

To investigate fitting an exponential function to field data, first set up a function to generate a normal distribution with mean u , and standard deviation s :

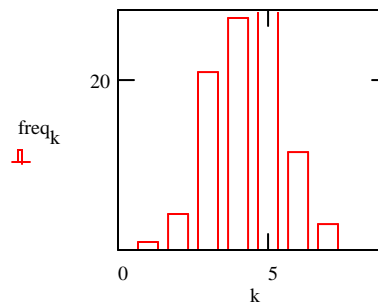
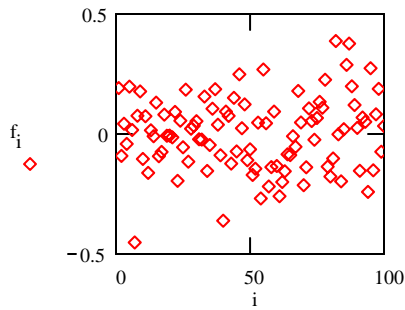
ORIGIN≡1 $\text{NORM}(u,s) := u + s \cdot \sqrt{-2 \cdot \ln(\text{rnd}(1))} \cdot \cos(2 \cdot \pi \cdot \text{rnd}(1))$ $i := 1..100$ $f_i := \text{NORM}(0,.15)$ $k := 1..8$

Make some bins for a histogram:

$$\text{bin}_k := \min(f) + (k - 1) \cdot \left(\frac{\max(f) - \min(f)}{7} \right)$$

$\text{freq} := \text{hist}(\text{bin}, f)$ $\text{freq}_{\text{last}(\text{freq})+1} := 1$

frequency table:



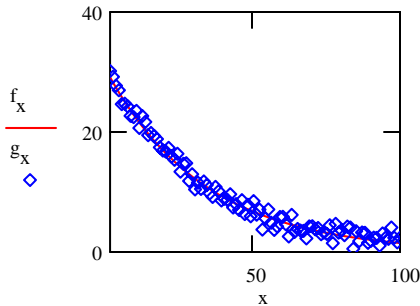
freq =	2	k
	5	1
	21	2
	27	3
	28	4
	12	5
	4	6
	1	7

Now generate some Gaussian noise around an exponential function.

$x := 1, 2..100$ $b := -.03$ $A := 30$ $f_x := A \cdot e^{b \cdot x}$ $g_x := f_x + \text{NORM}(f_x, 1) - f_x$ $\min(g) = 0.521$

Here's the function and noisy data:

And here's the setup to look at residuals:

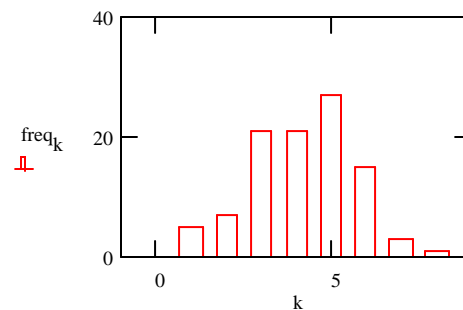
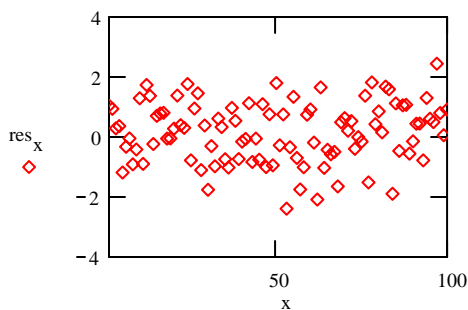


$$\text{res}_x := g_x - f_x$$

$$\text{bin}_k := \min(\text{res}) + (k - 1) \cdot \frac{\max(\text{res}) - \min(\text{res})}{7}$$

$\text{freq} := \text{hist}(\text{bin}, \text{res})$

$\text{freq}_{\text{last}(\text{freq})+1} := 1$



Now suppose we have the data $(g(x))$ and expect them to be from some non-linear exponential physical process like: Ae^{bx} . We want to invert the data to find the parameters of the exponential model, A & b . One approach is to take the natural log of $g(x)$ to linearize the non-linear function. Then we use classic least squares to find the best-fit line in $\ln\text{-}\ln$ space and then transform back to the $g(x)$ - x space. Here are some specific values for this example:

$x := 1, 2, \dots, 100$ $j_x := x$ $k_x := 1$ $i := \text{augment}(j, k)$ (This makes a $100 * 2$ array from j and k)

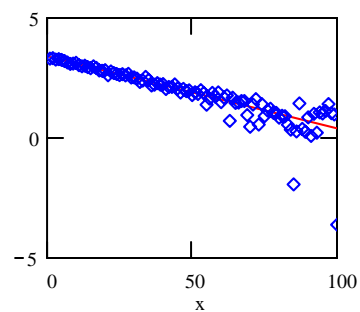
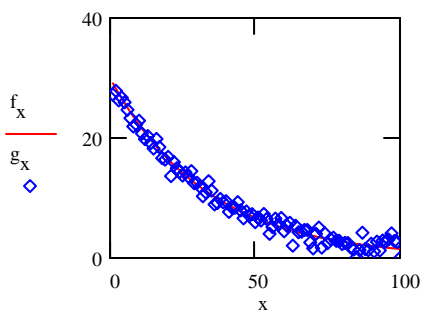
$b := -.03$ $A := 30$ $f_x := A \cdot e^{b \cdot x}$ $g_x := f_x + \text{NORM}(f_x, 1) - f_x$ $\text{min}(g) = 0.027$

To set this up as a linear inverse problem use these transforms: $A1 = \ln(A)$, $f2 = \ln(f(x))$, and $g2 = \ln(g(x))$. Now, using the rule $\ln(ax) = \ln(a) + \ln(x)$, write the model as a linear equation: $g2(x) = A1 + bx$, and solve it with standard least squares technique.

$$g2_x := \ln(g_x) \quad f2_x := \ln(f_x)$$

Original data and function:

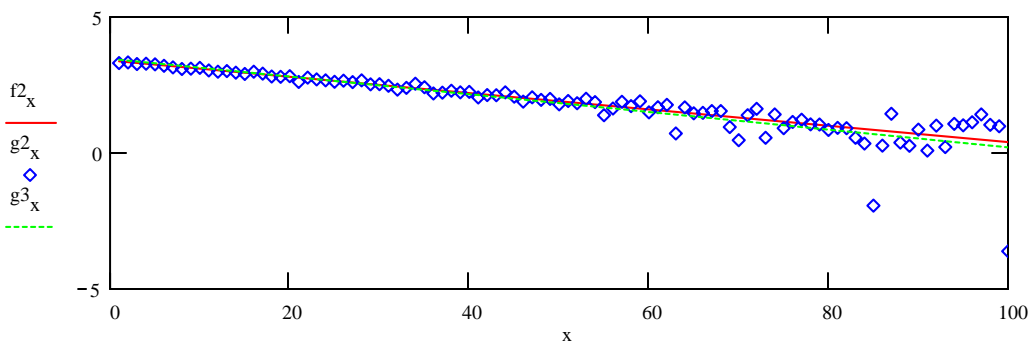
Here are the transformed data and function, note how the errors increase with x :



For the inversion use $i(x)$, we need to solve for A and b :

$$\begin{bmatrix} b \\ A1 \end{bmatrix} := (i^T \cdot i)^{-1} \cdot i^T \cdot g2 \quad \begin{bmatrix} b \\ A1 \end{bmatrix} = \begin{bmatrix} -0.033 \\ 3.461 \end{bmatrix} \quad \exp(A1) = 31.855 \quad g3_x := A1 + b \cdot x$$

Now plot up our least-squares line in \ln -linear space:



Now compare the errors between the original, transformed, function $f2(x)$ and the least squares fit to the transformed data, $g3(x)$. The least squares line, derived from the transformed data fits "better" than the original function.

$$\text{errf2} := \left[\sum_x (g_{2x} - f_{2x})^2 \right]^{\frac{1}{2}} \quad \text{errf2} = 5.587$$

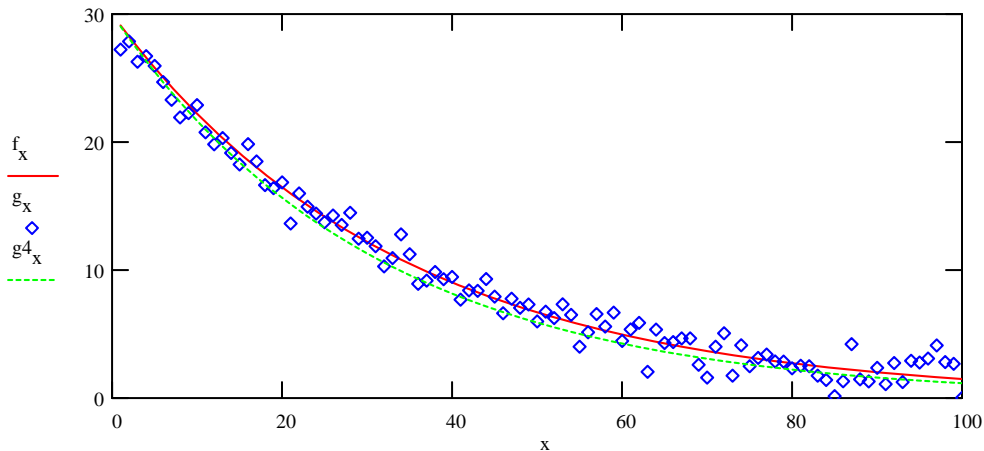
$$|\text{errf2}| = 5.587$$

$$\text{errf3} := \left[\sum_x (g_{2x} - g_{3x})^2 \right]^{\frac{1}{2}} \quad \text{errf3} = 5.496$$

$$|\text{errf3}| = 5.496$$

Now do the inverse transform and see what the two functions look like in the original space.

$$g_{4x} := A \cdot \exp(b \cdot x)$$



In the figure above, $f(x)$ is the original function, $g(x)$ are the original data with Gaussian noise, $g_4(x)$ is the least squares, best-fit line calculated by transforming from linear to natural log space. Compare this with the figures in natural log - linear space and you can see the effect of transforming the errors. If you are going to be rigorous, and demand Gaussian errors for the least squares fit, then the original data cannot have Gaussian errors; that's generally unlikely in geologic experiments.

A different technique is to do a non-linear least squares fit without transforming the axis and data. In MathCad, one sets this sort of problem up in a solve block with an assumed form of the function you want to use to fit the data. Then you provide an initial guess. MathCad uses Marquardt-Levenberg optimization to minimize the errors.

$$F(x, A, b) := A \cdot \exp(b \cdot x)$$

A sum of squares error function for this is:
$$\text{SSE}(A, b) := \sum_x (g_x - F(x, A, b))^2$$

Next, provide an initial guess as to the values of A and b ; usually these need to be tinkered with, the closer the better. The estimates from transforming the coordinates will probably make a good first guess.

$$A := 29 \quad b := -.04 \quad (\text{I got these from the earlier guess of } 20, -.04)$$

Another thing to tinker with is the tolerance. Often it is fastest to start with low tolerance, get close, then increase the tolerance and use the initial result as a first guess for the second round of iterations; the default is $TOL = .001$.

$TOL := .01$

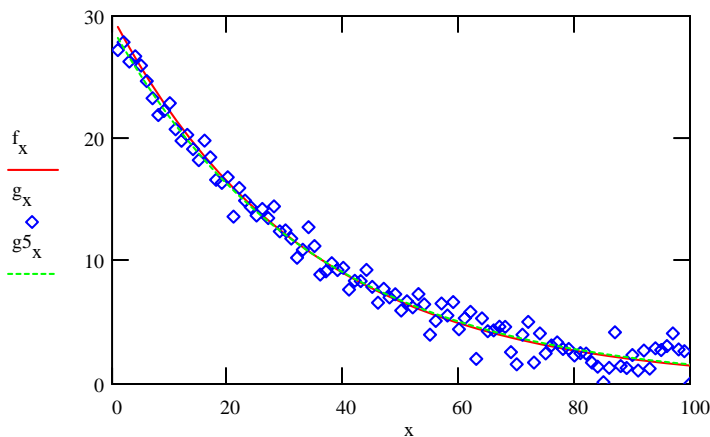
Now provide two givens for two unknowns (A,b). Obviously, $1 = 1$ is redundant but MathCad requires two conditions for two unknowns. The other condition comes right out of calculus, we want to minimize the errors:

given $SSE(A,b)=0$ $l=1$

$\begin{bmatrix} A \\ b \end{bmatrix} := \text{minerr}(A, b)$ Here is the result: $A = 29.04$ $b = -0.029$

Now compare the result to the original noisy data:

$g5_x := A \cdot \exp(b \cdot x)$



$$\text{errf3} := \left[\sum_x (g5_x - f_x)^2 \right]^{\frac{1}{2}}$$

$\text{errf3} = 2.664$

Even with low tolerance, the non-linear least squares function ($g5(x)$) is much closer to the original $f(x)$ than was the transformed fit; the error term is an order of magnitude smaller.