, nn),
          YY : makelist (ddata[j][2], j, 1, nn),
          Xav : Lsum (XX) / nn,
          XYav : Lsum (XX*YY) / nn,
          Xsqav : Lsum (XX^2) / nn,
       mm : float ( (XYav - CC*Xav) / Xsqav),

       RRsq : Lsum ( (YY - mm*XX - CC)^2 ),
       print ( " R² = ", RRsq ),
          scu : float (sqrt ( RRsq / (nn - 1) )),
          print (" data derived estimate of σy = ", scu),
          if CC = 0 then print (sconcat (" y = ", mm, "*x "))
               else if CC > 0 then print (sconcat (" y = ",mm,"*x + ",CC))
               else  print (sconcat (" y = ",mm,"*x - ", abs (CC))),
       if usenew then σyy : newsig else σyy : scu,
       print (" we used sigma-y = " , σyy),
       σmm : float (σyy / sqrt (nn*Xsqav)),
          print (" σm = ",σmm ),
          print ( " χ² = ", float ( RRsq/σyy^2) ),
          [mm, σmm ]  )$
```

7.2.1 Barlow, Prob. 6.1   y = m*x

"A trolley moves along a track with a constant speed, which you need to measure. It passes through the point d = 0 at exactly t = 0. At certain fixed times, determined by a stroboscope, you photograph the trolley and can thus measure its position with an error of 2 mm."
From the data given we infer that we can measure the time with an error of 0.1 sec.
We choose the independent variable 'x' to be the time t, and the dependent variable 'y'
to be the distance d from the origin. The results are presented here in the form
of a data list: [[t1, d1], [t2, d2], [t3, d3], ... [t6, d6]],
in which t is in units of seconds, and d is in units of mm.

```
(%i95)   mydata : [ [1.0,11], [2.0,19], [3.0,33], [4.0,40], [5.0,49], [6.0,61] ];
(mydata) [[1.0,11],[2.0,19],[3.0,33],[4.0,40],[5.0,49],[6.0,61]]
```

Find the constant speed v, the estimated error in the speed, and the value of X².

Our model is d = v*t with v a constant speed. The assumed constant error of the dependent variable d is 2 mm, which we take to be σy

```
(%i96)   [v, σv] :  OLS1 (mydata, 0, 2);
         R^2 =   12.11
         data derived estimate of σy =   1.5563
         y = 10.099*x
         we used sigma−y =   2
         σm =   0.20966
         χ^2 =   3.0275
(%o96)   [10.099,0.20966]
```

This says v = (10.1 +/- 0.2) mm/sec.

Two thirds of the values of X² should lie in the range described by mean +/- 1 standard dev. For the X² distribution describing a data set with n degrees of freedom, the mean of the distribution is n and one standard deviation is sqrt (2*n).

We have n = 5 degrees of freedom (6 - 1).

```
(%i98)   mean_chi2(5);
         std_chi2(5), numer;
(%o97)   5
(%o98)   3.1623
```

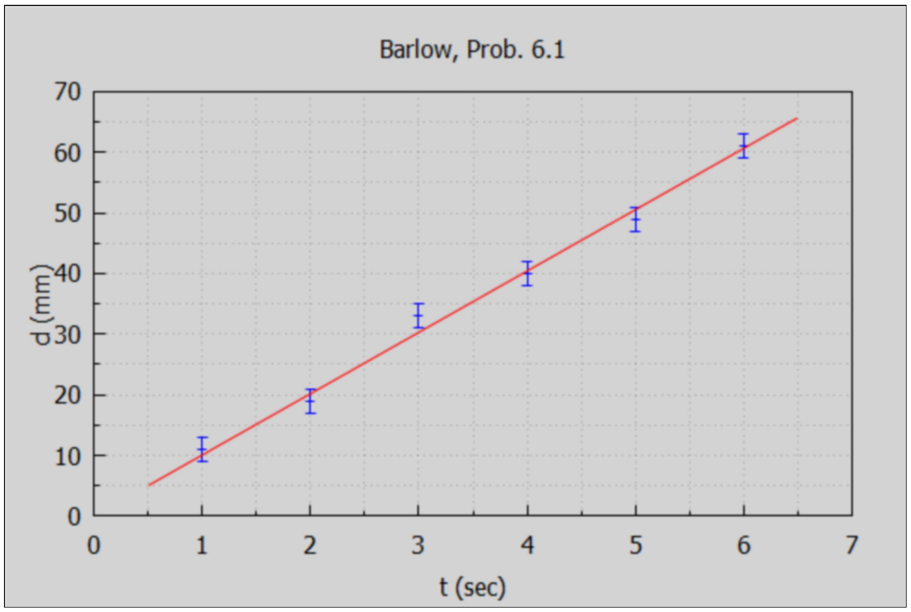
The X² value of 3.0 lies within the range [1.8, 8.2] and is acceptable so we can trust the value of v found.

7.2.2 Data Plot with Error Bars and Linear Trend Line

To use the draw2d format for error bars we need to construct the list needed as an argument to the draw2d function errors. Each sublist for our case has the form
[t, d, σd] = [t, d, 2] since σd = 2 mm.

```
(%i99)   ErrorL : makelist ([mydata[j][1], mydata[j][2], 2], j, 1, length (mydata) );
(ErrorL) [[1.0,11,2],[2.0,19,2],[3.0,33,2],[4.0,40,2],[5.0,49,2],[6.0,
         61,2]]
```

```
(%i100) wxdraw2d( xlabel = "t (sec)", ylabel = "d (mm)", xrange = [0, 7], yrange = [0, 70],
           error_type = 'y, line_width = 1, background_color = light_gray, grid = [2,2],
        errors( ErrorL ), title = "Barlow, Prob. 6.1", color = red, explicit (v*t, t, 0.5, 6.5) )$
```

(%t100)


7.2.3 Barlow, Prob. 6.2

"A trolley moves along a track with a constant speed, which you need to measure. It passes through the point d = 0 at exactly t = 0. At certain fixed distances, determined by sensing devices on the track, the time is measured with an error of 0.1 sec. "

We choose the independent variable 'x' to be the distance d, and the dependent variable 'y' to be the time t. The results are presented here in the form of a data list:
    [[d1, t1], [d2, t2], [d3, t3], ... [d6, t6]],
in which d is in units of mm and t is in units of seconds.

```
(%i101) mydata : [ [10, 1.1], [20, 2.2], [30, 2.9], [40, 4.1], [50, 5.0], [60, 5.8] ];
(mydata) [[10,1.1],[20,2.2],[30,2.9],[40,4.1],[50,5.0],[60,5.8]]
```

"Find the value of the constant velocity, the error in the velocity, and the value of X² from this data."

We infer from the given data that the uncertainty in the distance measurement is +/- 1 mm.

We plot the data with a linear trend line, assuming t = m*d, with m the unknown gradient (slope). m has units sec/mm. The dependent variable t has σt = σy = 0.1 sec.

```
(%i102) [m, σm ] : OLS1 (mydata, 0, 0.1);
         R^2 =  0.10604
         data derived estimate of σy =  0.14563
         y = 0.099341*x
         we used sigma−y =  0.1
         σm =  0.0010483
         χ^2 =  10.604
(%o102) [0.099341,0.0010483]
```

Then, roughly, m = (0.1 +/- 0.001) sec/mm.

Our model is t = m*d = (1/v)*d, so v = 1/m and dv = |∂v/∂m| * dm, or σv = σm / m^2.

```
(%i103) [v, σv] : [1/m, σm/ m^2];
(%o103) [10.066,0.10622]
```

Thus an experimental value for v from our data is (10.1 +/- 0.1) mm/sec.

We have n = 5 degrees of freedom (6 - 1).

```
(%i105) mean_chi2(5);
        std_chi2(5), numer;
(%o104) 5
(%o105) 3.1623
```

The X² value of 10.6 lies outside the acceptable range [1.8, 8.2] for five degrees of freedom and is not acceptable. Hence we cannot trust the value of v implied by the least squares one parameter fit to this data.

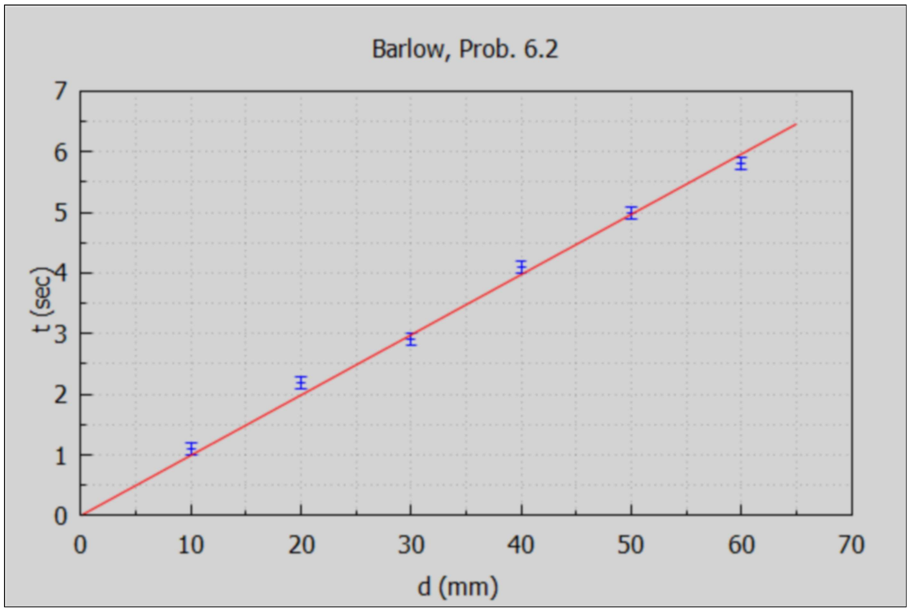
We now plot the data with error bars.

With independent variable d and dependent variable t, each sublist of ErrorL should have the form [d, t, σt] = [d, t, 0.1], since σt = 0.1 sec.

```
(%i106) ErrorL : makelist ([mydata[j][1], mydata[j][2], 0.1], j, 1, length (mydata) );
(ErrorL) [[10,1.1,0.1],[20,2.2,0.1],[30,2.9,0.1],[40,4.1,0.1],[50,5.0,
        0.1],[60,5.8,0.1]]
```

```
(%i107) wxdraw2d( xlabel = "d (mm)  ", ylabel = " t (sec)", xrange = [0, 70], yrange = [0, 7],
        error_type = 'y, line_width = 1,background_color = light_gray, grid = [2,2],
     errors( ErrorL ), title = "Barlow, Prob. 6.2", color = red, explicit (m*d, d, 0, 65) )$
```

(%t107)



7.2.4 Barlow, Prob. 6.3, "Linearizing the Data"

"You are determining the acceleration due to gravity by switching off an electromagnet to release a ball-bearing, and measuring the time t it takes to fall a fixed distance d, for several fixed distances. We assume d = (1/2)*g*t^2. The distances are measured precisely, the time with an accuracy of 0.01 sec."

We take d as the independent variable, and take the time to fall, t, as the dependent variable. Here is the given data, with each sublist of the form [d, t], d in meters, t in sec,

```
(%i108) given : [ [0.20,0.16], [1.00, 0.40], [2.00, 0.58], [3.00, 0.72], [5.00, 0.97] ];
(given) [[0.2,0.16],[1.0,0.4],[2.0,0.58],[3.0,0.72],[5.0,0.97]]
```

"Calculate the acceleration due to gravity, with the appropriate error, assuming
(A) first that the times are as given, and
(B) second that the field in the magnet takes an unknown but constant time to die away and release the ball-bearing.
Comment on the difference, and on the X^2 of the two fits."

We can 'linearize the data' by fitting the model
t = sqrt (2/g) * sqrt (d) == m*x,
and find a least squares fit to a straight line between t and x = sqrt (d) as a first step.
For part (A) we assume release of the ball from rest by the electromagnet happens exactly at t = 0. So we fit the equation y = m*x, with y = t and x = sqrt(d), and find the values of m and σm.
Our list of data presented to OLS1 should be of the form
[[x1, y1], [x2, y2], ...] ==> [[sqrt(d1), t1], [sqrt(d2), t2], ...].

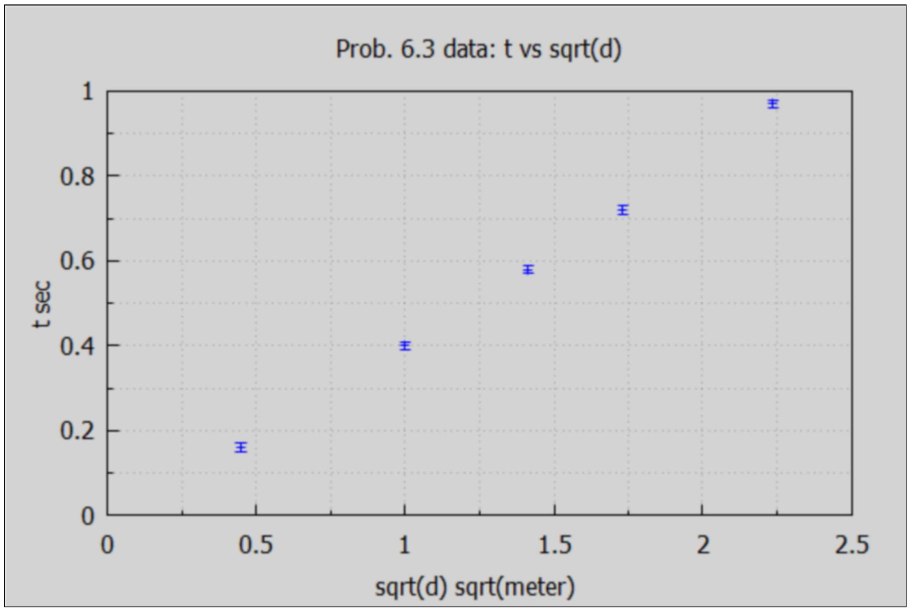
```
(%i109) mydata : makelist ([sqrt (given[j][1]), given[j][2] ], j, 1, length (given) );
(mydata) [[0.44721,0.16],[1.0,0.4],[1.4142,0.58],[1.7321,0.72],[2.2361,
0.97]]
```

We are told in the problem that the error in the time measurement is 0.0.1 sec.
We plot the data in terms of t vs. d^(1/2) using vertical error bars.

```
(%i110) errorL : makelist ([mydata[j][1],mydata[j][2],0.01], j, 1, length (mydata) );
(errorL) [[0.44721,0.16,0.01],[1.0,0.4,0.01],[1.4142,0.58,0.01],[1.7321,
0.72,0.01],[2.2361,0.97,0.01]]
```

```
(%i111) wxdraw2d (xlabel = "sqrt(d) sqrt(meter) ", ylabel = "t sec", xrange = [0,2.5],
    yrange = [0,1],  title = "Prob. 6.3 data: t vs sqrt(d)",   grid = [2,2],
    background_color = light_gray,  error_type = 'y, errors (errorL) )$
```

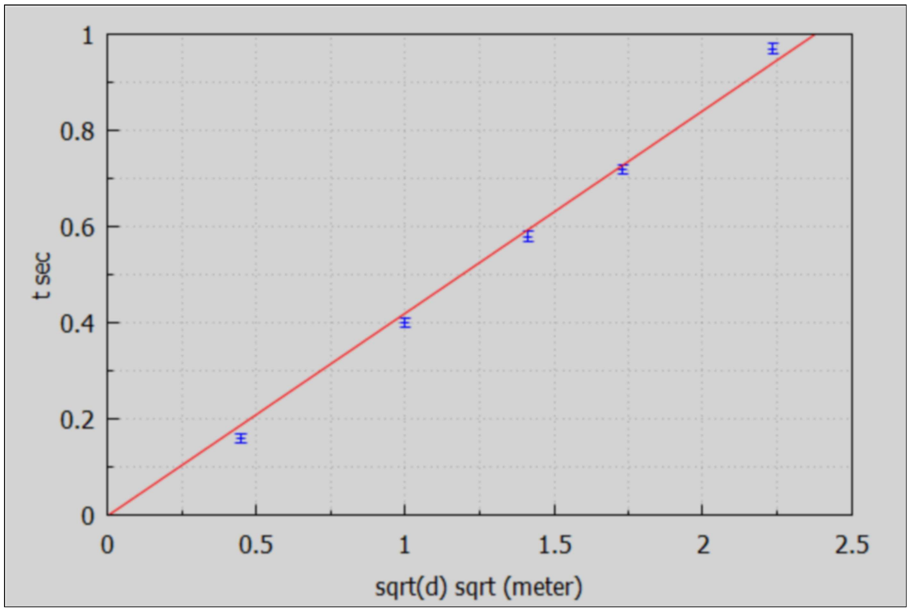
(%t111)



7.2.4.1

A.) Assume a one parameter fit y = m*x with y = t, σy = σt = 0.01 sec, x = sqrt(d) and the slope parameter m = sqrt(2/g). This assumes t = 0 when d = 0.

```
(%i112) [m1, σm1] : OLS1 (mydata, 0, 0.01);
        R^2 =  0.0023758
        data derived estimate of σy =  0.024371
        y = 0.42034*x
        we used sigma−y =  0.01
        σm =  0.0029881
        χ^2 =  23.758
(%o112) [0.42034,0.0029881]

(%i113) wxdraw2d (xlabel = "sqrt(d) sqrt (meter)", ylabel = "t sec", xrange = [0,2.5],
           yrange = [0,1],  error_type = 'y, background_color = light_gray, line_width = 1,
           grid = [2,2],  errors (errorL), color = red, explicit (m1*x, x, 0, 2.4) )$
(%t113)
```



We have 4 degrees of freedom (5 - 1) and get χ² = 24, so we cannot trust the answer provided by this one parameter fit, and the resulting value for g.

Since m = sqrt(2/g), g = 2/m^2, and dg = (2)*(-2/m^3)*dm, or σg = (4/m^3)*σm.

```
(%i115) g1 : 2/m1^2;
        σg1 : (4/m1^3) * σm1;
(g1)    11.319
(σg1)   0.16093

(%i117) mean_chi2(4);
        std_chi2(4), numer;
(%o116) 4
(%o117) 2.8284
```

The one parameter fit is not satisfactory since χ² = 24 is so large and far outside the range [1.2, 6.8], and thus we cannot trust the result that g = 11.3 meters/sec^2.

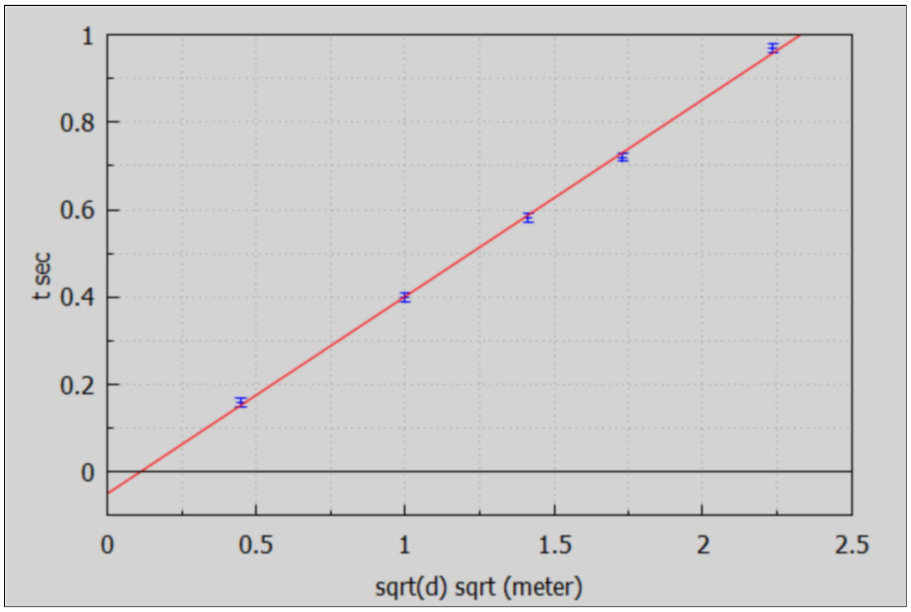
B.) Now assume a two parameter fit to the data, y = m*x + c, which can be interpreted as an acknowledgement that t is not zero when d equals zero. Since y = t and x = sqrt(d), when d = 0, t = c. We force OLS2 to use σy = σt = 0.01 sec.

```
(%i118) [m, σm, c, σc] : OLS2 (mydata, 0.01);
```

```
R^2 =  3.8692 10^-4
data derived estimate of σy =  0.011357
y = 0.4501*x − 0.048791
we used sigma−y =  0.01
σm =  0.0073099    σc =  0.01094
χ^2 =  3.8692
(%o118) [0.4501,0.0073099,−0.048791,0.01094]
```

```
(%i119) wxdraw2d (xlabel = "sqrt(d) sqrt (meter)", ylabel = "t sec",
    xrange = [0,2.5], yrange = [-0.1,1], error_type = 'y,  grid = [2,2],
    background_color = light_gray, line_width = 1,
    errors (errorL), color = red,  explicit (m*x + c, x, 0, 2.5),
    color = black,  explicit (0, x, 0, 2.5) )$
```

(%t119)



Since m = sqrt(2/g), g = 2/m^2, and dg = (2)*(-2/m^3)*dm, or σg = (4/m^3)*σm.

```
(%i121) g : 2/m^2;
        σg : (4/m^3) * σm;
(g)     9.8723
(σg)    0.32067
```

This is for n = 3 degrees of freedom (5 data points, 2 parameters).

```
(%i123) mean_chi2 (3);
        std_chi2 (3), numer;
(%o122) 3
(%o123) 2.4495
```

The χ² value of 3.9 lies within the range [0.55, 5.45] and is acceptable.
Rounding σg to 0.3, the data implies g = (9.8 +/- 0.3) meter/sec^2.
Keeping two significant figures on the error, g = (9.87 +/- 0.32) meter/sec^2.

7.3 Barlow, Prob. 6.4 y = a*x + b*sin(x)

"If it is believed that y is given by y = a x + b sin x, find the least squares estimate for a and b. Apply this to the data:" [the independent variable x is in radians]

Each sublist has the form [x_i, y_i].

```
(%i124) mydata : [ [0.2, 0.599], [0.3, 0.896], [0.4, 1.189], [0.5, 1.479], [0.6, 1.756], [0.7, 2.044] ];
(mydata) [[0.2,0.599],[0.3,0.896],[0.4,1.189],[0.5,1.479],[0.6,1.756],[
        0.7,2.044]]
```

7.3.1 Using lsquares_estimates

```
(%i125) MM : apply ('matrix, mydata);
```

$$(MM)\quad \begin{pmatrix} 0.2 & 0.599 \\ 0.3 & 0.896 \\ 0.4 & 1.189 \\ 0.5 & 1.479 \\ 0.6 & 1.756 \\ 0.7 & 2.044 \end{pmatrix}$$

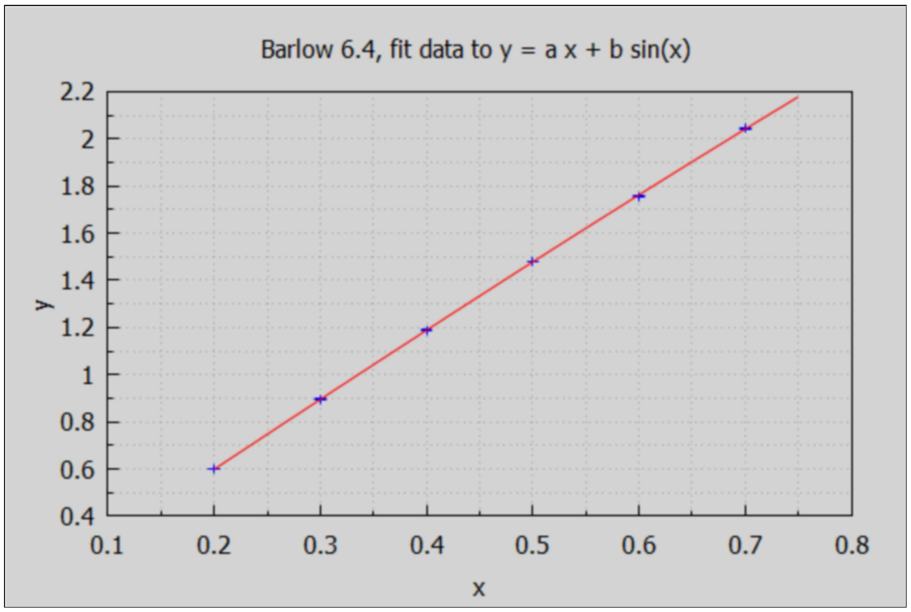
```
(%i126) solns : lsquares_estimates (MM, [x, y], y = a*x + b*sin (x), [a, b] ), numer;
(solns) [[a=1.9625,b=1.0355]]

(%i127) solns : solns [1];
(solns) [a=1.9625,b=1.0355]

(%i128) [a, b] : map ('rhs, solns);
(%o128) [1.9625,1.0355]
```

We infer from the given data that σy = 0.001 can be used for all data points.

```
(%i129) errorL : makelist ([mydata[j][1], mydata[j][2], 0.001], j, 1, length (mydata));
(errorL) [[0.2,0.599,0.001],[0.3,0.896,0.001],[0.4,1.189,0.001],[0.5,
         1.479,0.001],[0.6,1.756,0.001],[0.7,2.044,0.001]]
```

```
(%i130) wxdraw2d (xrange = [0.1, 0.8], yrange = [0.4, 2.2],  xlabel = "x", ylabel = "y",
            background_color = light_gray, grid = [2,2],
            title = "Barlow 6.4, fit data to y = a x + b sin(x) ",
            error_type = 'y,    errors (errorL),   line_width = 1,
            color = red, explicit (a*x + b*sin(x), x, 0.2, 0.75) )$
```

(%t130)



7.3.2 Using linear_regression (M)

linear_regression (M) is not designed for dealing with y = a*x + b*sin (x). linear_regression (M) can deal with y = c + m1*x1 + m2*x2 + ... in which the model is linear in each of the independent variables x1, x2, ...

7.3.3 Using Analytic Methods

y = a*x + b*sin (x) is linear in the fitting parameters a and b, which allows one to find analytic expressions for a and b, as well as for their errors.

We find the least squares best fit values of a and b by solving the pair of equations which require ∂X²/∂a = 0 and ∂X²/∂b = 0, in which
X² = (1/σ^2) * Σ (y_i - f(x_i))^2 = S / σ^2, with
S = Σ (y_i - a*x_i - b*sin(x_i))^2.

∂S/∂a = 2*Σ [(y_i - a*x_i - b*sin(x_i))*(- x_i)]
= -2* Σ [(x_i*y_i - a*(x_i)^2 - b*x_i*sin(x_i))] , so our first equation is
(I) Σ [(x_i*y_i - a*(x_i)^2 - b*x_i*sin(x_i))] = 0, which we can rewrite as

a*A + b*B = C, with A = Σ(x_i)^2, B = Σ (x_i*sin(x_i), C = Σx_i*y_i.

∂S/∂b = 2*Σ [(y_i - a*x_i - b*sin(x_i))* (- sin (x_i))]
= -2 * Σ [(y_i*sin(x_i) - a*x_i*sin(x_i) - b*sin(x_i) * sin(x_i))]
so our second equation is
(II) Σ [(y_i*sin(x_i) - a*x_i*sin(x_i) - b*sin(x_i) * sin(x_i))] = 0, which we write as

a*D + b*E = F, with D = Σx_i*sin(x_i), E = Σsin(x_i)*sin(x_i), F = Σy_i*sin(x_i).

We first find the symbolic solution for a and b given the pair of equations:
(I) a*A + b*B = C
(II) a*D + b*E = F.

```
(%i131) kill (a, b, c, A, B, D, E, F);
(%o131) done
```

```
(%i132) solve([A*a + B*b = C, D*a + E*b = F], [a, b] );
```

(%o132) $[[a=-\frac{CE-BF}{BD-AE},b=\frac{CD-AF}{BD-AE}]]$

```
(%i133) solns : %[1];
```

(solns) $[a=-\frac{CE-BF}{BD-AE},b=\frac{CD-AF}{BD-AE}]$

```
(%i134) [soln_a, soln_b] : map ('rhs, [solns[1], solns[2] ]);
```

(%o134) $[-\frac{CE-BF}{BD-AE},\frac{CD-AF}{BD-AE}]$

Let X be the list of the x_i values, Y be the list of y_i values.

```
(%i136) X : makelist (mydata[j][1], j, 1, 6);
        Y : makelist (mydata[j][2], j, 1, 6);
(X)     [0.2,0.3,0.4,0.5,0.6,0.7]
(Y)     [0.599,0.896,1.189,1.479,1.756,2.044]
```

The Maxima function sin automatically distributes over lists and list1*list2 returns a list whose elements are the product of the respective elements of the given lists list1 and list2, assumed to be the same length. For example:

```
(%i138) sin (X);
        X*sin(X);
(%o137) [0.19867,0.29552,0.38942,0.47943,0.56464,0.64422]
(%o138) [0.039734,0.088656,0.15577,0.23971,0.33879,0.45095]
```

Looking at our definitions of A, B, ..., F, from above:
A = Σ(x_i)^2, B = Σ (x_i*sin(x_i), C = Σx_i*y_i, D = Σx_i*sin(x_i),
E = Σsin(x_i)*sin(x_i), F = Σy_i*sin(x_i),
we use Maxima's list arithmetic abilities to get:

```
(%i144) A : Lsum (X*X);
        B : Lsum (X*sin(X));
        C : Lsum (X*Y);
        D : B;
        E : Lsum (sin(X)*sin(X));
        F : Lsum (Y*sin(X));
(A)     1.39
(B)     1.3136
(C)     4.0881
(D)     1.3136
(E)     1.2421
(F)     3.8642
```

We can now get explicit solutions for a and b by using two single quotes in front of soln_a and in front of soln_b. The two single quotes force Maxima to evaluate each expression making use of the values of A, B, ..., F.

```
(%i146) a : ''soln_a;
        b : ''soln_b;
(a)     1.9625
(b)     1.0355
```

So we have found the best fit values of the parameters a and b in this example, using analytic methods. These values agree with those returned by lsquares_estimates.

7.3.4 Use and Abuse of the χ2 Value For this Nonlinear Fit

Since X² = (1/σy^2) * Σ [y_i - f (x_i)]^2 = (1/σy^2) * Σ [y_i - a*x_i - b*sin(x_i)]^2,
and we inferred σy = 0.001 for all the data points,

```
(%i147) Lsum ( ( Y - a*X - b*sin (X) )^2 ) / (0.001)^2;
(%o147) 52.7
```

This large value of χ2 for this nonlinear fit for a 4 degrees of freedom example serves as a warning about the use and abuse of judging the quality of fit for models which are not linear in the independent variable(s) using the value of χ2.
Here is the abstract of an archive paper relevant to this issue:
(If K denotes the number of degrees of freedom, "reduced χ2" is then defined by
χ2_red = χ2 / K.)

Dos and don'ts of reduced chi-squared
Rene Andrae, Tim Schulze-Hartung, Peter Melchior
http://arxiv.org/abs/1012.3754

"Reduced chi-squared is a very popular method for model assessment, model comparison, convergence diagnostic, and error estimation in astronomy. In this manuscript, we discuss the pitfalls involved in using reduced chi-squared. There are two independent problems: (a) The number of degrees of freedom can only be estimated for linear models. Concerning nonlinear models, the number of degrees of freedom is unknown, i.e., it is not possible to compute the value of reduced chi-squared. (b) Due to random noise in the data, also the value of reduced chi-squared itself is subject to noise, i.e., the value is uncertain.
This uncertainty impairs the usefulness of reduced chi-squared for differentiating between models or assessing convergence of a minimisation procedure. The impact of noise on the value of reduced chi-squared is surprisingly large, in particular for small data sets, which are very common in astrophysical problems. We conclude that reduced chi-squared can only be used with due caution for linear models, whereas it must not be used for nonlinear models at all. Finally, we recommend more sophisticated and reliable methods, which are also applicable to nonlinear models."

7.3.5 Parameter Errors via Analytic Methods

Let the common denominator of the symbolic solutions for a and b be Δ = B*D - A*E.
Δ is a function of the set of x_i's and does not depend on the values of the y_i.

We are using the values of A, B, ... found just above here.

```
(%i148) Δ : B*D - A*E;
(Δ)     −0.0010016
```

In the next section of this worksheet we derive the variance of a, V(a), as
V(a) = E*(A*E - B^2) *(σ/Δ)^2,
where σ is the assumed constant error of the y_i measurements.
From the given data we infer that σ_i = σ = 0.001.
Then
σa = sqrt (V(a)), or σa = sqrt (E*(A*E - B^2))*σ/|Δ|.

```
(%i149) σa : sqrt (E*(A*E - B^2))*0.001/abs(Δ);
(σa)    0.035216
```

So a = 1.96 +/- 0.04.

In a similar manner we can find a value for σb.

```
(%i150) σb : sqrt (A*(A*E - D^2))*0.001/abs (Δ);
(σb)    0.037253
```

So b = 1.04 +/- 0.04.

7.3.6 Analytic Parameter Error Derivations

We have used the results of this section in the section just above.

With
A = Σ(x_i)^2,
B = Σ (x_i*sin(x_i),
C = Σx_i*y_i,
D = Σx_i*sin(x_i),
E = Σsin(x_i)*sin(x_i),
F = Σy_i*sin(x_i),
Δ = B*D - A*E,

only F and C depend on the y_i values.
We also have
a = (B*F - C*E)/Δ = (1/Δ) * { B*Σ [sin(x_i) * y_i] - E*Σ(x_i*y_i) }
  = (1/Δ) * Σ [B*sin(x_i) - E*x_i] * y_i
  = Σ α(x_i) * y_i,
where α(x_i) = [B*sin(x_i) - E*x_i] / Δ.
Since errors add in quadrature, the variance of a is (σ is the constant σy)
V(a) = Σ α(x_i)^2 * σ^2 = (σ/Δ)^2 * Σ [B*sin(x_i) - E*x_i]^2.
Using the definitions, this simplifies to
  V(a) = (σ/Δ)^2 * E * (A*E - B^2).
Taking the square root,
σa = (σ/|Δ|) * [E*(A*E - B^2)]^(1/2).

In a similar manner,
$b = (1/\Delta)*[C*D - A*F]$
$= \Sigma\, \beta(x_i) * y_i$,
with $\beta(x_i) = (1/\Delta)* [D*x_i - A*\sin(x_i)]$.
Hence the variance of b is
$V(b) = \Sigma\, [\beta(x_i)]^2 * \sigma^2 = (\sigma/\Delta)^2 * [D*x_i - A*\sin(x_i)]^2$,
which simplifies to
$V(b) = (\sigma/\Delta)^2 * A * [A*E - D^2]$, so
$\sigma b = (\sigma/|\Delta|) * [A*(A*E - D^2)]^{(1/2)}$.

8 *Two Parameter Weighted Linear Least-Squares Fit*

8.1 Best Fit values of m and c in model y = m*x + c

We follow our reference Luca Lista (p. 115 ff) in this section where we find analytic formulas for m and c if each data point has its own value of σy.

Given data which includes a different value of σy for each (x,y) data point, in order to find the best (in the sense of least squares) fit line y = m*x + c, we need to find m and c such that, with

$\chi^2 = \Sigma\, [(y_i - m*x_i - c) / \sigma_i]^2$, $\partial\chi^2/\partial m = 0$ and $\partial\chi^2/\partial c = 0$.

The equation $\partial\chi^2/\partial c = 0$ implies

$\Sigma\, (y_i - m*x_i - c) / (\sigma_i)^2 = 0$ (1)

Let $K = \Sigma\, (1/(\sigma_i)^2)$, (2)
a known constant.

Define weights

$w_i = [1/(\sigma_i)^2] / K$. (3)

Note that the sum of the weights is unity,

$\Sigma\, w_i = 1$. (4)

Define the symbol <x> by

$\langle x\rangle = \Sigma\, w_i * x_i$, (5)

and the symbol <y> by

$\langle y\rangle = \Sigma\, w_i * y_i$. (6)

Then note that the above definitions imply

$1/(\sigma_i)^2 = K * w_i$ (7)

In terms of these definitions, we can rewrite the consequence of $\partial\chi^2/\partial c = 0$ as

$c = \langle y\rangle - m * \langle x\rangle$. (8)

We further define the symbols <xy> and <x^2> as

<xy> = Σ w_i * x_i * y_i , (9)

<x^2> = Σ w_i * (x_i)^2. (10)

The equation ∂χ²/∂m = 0 implies

Σ (x_i * y_i - m * (x_i)^2 - c * x_i) / (σ_i)^2 = 0, or using (7),

Σ (K * w_i) * (x_i * y_i - m * (x_i)^2 - c * x_i) = 0, or, dividing by the constant K,

Σ w_i * x_i * y_i - m * Σ w_i * (x_i)^2 - c * Σ w_i * x_i = 0,

or using our above definitions,

<xy> - m * <x^2> - c * <x> = 0. (11)

Substituting for c using Eqn (8), we get

<xy> - m * <x^2> - (<y> - m * <x>) * <x> = 0, or

<xy> - m * <x^2> - <x> * <y> + m * <x>^2 = 0.

Solving for m we get

m = [<xy> - <x>*<y>] / [<x^2> - <x>^2] (12).

In our code, we first use (12) to find m and then use (8) to find c.

8.2 Solving for uncertainties σm and σc

Luca Lista argues that the uncertainties σm and σc can be found from the equations:

1/σm^2 = (1/2) * ∂²χ²/∂m² , and (13)

1/σc^2 = (1/2) * ∂²χ²/∂c² . (14)

Eqn (14) implies (using symbols defined above)

1 / σc^2 = K, or σc = 1 / K^(1/2) . (15)

Eqn (13) implies

1 / σm^2 = K * <x^2), or σm = 1 / [K*<x^2>]^(1/2). (16)

8.3 WLS2 (xysData)

WLS2 (xysData) returns the list [m, σm, c, σc].

```
(%i151) WLS2  (xys) :=
          block ([ nn, mm, cc, XX,YY, Xav, Yav, XY_av, Xsq_av,SS,
             W, K, sm, sc, Chisq, sm2 ],
           nn : length (xys),
           XX : makelist (xys[j][1], j, 1, nn),
           YY : makelist (xys[j][2], j, 1, nn),
           SS : makelist (xys[j][3], j, 1, nn),
           K : Lsum (1 / SS^2),
           W : (1 / SS^2) / K,
           Xav : Lsum ( W*XX),
           Yav : Lsum (W*YY),
           XY_av : Lsum (W*XX*YY),
           Xsq_av : Lsum (W*XX^2),
           mm : (XY_av - Xav*Yav) / (Xsq_av - Xav^2),
           cc : Yav - mm*Xav,
           if cc = 0 then print (sconcat (" y = ", mm, "*x "))
                else if cc > 0 then print (sconcat (" y = ",mm,"*x + ", cc))
                else  print (sconcat (" y = ",mm,"*x - ", abs (cc))),
           sc : sqrt (1/K ),
           sm : 1 / sqrt ( K * Xsq_av),
           print (" σm = ", sm, " σc  = ", sc),
           print (" χ2 = ", Lsum ( ( (YY - mm*XX - cc) / SS )^2 )  ),
           [mm, sm, cc, sc] )$
```

8.4 An Example Problem using WLS2 (xysData)

We start with some dummy data in the form of a list in which each sublist has the form [x, y, σy].

```
(%i152) given : [ [3.103, 5.211, 0.392], [6.273, 8.070, 0.470], [8.738, 8.686, 0.588],
           [12.656, 10.138, 0.784], [15.079, 13.939, 1.176], [18.382, 12.642, 2.353] ];
(given) [[3.103,5.211,0.392],[6.273,8.07,0.47],[8.738,8.686,0.588],[
        12.656,10.138,0.784],[15.079,13.939,1.176],[18.382,12.642,2.353]]

(%i153) mydata1 : makelist ([given[j][1], given[j][2]], j, 1, 6);
(mydata1) [[3.103,5.211],[6.273,8.07],[8.738,8.686],[12.656,10.138],[
        15.079,13.939],[18.382,12.642]]
```

We first use OLS2 (mydata1), which produces estimates of m, σm, c, σc based on the data derived estimate of σy.

```
(%i154) [m1, σm1, c1, σc1] : OLS2 (mydata1);
        R^2 =  5.9961
        data derived estimate of σy =  1.2243
        y = 0.5229*x + 4.1832
        we used sigma−y =  1.2243
        σm =  0.095846    σc =  1.1413
        χ^2 =  4.0
(%o154) [0.5229,0.095846,4.1832,1.1413]

(%i155) [m2, σm2, c2, σc2] :  WLS2 (given);
        y = 0.58688*x + 3.6848
        σm =  0.032484   σc  =  0.2465
        χ2 =  6.4308
(%o155) [0.58688,0.032484,3.6848,0.2465]
```

Since the data list given includes estimated values of σy for each data point, the values of the line parameters returned by WLS2 (xysData) should be a better fit to the data.

Two thirds of the values of χ2 should lie in the range described by mean +/- 1 standard dev. For the χ2 distribution describing a data set with n degrees of freedom, the mean of the distribution is n and one standard deviation is sqrt (2*n).

We have n = 4 degrees of freedom (6 - 2).

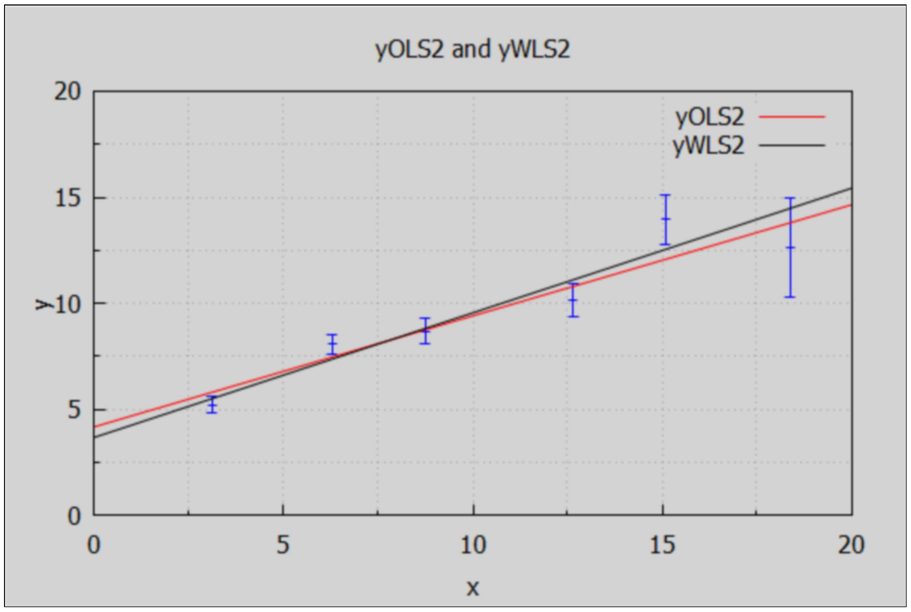
```
(%i157) mean_chi2(4);
        std_chi2(4), numer;
(%o156) 4
(%o157) 2.8284
```

Two thirds of the values of χ2 should lie in the range described by 4 +/- 2.8.

The χ2 value of 6.4 lies within the range [1.2, 6.8] and is acceptable so we can trust the values of m and c found.

```
(%i158) wxdraw2d ( xrange = [0,20], yrange = [0, 20], xlabel = "x",
          ylabel = "y",   title = "yOLS2 and yWLS2", grid = [2,2],
        background_color = light_gray, line_width = 1, errors (given),
        color = red, key = "yOLS2", explicit (m1*x + c1, x, 0, 20) ,
        color = black, key = "yWLS2", explicit (m2*x + c2, x, 0, 20) )$
```

(%t158)



For the first data point (3.103, 5.211), which has the smallest error bars, the line yWLS2 is closer to the data point than the line yOLS2. Because our first point has the smallest error bars, we want our line to do a really good job of fitting the first point. We are ok if we are a little further away from the last point, which has the largest error bars.

9 Barlow Prob. 6.5, Half-Life of a Radioactive Source

A decaying radioactive source is observed with a Geiger counter. Readings are taken for a short period (1 minute) at hourly intervals. The number of counts measured is as follows—use them to find the half-life. The half-life τ is the time needed for the count rate to decrease by a factor of 2: N(t + τ) = N(t)/2.

The list given has sublists [t, N], with t in hours and N in counts/minute.

```
(%i159) given : [ [0, 997], [1, 520 ], [2, 265], [3, 127], [4, 70], [5, 35], [6, 16], [7, 7], [8, 3] ];
(given) [[0,997],[1,520],[2,265],[3,127],[4,70],[5,35],[6,16],[7,7
        ],[8,3]]
```

With an array of Geiger counters surrounding the radioactive source, for example, each detector observes a slightly different value for the counts/minute at t = 0, and also at t = 1 hr, and so on. The number of counts/min is a discrete random variable which takes values which are non-negative integers only (not drawn from a continuum of values), and the probability of finding exactly k counts in 1 minute if λ is the mean number of counts in 1 minute is given by the probability distribution function P(k, λ) = λ^k * exp(-λ) / k!, and this value is produced by the Maxima function pdf_poisson (k, λ). The Maxima function mean_poisson (λ) ---> λ. We have discussed the (discrete) Poisson distribution in Statistics with Maxima, Ch. 3, Stat03-Poisson.wxmx.

9.1 Review of Poisson Statistics

Quoting Luca Lista in 'Statistical Methods for Data Analysis in Particle Physics', Lecture Notes in Physics 909:

"A discrete variable k is said to be a Poissonian variable if it obeys the Poisson distribution defined below for a given value of the parameter λ = mean number of occurrences in a specified time interval, k = measured number of occurrences in the specified time interval:
P(k; λ) = λ^k * exp (- λ)/k! "
The numbers k and λ are dimensionless in this definition.
The numerical value of the Maxima function pdf_poisson (k, λ) returns P(k; λ), the probability of measuring exactly k occurrences when the mean number is λ. The Maxima function mean_poisson (λ) --> λ, the mean number of occurrences.

For example,suppose the mean number of occurrences is 15

```
(%i160) mean_poisson (15);
(%o160) 15
```

"For example, suppose we want to know how likely it will be for k = 10 molecules to arrive over a 1μs interval [of time] at the surface of a sensor." If the average rate r has been measured to be r = 15 molecules/μs, and we let t = 1μs, then λ = r * t = (15 molecules/μs) * 1 μs = 15 molecules = average count over 1 μs. We then use the poisson probability distribution to determine the probability P(10, 15) of actually finding k = 10 molecules arriving in 1 μs in a given measurement:

```
(%i162) (15^10) * exp (- 15) / factorial (10), numer;
        pdf_poisson (10, 15), numer;
(%o161) 0.048611
(%o162) 0.048611
```

Then given that the probability is 0.0486, then the chance of measuring exactly 10 molecules arriving in a given micro-second is about 5%

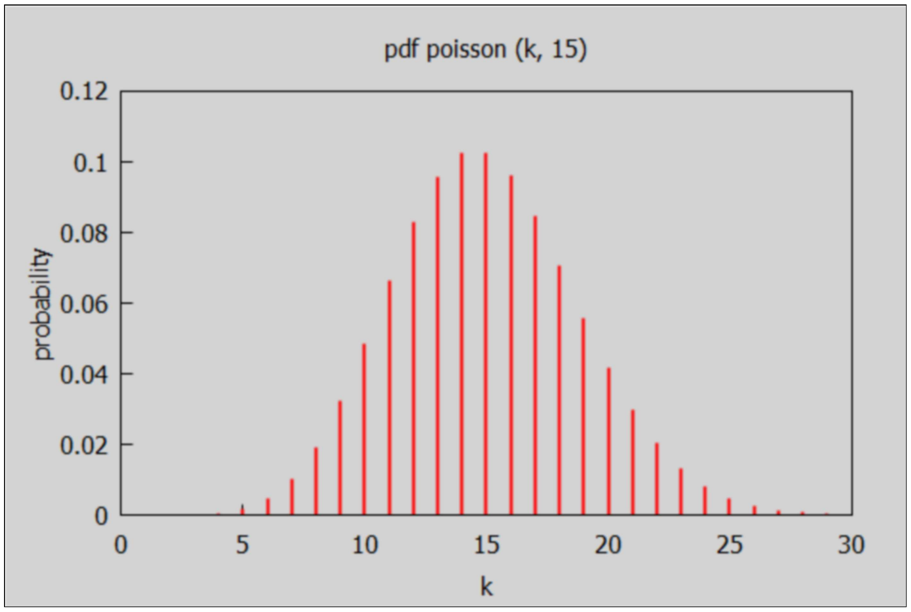
We can show the theoretical Poisson distribution for the case λ = 15, by making a list of the values returned by pdf_poisson with different values of k.

```
(%i164) mypoints : makelist ( [k, float (pdf_poisson (k, 15))] , k, 1, 30)$
        fll (mypoints);
```

(%o164) $[[1,4.5885\ 10^{-6}],[30,2.2114\ 10^{-4}],30]$

```
(%i165) wxdraw2d ( xrange = [0, 30], yrange = [0, 0.12], background_color = light_gray,
         points_joined = impulses,  title = "pdf poisson (k, 15)",
         xlabel = "k", ylabel = "probability",  color = red, line_width = 2,
         points (mypoints) )$
```

(%t165)



From this plot you can see that the theoretical probability of measuring exactly 10 occurrences in the specified time interval is about 0.5, implying a 5% chance.

"The width of the discrete Poisson distribution is characterized by the standard deviation σ. Roughly two-thirds of all values of counts per minute recorded by this hypothetical set of Geiger counters at any specific time t will lie in the interval [λ - σ, λ + σ]. For the Poisson distribution, σ = sqrt (λ) = 'standard deviation of the Poisson distribution' " and we can use the Maxima function std_poisson (λ) ---> sqrt(λ).

In the absence of knowledge of the true mean λ, we can use sqrt(N) as the uncertainty of our measurement N, and quote the result as N +/- sqrt(N).

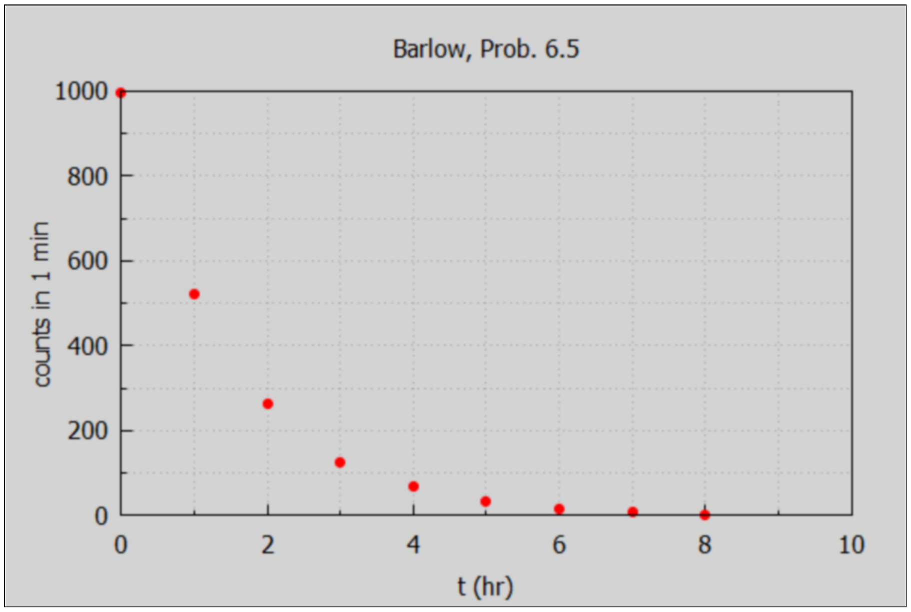
9.2 Theoretical Radioactive Decay (Counts/min) Curve

The theory of radioactive decay predicts N(t) = N(0)*exp (- r*t), an 'exponential' decay curve. The constant r is the rate of decay and N is a non-negative integer. The produce r*t is dimensionless, so the units of r are the inverse of the units of t.

From the definition of the half-life τ, N(t+τ) = N(t)/2 for any t, we conclude τ = r / ln (2).

We plot the given data, counts per minute, measured every hour, in the list given.

```
(%i166) wxdraw2d (ylabel = "counts in 1 min", xlabel = "t (hr)", xrange = [0, 10], yrange = [0, 1000],
          title = "Barlow, Prob. 6.5", background_color = light_gray, grid = [2,2],
         point_type = 7, point_size = 1, color = red, points (given) )$
```

(%t166)


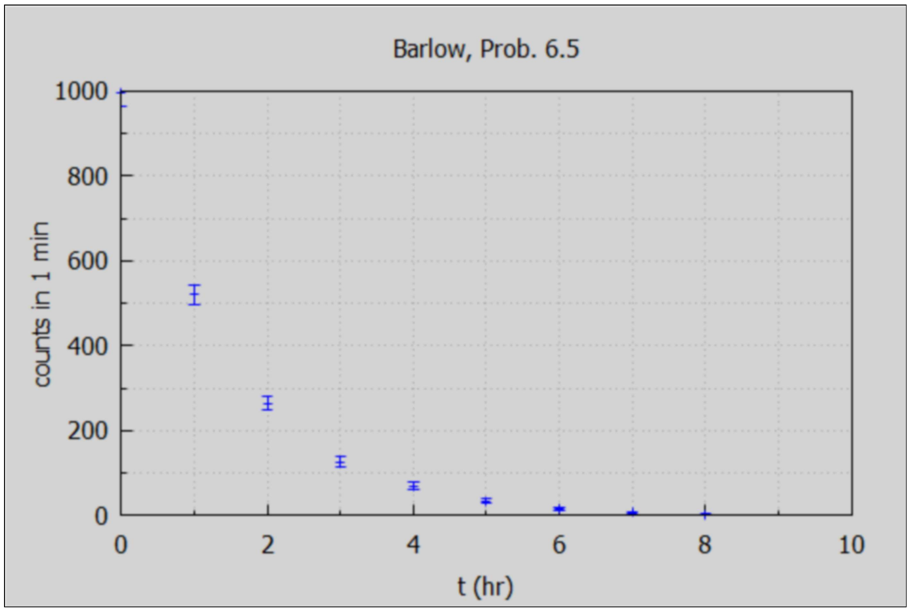
From this curve we see that the half life is about one hour.

We can add vertical error bars, different for each data point, using σN ~ N^(1/2).

```
(%i167) errorL : makelist ([given[j][1], given[j][2], float (sqrt (given[j][2])) ], j, 1, length (given) );
(errorL) [[0,997,31.575],[1,520,22.804],[2,265,16.279],[3,127,11.269
         ],[4,70,8.3666],[5,35,5.9161],[6,16,4.0],[7,7,2.6458],[8,3,1.7321]]
```

```
(%i168) wxdraw2d (ylabel = "counts in 1 min", xlabel = "t (hr)", xrange = [0, 10], yrange = [0, 1000],
         title = "Barlow, Prob. 6.5", background_color = light_gray, grid = [2,2],
        line_width = 1, errors (errorL) )$
```

(%t168)



In Maxima, log(x) means the natural log of x, ln (x).

```
(%i170) log (exp (x));
        diff (log(x), x);
```

(%o169) x

(%o170) $\frac{1}{x}$

9.3 Fitting a Straight Line to (t, ln(N)) Points

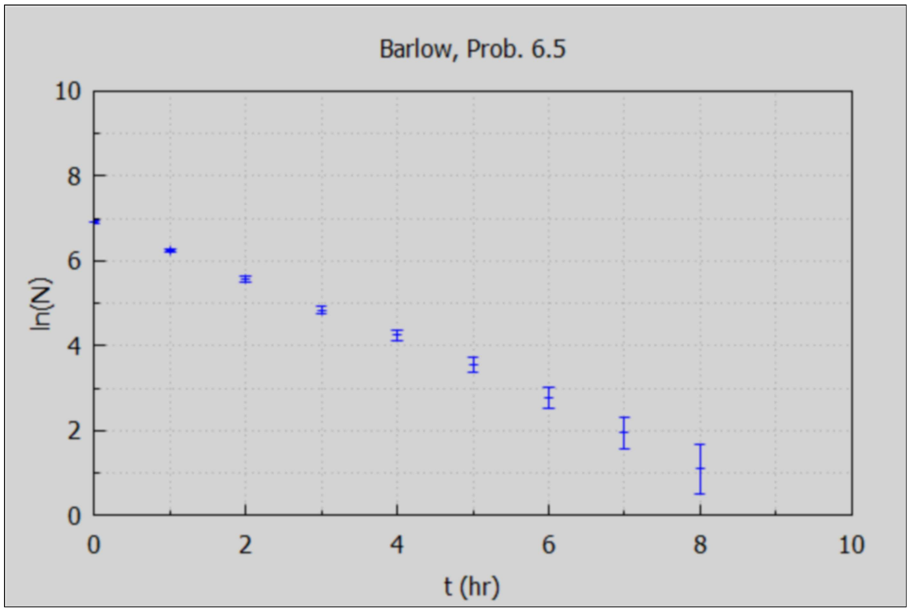
We can find something to fit to a straight line if we plot the natural log of N vs the time t. N = No*exp (-r*t) implies ln(N) = ln(No) - r*t. With y = ln(N), we evaluate σy using σy = |∂y/∂N| * σN = (1/N) * N^(1/2) = 1/N^(1/2). We then define a new error list appropriate to our straight line fit.

```
(%i171) errorL : makelist ([given[j][1], float ( log (given[j][2] )), float (1/sqrt(given[j][2])) ],
                     j, 1, length (given) );
(errorL) [[0,6.9048,0.03167],[1,6.2538,0.043853],[2,5.7797,0.06143],[
         3,4.8442,0.088736],[4,4.2485,0.11952],[5,3.5553,0.16903],[6,2.7726,
         0.25],[7,1.9459,0.37796],[8,1.0986,0.57735]]
```

and plot the points with error bars:

```
(%i172) wxdraw2d (ylabel = " ln(N) ", xlabel = "t (hr)", xrange = [0, 10], yrange = [0, 10],
         title = "Barlow, Prob. 6.5", background_color = light_gray, grid = [2,2],
        line_width = 1, errors (errorL) )$
```

(%t172)



9.3.1 Using OLS2 (xyData) for Straight Line Fit to (t, ln(N)) Points

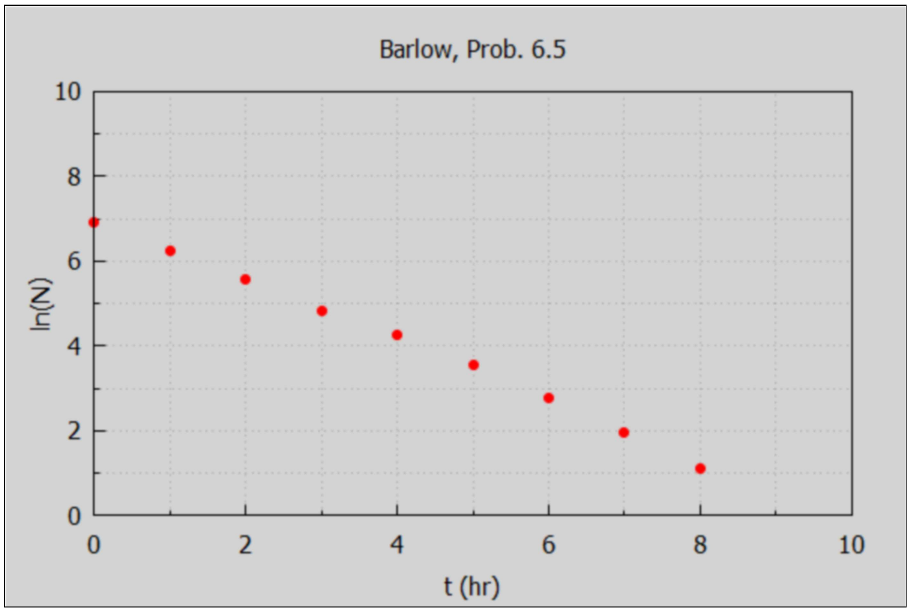
To use ordinary least squares (OLS), we create a list of [t, ln(N)] sublists.

```
(%i173) mydata : makelist ([errorL[j][1], errorL[j][2]], j, 1, length (errorL) );
(mydata) [[0,6.9048],[1,6.2538],[2,5.5797],[3,4.8442],[4,4.2485],[5,
         3.5553],[6,2.7726],[7,1.9459],[8,1.0986]]
```

We can show the points in the list mydata, ignoring errors bars.

```
(%i174) wxdraw2d (ylabel = " ln(N) ", xlabel = "t (hr)", xrange = [0, 10], yrange = [0, 10],
          title = "Barlow, Prob. 6.5", background_color = light_gray,
           line_width = 2,  grid = [2,2],
         point_type = 7, point_size = 1, color = red, points (mydata) )$
```

(%t174)



If we do an unweighted fit (ordinary least squares), we can use OLS2 (xydata).
We are going to fit the transformed raw data (t, ln (N)) to the equation
  y = m*x + c,
with y = ln (N), x = t, m = - r, and c = ln (No)

```
(%i175) [m1, σm1, c1, σc1] : OLS2( mydata);
  R^2 =  0.077509
  data derived estimate of σy =  0.10523
  y = −0.71752*x + 7.0038
  we used sigma−y =  0.10523
  σm =  0.013585   σc =  0.064676
  χ^2 =  7.0
(%o175) [−0.71752,0.013585,7.0038,0.064676]
```

With 9 data points and two parameters (m, c), the number of degrees of freedom is
n = 9 - 2 = 7.

Two thirds of the values of X^2 should lie in the range described by mean +/- 1 standard dev.
For the X^2 distribution describing a data set with n degrees of freedom, the mean of the distribution is n and one standard deviation is sqrt (2*n).

We have n = 7 degrees of freedom (9 - 2).

```
(%i177) mean_chi2(7);
        std_chi2(7), numer;
(%o176) 7
(%o177) 3.7417
```

Two thirds of the values of X² should lie in the range described by 7 +/- 3.7.

The X² value of 7 found lies within the range [3, 10.7] and is acceptable so we can trust the values of m and c found using the OLS fit.

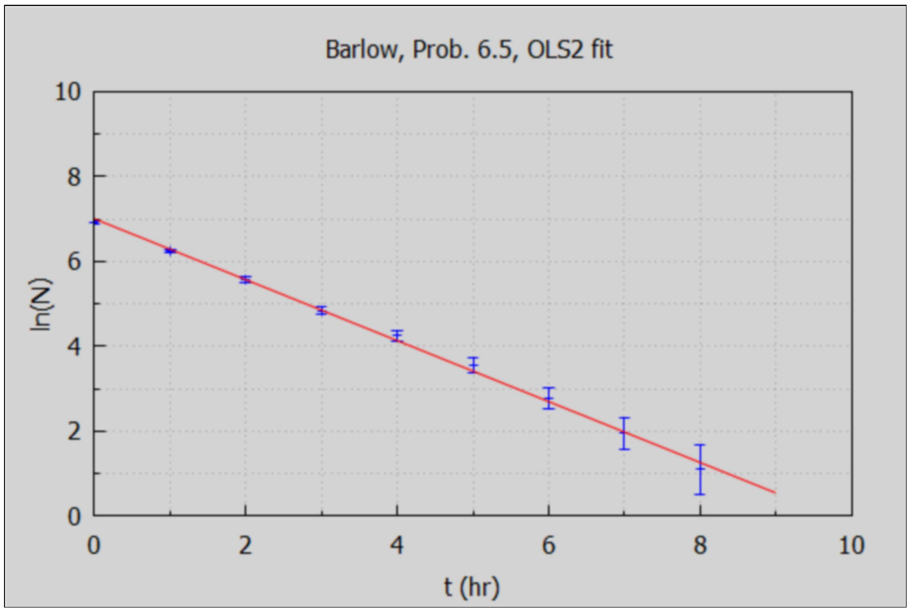
OLS2 (xydata) asserts m1 = - r = - 0.71752, c1 = ln (No) = 7.0038. Here r has units 1/hr.

Using exp (ln(A)) = A, we can find No from c. No has units counts/min.
c = ln (No) ==> exp(c) = exp [ln (No)] = No.

```
(%i179) c1;
        No : exp (c1);
(%o178) 7.0038
(No)    1100.8
```

We show both the points and the OLS2 (xydata) best fit line.

```
(%i180) wxdraw2d (ylabel = " ln(N) ", xlabel = "t (hr)", xrange = [0, 10], yrange = [0, 10],
     title = "Barlow, Prob. 6.5, OLS2 fit ", background_color = light_gray,
     line_width = 1,  grid = [2,2], errors (errorL),
     color = red, explicit (m1*x + c1, x, 0, 9) )$
```

(%t180)


The half-life τ = ln (2)/ r, the time for an inital count rate to drop by one half,

```
(%i181) τ1 : log(2)/(- m1), numer;
(τ1)    0.96603
```

If we trust the value of σm1, we can estimate dτ = |∂τ/∂m1|*σm1 = ln(2)*σm1/(m1)^2

```
(%i182) στ1 : log(2)*σm1/m1^2, numer;

(στ1)   0.01829
```

Then τ1 = (0.97 +/- 0.02) hr, is the estimate of the half-life using ordinary least squares (OLS2).

9.3.2 Using WLS2 (xysData) for fit

This should be more accurate since we take into account a different weighting factor for each data point.

```
(%i183) mydata;
(%o183) [[0,6.9048],[1,6.2538],[2,5.5797],[3,4.8442],[4,4.2485],[5,
        3.5553],[6,2.7726],[7,1.9459],[8,1.0986]]

(%i184) errorL;
(%o184) [[0,6.9048,0.03167],[1,6.2538,0.043853],[2,5.5797,0.06143],[
        3,4.8442,0.088736],[4,4.2485,0.11952],[5,3.5553,0.16903],[6,2.7726,
        0.25],[7,1.9459,0.37796],[8,1.0986,0.57735]]
```

The list errorL has sublists of the form [t, ln(N), 1/N^(1/2)] = [x, y, σy] for each point, in which x = t, y = ln(N), σy = 1/N^(1/2). Each point has its own value for σy and we use a method in which each point has a separate weight w_i.

```
(%i185) [m2, σm2, c2, σc2] : WLS2 (errorL);
         y = −0.6789*x + 6.9157
         σm =  0.013098   σc  =  0.02214
         χ2 =  1.6109
(%o185) [−0.6789,0.013098,6.9157,0.02214]
```

The chisq value of 1.6 is outside the acceptable range [3, 10.7].

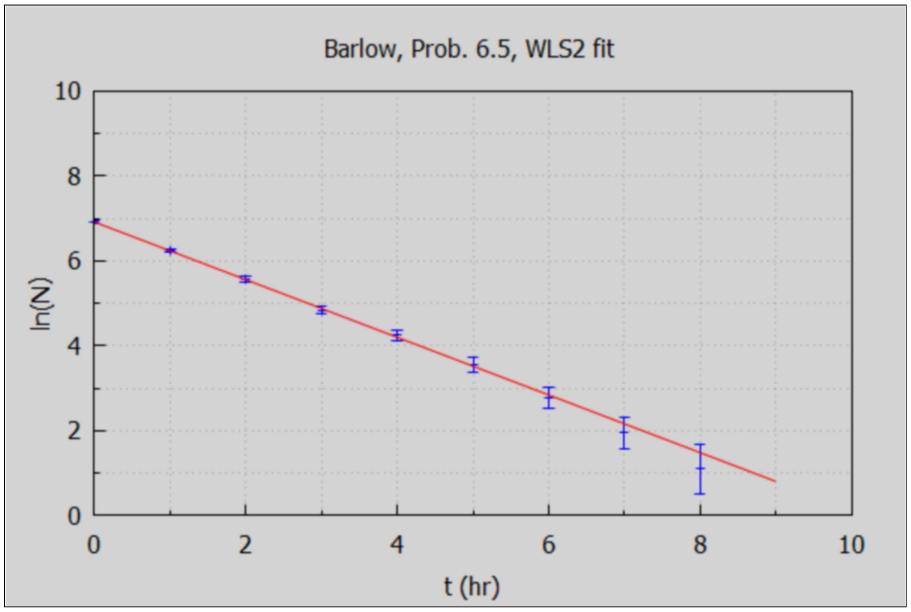
This value of χ² is not in the acceptable range. The data should be retaken with more care.

Since τ = ln(2)/r, στ = |∂τ/∂r| * σr = [ln(2)/ (m2)^2] * σm2

```
(%i186) [m2, σm2];
(%o186) [−0.6789,0.013098]

(%i187) τ2 : log(2)/abs(m2), numer;
(τ2)    1.021

(%i188) στ : log(2)*σm2/m2^2, numer;
(στ)    0.019698
```

This WLS2 fit produces the estimate of the half-life as
τ = (1.02 +/- 0.02) hr. Barlow has (1.03 +/- 0.02) hr as an answer.

```
(%i189) wxdraw2d (ylabel = " ln(N) ", xlabel = "t (hr)", xrange = [0, 10], yrange = [0, 10],
    title = "Barlow, Prob. 6.5, WLS2 fit ",background_color = light_gray,
    line_width = 1,  grid = [2,2], errors (errorL),
    color = red, explicit (m2*x + c2, x, 0, 9) )$
```

(%t189)


10 One Parameter Weighted Linear Fit y = m*x + C, given C

We seek to find the best-fit value of the parameter m, by minimizing
χ² = Σ [(y_i - m*x_i - C) / σ_i]^2 with respect to m, with C given.

Using 1/(σ_i)^2 = K*w_i as in our derivation of the weighted two parameter fit y = m*x + c above, ∂χ²/∂m = 0 then implies:

Σ [x_i*y_i - m* (x_i)^2 - C*x_i] / (σ_i)^2 = 0, or
Σ K*w_i * x_i*y_i - m* Σ K*w_i* (x_i)^2 - C* Σ K*w_i*x_i = 0, or
<xy> - m*<x^2> - C* <x> = 0, or
m*<x^2> = <xy> - C*<x>, and finally

m = (<xy> - C*<x>) / <x^2>.

We use
1/σm^2 = (1/2) * ∂²χ²/∂m² to find σm.

χ² = Σ K*w_i* (y_i - m*x_i - C)^2,
∂χ²/∂m = -2*K* Σ w_i * (x_i*y_i - m*(x_i)^2 - C),
∂²χ²/∂m² = 2*K*Σw_i*(x_i)^2 = 2*K*<x^2),
1/σm^2 = (1/2) * ∂²χ²/∂m² = K*<x^2),
σm = 1 / [K*<x^2>]^(1/2).

10.1 Example with C = 1

An example from K. K. Gan is to fit some (x_i, y_i, σ_i) data to the equation y = m*x + 1, finding the best fit value for m.

```
(%i190) mydata : [ [1.0, 2.2, 0.2], [2.0, 2.9, 0.4], [3.0, 4.3, 0.3], [4.0, 5.2, 0.1] ];
(mydata) [[1.0,2.2,0.2],[2.0,2.9,0.4],[3.0,4.3,0.3],[4.0,5.2,0.1]]
```

To find the best fit value of m, we need to calculate <xy>, <x> and <x^2>.

```
(%i193) X : makelist ( mydata[j][1], j, 1, 4);
        Y : makelist ( mydata[j][2], j, 1, 4);
        S : makelist ( mydata[j][3], j, 1, 4);
(X)     [1.0,2.0,3.0,4.0]
(Y)     [2.2,2.9,4.3,5.2]
(S)     [0.2,0.4,0.3,0.1]
```

```
(%i201) K : Lsum (1 / S^2);
        W : (1 / S^2) / K;
        Xav : Lsum ( W*X);
        Yav : Lsum (W*Y);
         XY_av : Lsum (W*X*Y);
         Xsq_av : Lsum (W*X^2);
         m : (XY_av - Xav) / Xsq_av ;
        sig_m : 1/sqrt(K*Xsq_av);
(K)      142.36
(W)      [0.17561,0.043902,0.078049,0.70244]
(Xav)    3.3073
(Yav)    4.502
(XY_av)  16.259
(Xsq_av) 12.293
(m)      1.0536
(sig_m)  0.023905
```

Having tested a path to the answer "by hand", we write a Maxima function to do the routine work.

10.2 WLS1 (xysData, C) to fit y = m*x + C, with C specified

WLS1 (xysData, C) returns the list [m, σm].

```
(%i202) WLS1  (xys, C) :=
          block ([ nn, mm,  XX,YY, Xav, XY_av, Xsq_av,SS, W, K, sm ],
            nn : length (xys),
            XX : makelist (xys[j][1], j, 1, nn),
            YY : makelist (xys[j][2], j, 1, nn),
            SS : makelist (xys[j][3], j, 1, nn),
            K : Lsum (1 / SS^2),
            W : (1 / SS^2) / K,
            Xav : Lsum ( W*XX),
            XY_av : Lsum (W*XX*YY),
            Xsq_av : Lsum (W*XX^2),
            mm : float( (XY_av - C*Xav) / Xsq_av),
            if C = 0 then print (sconcat (" y = ", mm, "*x "))
                 else if C > 0 then print (sconcat (" y = ",mm,"*x + ", C))
                 else  print (sconcat (" y = ",mm,"*x - ", abs (C))),
            sm : float (1 / sqrt ( K * Xsq_av)),
            print (" σm = ", sm ),
            print (" χ2 = ", float ( Lsum ( ( (YY - mm*XX - C) / SS )^2 ) ) ),
            [mm, sm ] )$
```

```
(%i203) [m, sm] : WLS1 (mydata, 1);
        y = 1.0536*x + 1
        σm =   0.023905
        χ2 =   1.0402
(%o203) [1.0536,0.023905]
```

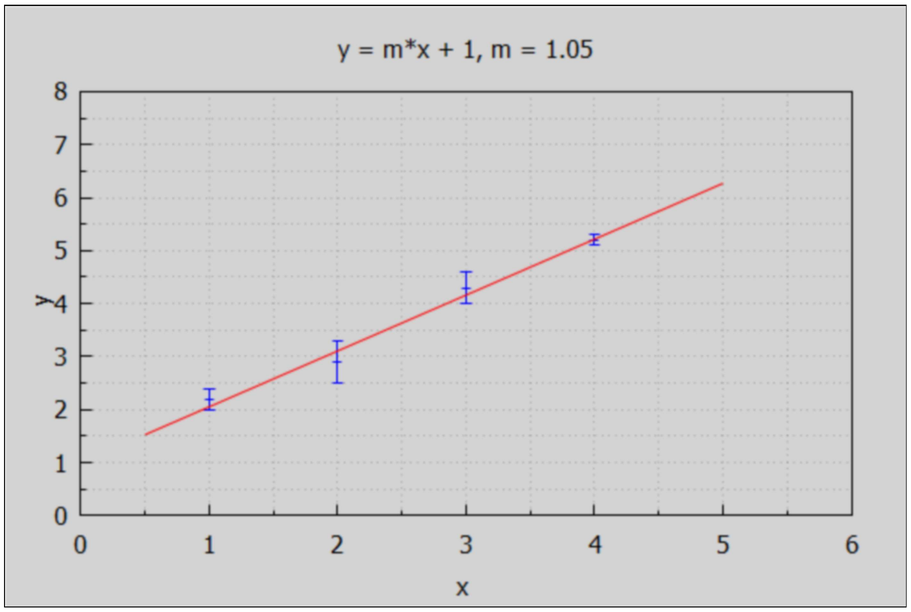
The best fit value of m is m = 1.05 +/- 0.02.

With 4 data points and one parameter, the number of degrees of freedom (dof) is 3.

```
(%i205) mean_chi2(3);
        std_chi2(3), numer;
(%o204) 3
(%o205) 2.4495
```

The X² value of 1.04 lies within the acceptable range [0.55, 5.4] for three degrees of freedom and is acceptable. Hence we can trust the result.

```
(%i206) wxdraw2d( xlabel = "x ", ylabel = " y", xrange = [0, 6], yrange = [0, 8],
            error_type = 'y, line_width = 1,background_color = light_gray,
          grid = [2,2],  errors( mydata ), title = "y = m*x + 1, m = 1.05",
             color = red, explicit (m*x + 1, x, 0.5, 5) )$
(%t206)
```



11 Appendix: Greek symbols, superscripts,draw2d options

σ   χ2   Δ χ^2
σ from Esc, s, Enter.
χ from Esc, c, Enter
Δ from Esc, Shift + d, Enter
Σ from Esc, Shift + s, Enter
To get ∂ symbol (partial derivative symbol), use Esc, p, then down-key once to 'partial', then press enter.

To get small superscript ^{2}, press Esc, then down-key past Greek symbols to ^2,
or page-down 6 times to get to ^2.

R^2 $\partial F/\partial x$ $\partial^2 F/\partial x^2$

useful draw2d options:
background_color = light_gray, line_width = 2,
point_type = filled_circle, grid = [2,2], point_size = 0.6