NASA/TP-2019- 220172

John F. Kennedy Space Center
Kennedy Space Center, Florida 32899

Technical Support Package

Real-Time Exponential Curve Fits Using Discrete Calculus

NASA Tech Briefs
KSC-13153

National Aeronautics and
Space Administration

Technical Support Package

for

REAL-TIME EXPONENTIAL CURVE FITS USING DISCRETE CALCULUS

KSC-13153

NASA Tech Briefs

The information in this Technical Support Package comprises the documentation referenced in **KSC-13153** of *NASA Tech Briefs*. It is provided under the Commercial Technology Program of the National Aeronautics and Space Administration to make available the results of aerospace-related developments considered having wider technological, scientific, or commercial applications. Further assistance is available from sources listed in *NASA Tech Briefs* on the page entitled "NASA Innovative Partnerships Program."

For additional information regarding research and technology in this general area, contact:

Kennedy Space Center
Innovative Partnerships Office
Mail Stop: KT-A2
John F. Kennedy Space Center, FL 32899

Telephone: (321) 861-7158

National Aeronautics and Space Administration Inventions and Space Act Awards

Title:	"Discrete Calculus Solution Removes Offset & Negative Limits To Fast Exponential Curve Fits: *50x faster than Non-Linear Methods.*"
Innovator(s):	Geoffrey K. Rowe, Arctic Slope Regional Corp.
Place of Performance:	Data Acquisition Lab, LETF (Launch Equipment Test Facility)
NASA Branch:	Kennedy Space Center
Concept First Proven:	April, 2004
Submitted:	July 2007
Last Update:	May 2008
Stage of Development:	Prototype, Proof of Concept Finished, and used on NASA test data (BSM Aft Skirt Debris).
First Disclosure to Others:	July 2007

Table of Contents

Table of Figures

Discrete Calculus Solution Removes Limits On Exponential Curve Fits

Developed at
Kennedy Space Center
Launch Equipment Test Facility
Geoffrey K. Rowe, ASRC, USTDC Contract

Abstract. This paper presents an improved solution for curve fitting data to an exponential equation ($Y = Ae^{Bt} + C$). This improvement is in four areas — speed, stability, determinant processing time, and the removal of limits. The solution presented in this paper avoids iterative techniques[5] and their stability errors by using three mathematical ideas — discrete calculus, a special relationship (between exponential curves and the Mean Value Theorem for Derivatives), and a simple linear curve fit algorithm. This method can also be applied to fitting data to the general power law equation $Y = Ax^B + C$ and the general geometric growth equation $Y = Ak^{Bt} + C$. The advantages are as follows:

- Speed: Iterative (non-linear) methods are 50 to 100 times slower. Previously, only iterative methods could be used when C was not zero, or when all the samples were not zero or greater.
- Stability: No bad guesses. There is no chance of making a bad first guess as sometimes happens in iterative (non-linear) techniques. Sometimes the iterative techniques "blow up" when they start with a bad guess.
- Real-Time requires determinism: Being faster would allow this method to be used in real-time applications where non-linear methods take too much processing time. But, most real-time applications require determinism (a consistent processing time from curve fit to curve fit, or at least a known maximum processing time). Iterative techniques vary greatly in the processing time they consume. This new method takes the same amount of processing time (for the same number of data points), no matter what the error distribution is. Iterative techniques sometimes only take 50 times longer, and sometimes more than a 100 times longer, depending on how the data varies and the initial guesses.
- Y can cross zero: Even if C = 0, the non-iterative methods require all sample points to be zero or greater (or transformed to greater than zero). This method does not have any such limitation.

$$Y = Ae^{Bt} + C \tag{1}$$

Problem Defined: Many tests in the real world have measurements that change exponentially *(equation 1)* where 'C' does not equal zero. Hot liquids will cool down following an exponential curve *(Newton's Cooling Law)*. Opening a valve may change the downstream pressure and temperature exponentially. Charging up a capacitor often varies exponentially. Consumer price index with inflation can follow an exponential curve. C_{14} buildup in the atmosphere should increase exponentially. In some PID control systems, a change in the set-

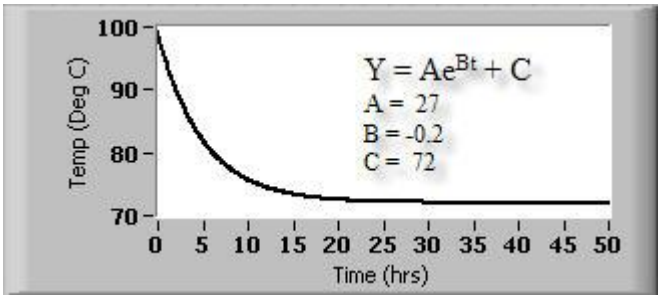

Figure 1: Cooling Exponentially To Non-Zero

point leads to exponential changes in the process variable. But all the curve fitting algorithms that I have found[5] must have C = 0 in equation 1, and the data must be all positive, or all negative. *(The nonlinear iterative techniques, that require 50 to 500 times more processing resources, can get around these problems.).*

Settled value can be predicted. Sometimes we want to know what value the data would have settled to, if we had not made any more changes to our system. Many exponential processes *(equation 1)* have 'B' < 0, which means the process will settle to some value 'C' if we wait long enough. *(As time, 't', becomes large, 'Bt' becomes a large negative number, causing Ae^{Bt} to approach zero, which leaves Y = C.)* But when we are not able to wait until the process settles, we may still want to know what value (C) the measurement would have reached, if we had waited. Many real world processes do not have C = 0. This paper attempts to address this curve fitting problem when C is not zero. This paper also addresses the fact that existing algorithms take much more processing time and can unexpectedly give huge errors that would cause control systems to become unstable. Since so many of our real world processes change exponentially, this solution would have wide applications in medicine, control systems, ice age climate change models, chemical plants, economics, politics, and various areas of science. This advantage will become clearer as we study typical statistical methods used to fit experimental data.

Typical Curve Fit Uses Logarithms. Statistical curve fits often try to minimize the sum of the square of the error (SSE). The error for each data point is computed by subtracting the expected value from the actual value. Let us review the process for fitting an exponential curve to a set of data by linearizing the relationship between a measured value and the expected value (exponential). Let us start with the simpler solution when C = 0.

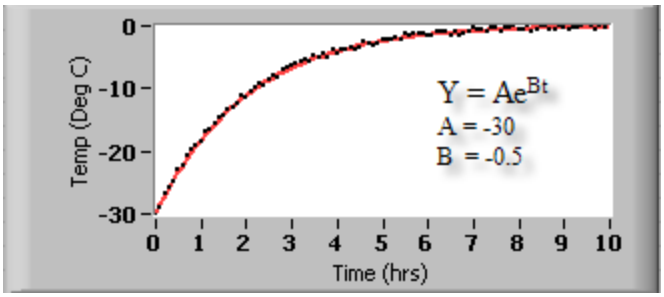

Figure 2: Typical Curve Fit, Decay To Zero

$$Y(t) = Ae^{Bt} \tag{2}$$

This says that the measured Y(t) value is expected to be equal to Ae^{Bt} for each value of time 't'. When we take the natural logarithm (*ln*) of both sides we get

$$ln(Y) = ln(A) + Bt \tag{3}$$

This provides us with a linear relationship between the *ln* of the measured value 'Y' and a combination of a constant [*ln*(A)] and a function of time (Bt). Now we can minimize the sum of the square of the errors using linear statistical methods. *(Simple **linear** curve fitting, deals with functions that are **linear in their parameters**, even though they may be **nonlinear in their variables.**[6])*

Since we are using real world data, we do not have an infinite number of Y-axis values between any two X-axis values. *(In calculus, you have a function that has an infinite number of Y-axis values between any two X-axis points, because the function is assumed to be "continuous" between any two points. We are using examples where the X-axis is time, but this method is not limited to the X-axis having to be time.)* Since we have discrete values for Y and 't', we can add the subscript '$_i$' denoting each sample with which we are working. Therefore equations 2 and 3 would be

$$Y(t_i) = Ae^{Bt_i} \tag{4}$$

$$ln(Y_i) = ln(A) + Bt_i \tag{5}$$

Using equation 5, the error (E_i) for each sample would be: E_i = measured – expected

$$E_i = ln(Y_i) - [ln(A) + Bt_i] = ln(Y_i) - ln(A) - Bt_i \tag{6}$$

Now that the model has been linearized, we can add up the errors for all the samples and try to derive values for A and B that will minimize the errors. In order to derive the best curve fit, we have to make our estimator for each of the errors positive before we add them. Otherwise, large negative errors and large positive errors will cancel each other. The sum of the errors would be zero, but the individual errors could still be very large. Figure 3 shows an example where the errors were large, yet they summed to zero. Notice the large positive errors for the first part (time < 0.2), then large negative errors (0.2 <

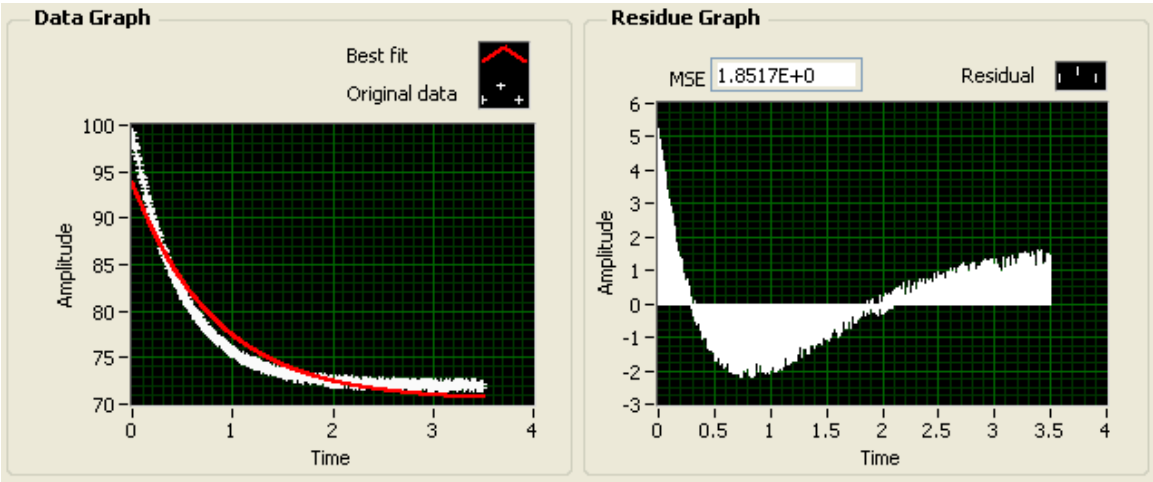

Figure 3: Large Error, Even Though Sum Of Errors Is Zero

time < 2), and finally large positive errors (time > 2). Again, the sum of these errors is zero. But the average (mean) of the sum of the square of the errors is large (MSE on the graph means Mean Square of the Error).

<u>Requires making all the errors positive.</u> We could take the absolute value of each error, but then we could not apply calculus *(calculus is used as one of the steps in statistical curve fitting)* to solve for A and B because the absolute value makes the equation for the error estimator discontinuous.[1] Therefore mathematicians usually square all the errors

and then add. This introduces a bias toward data points that are farthest from the *true* curve.[1] But this bias is not the only bias.

Statistical bias due to taking logarithms. When we took the natural logarithm of our samples, we introduced another bias toward samples with smaller values. See table 1 where each sample has an error of 0.5. Then notice the biased error that results. If we have a data acquisition system that is measuring 5000 psi with only 0.5 psi of error, look at what happens when the pressure falls to 5 psi. The 0.5 psi error has been biased such that the final error is 953 times greater at 5 psi compared to 5000 psi.

Real Error	e^{bt}	e^{bt} + Error	$ln(e^{bt})$	Y = ln (e^{bt}+Error)	Biased error Y - $ln(e^{bt})$
0.500	**5000.0**	**5000.5000**	**8.5172**	**8.5173**	**-0.000100**
0.500	4000.0	4000.5000	8.2940	8.2942	-0.000125
0.500	1000.0	1000.5000	6.9078	6.9083	-0.000500
0.500	500.0	500.5000	6.2146	6.2156	-0.001000
0.500	100.0	100.5000	4.6052	4.6102	-0.004988
0.500	50.0	50.5000	3.9120	3.9220	-0.009950
0.500	12.0	12.5000	2.4849	2.5257	-0.040822
0.500	**5.0**	**5.5000**	**1.6094**	**1.7047**	**-0.095310**
0.500	3.0	3.5000	1.0986	1.2528	-0.154151
0.500	2.0	2.5000	0.6931	0.9163	-0.223144
0.500	1.1	1.6000	0.0953	0.4700	-0.374693
0.500	0.70	1.2000	-0.3567	0.1823	-0.538997
0.500	0.20	0.7000	-1.6094	-0.3567	-1.252763
0.500	0.02	0.5200	-3.9120	-0.6539	-3.258097

Table 1: Logarithm Causes Error To Be Skewed (Biased)

This also violates the assumption of a ***constant standard deviation for all samples.*** And in addition to all this, we violate the assumption that the errors have a Gaussian distribution *('bell curve')*. Taking the logarithm has made the error distribution move farther away from a Gaussian distribution[3].

Also, standard curve fitting algorithms assume that all the variation in the measured value is due to errors in Y and not due to errors in our measurement of time ('t'), or any X-axis variable. In other words, they assume all the error is on the Y axis and we know the X-axis value perfectly. Also, the errors that are being minimized are vertical errors, not perpendicular errors *(perpendicular to the curve fit)*. This is shown by figure 4 and explained by Weisstein, Eric W. *"Least Squares Fitting."* From *MathWorld*--A Wolfram Web Resource.[1]

> In practice, the *vertical* offsets from a line (polynomial, surface, hyperplane, etc.) are almost always minimized instead of the perpendicular offsets. This provides a fitting function for the independent variable X that estimates y for a given x (most often what an experimenter wants), allows uncertainties of the data points along the X- and Y-axes to be incorporated simply, and also provides a much simpler analytic form for the fitting parameters than would be obtained using a fit based on perpendicular offsets. In addition, the fitting technique can be easily generalized from a best-fit *line* to a best-fit *polynomial* when sums of vertical distances are used. In any case, for a reasonable number of noisy data points, the difference between vertical and perpendicular fits is quite small.

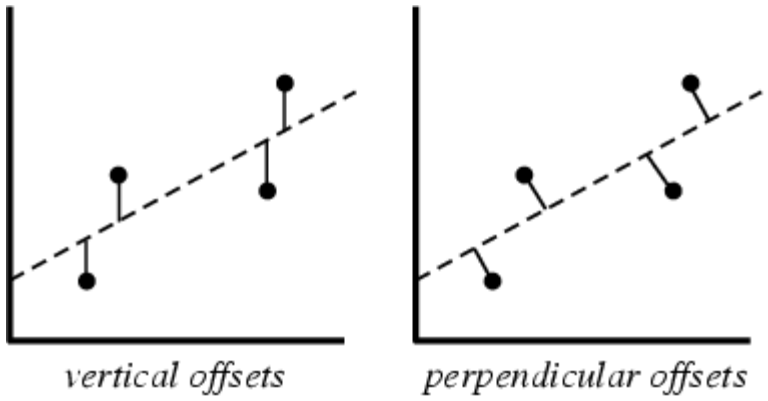

Figure 4: Vertical Versus Perpendicular Offsets

Requires making all samples positive. Another Problem with the standard exponential curve fit is that sample values cannot be less than zero. For $ln(Y_i) = ln(A) + Bt_i$ we cannot have any sample values (Y_is) that are less than zero, because you cannot take the logarithm of a negative value. *(If **all** Y_i's are less than zero, you can multiply each Y_i by negative one and then swap the sign on the derived value for A.)* Yet, when our real world data approaches zero, the noise in our measurement allows some samples to be less than zero. If we throw out those samples less than zero, we bias the curve fit upward *(and this shifts the value for 'C', which is the value we need to predict where the system will eventually settle).*

Also the standard curve fitting algorithms for a simple linear curve fit have a bias due to squaring the error term.[1] Remember, we must make all our error estimators positive so we can minimize the sum of these positive errors. The typical way, is to square the error. But, squaring the error can cause the final curve fit to be biased toward the sample values farthest from the real curve fit.

Bias in Sum of the Square of the Errors Method (SSE). Suppose you run an experiment where you measure the change in temperature after you mix two chemicals together. You record the temperature every hour for ten hours. Then you repeat your experiment five more times. You now have six temperatures for EACH hour, from zero to ten hours. Therefore, for each hour you have six data points. Each data point is *"pulling"* on the final curve fit. Each recorded temperature *(out of the six)* wants to minimize the error *(for its time period)* by *pulling* the curve fit line toward its recorded temperature. Let us look at the six recorded temperatures when time equaled to three hours.

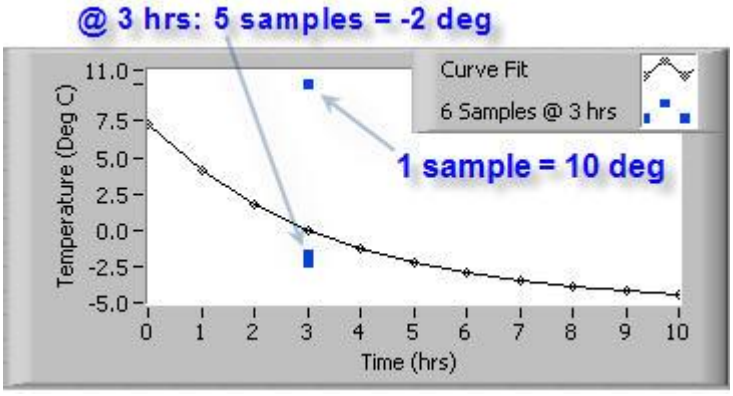

Figure 5: Data *"Pulls"* The Curve Fit Line

Imagine five of these six points are two degrees C below zero, and there is one recorded temperature ten degrees above zero. If you minimized the error by using the sum of the absolute value of the errors, the five temperatures below the curve fit are pulling downward (-) with ten *units of force* (5*[-2] = -10). The one recorded temperature ten degrees above the fit line is pulling upward with 10 *units of force* (1*10 = 10). You can visualize how this does a good job of minimizing the error (-10 + 10 = 0). *(Please ignore the obvious: the +10 deg C value was probably bad since the -2 deg C values repeated five times. Assume the 10 deg value was valid.)*

But, what if you use the standard linear statistical method of minimizing the square of the errors *(so you can use calculus to solve for the answer)*? How much "*force" is pulling* upward, versus "*force" pulling* downward? Now you have $5 * (-2)^2 = 20$ units of *force pulling* downward, and $1 * (10)^2 = 100$ *units of force* pulling upward. Notice how squaring the error biased the final curve fit toward the temperature that was 10 degrees higher *(20 down + 100 up = 80 units of force pulling upward, compared to -10 + 10 = 0 if we used the sum of the absolute value of the errors)*. Therefore, the curve fit would be pulled upward until the sum of the square of these errors = minimum. *(Note: I do not think this example, of how much "force" each point pulls on the curve fit line,* is *completely accurate, but it gives you an idea. I have tried to explain this bias, of the sum of the square of the errors, as talked about in the statistics literature. I have run simulations between using the sum of the absolute value of the errors and using the sum of the square of the errors - as talked about here. I have not been able to confirm the bias in the sum of the square of the area as discussed above, when compared to using the sum of the absolute value of the errors.)*

Slower iterative techniques can avoid many of these biases if they try to minimize the sum of the absolute values of the errors. Therefore, such iterative techniques should give an improved curve fit over **any** linear method. But, iterative techniques require much more processing resources and the improved curve fit is not much better then the linear statistical curve fitting techniques in many applications. So let us try the linear curve fitting method for fitting data to an exponential curve.

Typical solution for the A and B coefficients in $ln(Y_i) = ln(A) + Bt_i$ *(equation 5)*. We write the equation for the sum of the square of the errors.

$$Sum(E_i)^2 = Sum[\, ln(Y_i) - ln(A) - Bt_i \,]^2 \qquad (7)$$

Let us rewrite equations 5 and 7 with the following substitutions

$$Z_i = ln(Y_i) \qquad (8)$$

$$K = ln(A) \qquad (9)$$

Let R be the sum of the square of the error. Then R will be a function of K, Z, and B.

$$R = Sum(E_i)^2 = \sum E_i^2 \qquad (10)$$

Therefore equations 5 and 7 become

$$Z_i = K + Bt_i \tag{11}$$

$$R = \sum[Z_i - K - Bt_i]^2 \tag{12}$$

We then take the partial derivative of R with respect to K. Then we set this partial derivative to zero *(remember the minimums and maximums of an equation can be found where its derivative is equal to zero, i.e., zero slope.)* We derive a second equation by taking the partial derivative of R with respect to B. We set this second partial derivative to zero. If K and B are independent from each other, then the partial derivatives may give independent equations. This gives us two independent equations with only two variables (K & B).

$$dR/_{dK} = -2*\sum[Z_i - K - Bt_i] \quad = 0 \tag{13}$$

$$dR/_{dB} = -2*\sum[t_i(Z_i - K - Bt_i)] = 0 \tag{14}$$

Solving the equation either through inverse matrix operations or through variable substitution gives

$$K = [\ \sum Z_i \sum t_i^2 - \sum t_i \sum (t_i Z_i)] \ / \ [\ n\sum t_i^2 - (\sum t_i)^2\ ] \tag{15}$$

$$B = [\ n\sum t_i\, Z_i - \sum t_i \sum Z_i\ ] \ / \ [\ n\sum t_i^2 - (\sum t_i)^2\ ] \tag{16}$$

Therefore $ln(A)$ and B are

$$ln(A) = [\ \sum ln(Y_i) \sum t_i^2 - \sum t_i \sum (t_i\, ln(Y_i))\ ] \ / \ [\ n\sum t_i^2 - (\sum t_i)^2\ ] \tag{17}$$

$$B = [\ n\sum (t_i\, ln(Y_i)) - \sum t_i \sum ln(Y_i)\ ] \ / \ [\ n\sum t_i^2 - (\sum t_i)^2\ ] \tag{18}$$

Remember, A and all Y_is must be greater than zero because we cannot take the log of a negative number *(again, if **all** Y_i's are less than zero, then we can get around this problem by multiplying all Y_i's by -1, and changing the sign of A, that we derive).*

How to use slow iterative methods when C is not zero. Can we use SSE method to solve for A, B, and C when C does not equal zero? Remember, our equation is $Y = Ae^{Bt} + C$ *(eq. 1).* If we take the *ln* of both sides we get

$$ln(Y_i) = ln(Ae^{Bt} + C) \tag{19}$$

This equation cannot be manipulated to give us a linear relationship[5,10]. Therefore we cannot use the standard linear statistical methods. Mathematicians have resorted to

nonlinear techniques where numerical methods are used[4]. The specific nonlinear method typically used *(and claimed to be the most accurate)* is called the Levenberg-Marquardt method. H.J. Motulsky and A Christopoulos say[2]

Every nonlinear regression method follows these steps:

1. **Start with an initial estimated value for each variable** in the equation. The program can't determine the best-fit values unless you first give it some estimates. In most cases, these estimates don't have to be very accurate. 2. Generate the curve defined by the initial values. Calculate the sum-of-squares (the sum of the squares of the vertical distances of the points from the curve). Why is goodness of fit defined as the sum of squares? Adjust the variables to make the curve come closer to the data points. There are several algorithms for adjusting the variables, as explained below. 4. Adjust the variables again so that the curve comes even closer to the points. Repeat many times. 5. Stop the calculations when the adjustments make virtually no difference in the sum-of-squares. 6. Report the best-fit results. The precise values you obtain will depend in part on the initial values chosen in step 1 and the stopping criteria of step 5. This means that repeat analyses of the same data will not always give exactly the same results.

Step 3 is the only difficult one (for the computer). Prism (and most other nonlinear regression programs) uses the method of Marquardt and Levenberg, which blends two other methods, the method of linear descent and the method of Gauss-Newton.

Motulsky and Christopoulos show a 3D graph (figure 6) that illustrates this nonlinear method for two unknowns. *(They use the sum of the square of the errors and not the sum of the absolute value of the errors.)* Also see the top of figure 6 for the equation being minimized. Notice that the best fit is where the sum of the square of the errors is minimum. While they were not using an exponential curve fit *(and they only had two variables - B & K - instead of our three variables - A, B, & C)*, their example does show how the initial guess could start somewhere other than the bottom, and work its way down. They were trying to fit the following equation to their data.[2]

$$Y = \frac{B_{max} \cdot X}{K_d + X}$$

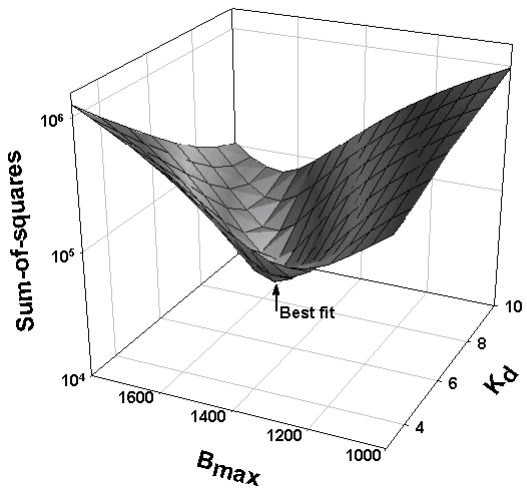

Figure 6: Non-Linear Fits Keep Guessing To Find Bottom

Solution(s): So how are we going to avoid this processor intensive nonlinear method? And can we also avoid the limitation that A and all Y_is have to be greater than zero? What we need is a relationship *(equation)* that does not include C *(or maybe not even A)*. If we could first solve for B alone, then we could easily solve for A and C using linear techniques later. If we use regular or discrete calculus, we can take the derivative of equation 1, which gets rid of C.

$$dY/_{dt} = ABe^{Bt} \tag{20}$$

This says the slope of our curve fit is not related to C. We could then take the *ln* of both sides and proceed like we did before *(like equations 5-12).*

$$ln(dY/_{dt}) = ln(AB) + Bt \tag{21}$$

$$Sum(E_i)^2 = Sum[\ ln(dY/_{dt}) - ln(AB) - Bt\ ]^2 \tag{22}$$

Will Partial Derivatives Work? But, when we try to take the partial derivative with respect to 'AB' and another partial derivative with respect to B *(of the sum of the square of the errors method)*, we cannot get two independent equations. This is because 'AB' is not independent of B. Partial derivatives require that the independent variables be independent from each other.

So how do we get around this problem? Let us add in the discrete mathematics notation ('$_i$'). Equation 20 becomes

$$dY(t_i)/_{dt} = ABe^{Bt_i} \tag{23}$$

So use discrete calculus to isolate B. We notice that the **change in the slope** (from t_1 to t_2) is only related to B and 't'. This can be seen by dividing the slope at one point by the slope at the next point. A will drop out.

$$\frac{dY(t_1)/_{dt}}{dY(t_2)/_{dt}} = \frac{ABe^{Bt_1}}{ABe^{Bt_2}} \tag{24}$$

Let

$$dY(t_i)/_{dt} = Y'(t_i) = Y_i' \tag{25}$$

Then equation 24 can be rewritten as

$$\frac{Y'_1}{Y'_2} = \frac{e^{Bt_1}}{e^{Bt_2}} = e^{B(t_1 - t_2)} \tag{26}$$

Now we can take the *ln* of both sides and get

$$ln(Y'_1/Y'_2) = B(t_1 - t_2) \tag{27}$$

Therefore B is a function of the change in the ratio of two slopes and the time between these slopes.

$$B = \ln(Y'_1/Y'_2) / (t_1 - t_2) \qquad (28)$$

Notice that this means you **only need three data points** *(with $\Delta t_1 = \Delta t_2$)* to define an exponential curve fit. Any three points *(equally spaced)* will work. Notice that this seems intuitively reasonable: three unknowns, three points, three equations. Intuitively we may understand how the first point, when t=0, would give you A+C. Another point when 'Bt' is a large negative number should be very close to 'C'. And another point in-between should somehow be related to A & B. It is this *"somehow"* relationship that we will derive.

Another advantage: Y_is can go from positive values to negative values. *(Remember even if C = 0, the normal methods leaves you with $\ln(Y_i)$ so none of your data can go from positive to negative values. You cannot take the logarithm of a negative value.)* It will be shown that you can use a simple linear (1st order) curve fit that is computationally light.

But, this slope estimator is wrong. This estimator for B should not be a "good" estimator because the real derivative at any point is the tangent line at that point. Yet the discrete derivative[10] we took is the slope of the secant line between two points. Also, this discrete derivative is highly affected by even the smallest amounts of noise.

Side Note: Over two years later, I found a private web site that uses the ratio method (Y'_1/Y'_2), only as a first guess for B for his own nonlinear iterative technique[8]. His point by point slope ratios will not give reliable estimates for B unless the noise is very very low. Perhaps this is why he only uses this ratio method as a first guess to his non-linear technique. (*Also see "*Middle terms are thrown out*", page 17, for a discussion on why averaging the ln of ratios does not work.)* No company that sells statistics software, seems to use this method *(probably because of its instability on real world data)*. Real world data usually has too much noise which greatly affects these slopes calculated **between adjacent** data points. This noise can be averaged out, but only after you throw away all slope ratios that are negative. In other words, sometimes this noise will cause the slope at one point to be positive *(or negative)* and the slope at the next point to be the negative *(or positive)*. This will cause the estimator to fail *(remember equation 28 shows that Y'_1/Y'_2 must be > 0)*. If you throw away these negative slope ratios you are left with an even worse biased estimator for B. For more information about this ratio method on real world data, see figure 10 and then the "Noise" discussion in part 3 on page 17.

I have simulated this B estimator *(with different levels of noise)* and many other B estimators using LabVIEW. We must get away from these noise problems for real world applications. If the real world data had zero noise, then the estimator for B would be exact, and no iterations would be needed. We would know A, B, and C perfectly *(if there was no noise)* from any three equally spaced data points *(this will be proven shortly)*.

Four Problems. What method do we use on real world data? First, let us **define the problems** we are left with.

1. The discrete derivative has an error for exponential curves. This is because we used the secant line's slope as an estimator for the real slope at a particular point in time. But, the secant line's slope is not the real slope at the place where we took the derivative. Yet, the secant line's slope is the real slope somewhere between the two points we used for the secant line *(remember the "Mean Value Theorem for Derivatives")*. But it is not the halfway point between the points used for the secant line. The fraction (F) of the distance between the points, where the secant line's slope is the real slope, is a function of the exponential coefficients that we are trying to derive. See figures 7 and 8.

2. A/D converters *(analog to digital converters)* are not perfect. Therefore they inject noise into our sampled system.

3. Real world processes have noise usually much greater than the A/D noise.

4. The change between two adjacent data points is usually very small. We usually sample data much faster than the Nyquist criteria. The data must change from sample to sample more than the noise at any one sample. Otherwise we can end up dividing by zero or end up with a negative slope ratio *(and again, we cannot compute the ln of a negative slope ratio).*

Problem #1: The Discrete Derivative. Where is the real slope equal to the secant line?

When we use the secant line as an estimator for the derivative, why do we have an error? Does this error vary depending on the parameters of the exponential equation we are trying to derive?

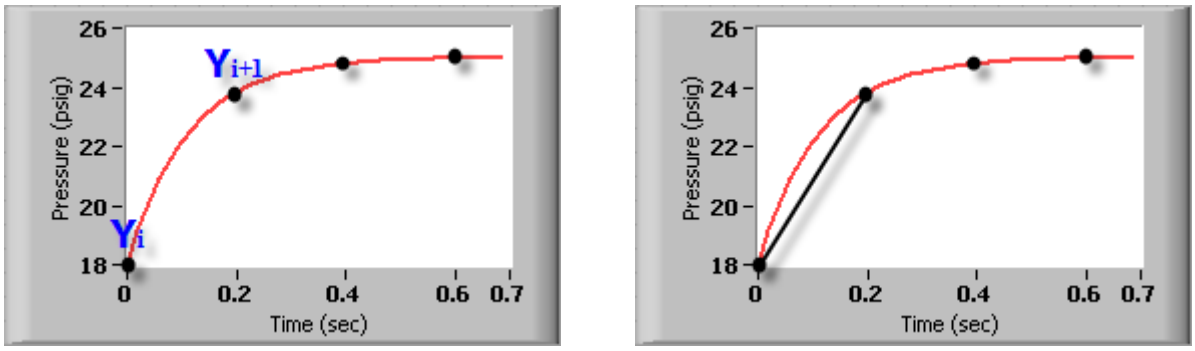

Figure 7: Adjacent Data Points As An Estimator For The Slope.

Notice that the **time** (Fig 8, left graph, blue dot) when the secant line's slope is the real slope is **not halfway between** the data points Y_i and Y_{i+1}. (Fig 8, right graph). **This fractional distance** between the sampled data points is **not the same for all exponential curves.** It is a function of the parameters we are trying to derive.

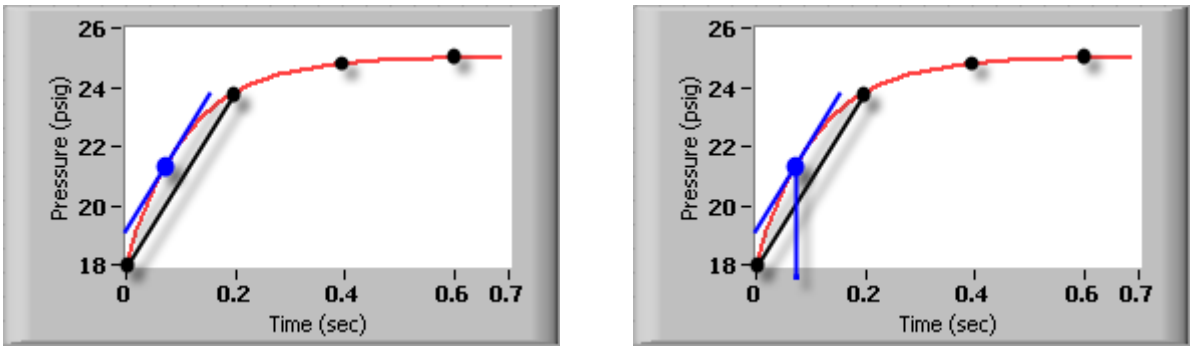

Figure 8: Secant Line Slope Is Somewhere Between.

The good news is that this fractional distance (F) is the exact **SAME fraction for the same exponential curve.** In other words, the secant line's slope is the real slope at the same fractional distance between any two points on the same exponential curve. Or another way to say this is, for any given exponential curve, 'F' is a constant.

This means that the time between data points (*Δt used for the secant lines)* is the real time between (Δt) the real tangent line points *(points where the slope of the exponential curve is precisely equal to the estimators for the slopes – that we derived by using the secant line slopes)*. This is true even if we do not know the exact point where the secant line's slope is the real slope of the curve we are trying to derive. (See figure 9) The time between points is 0.2 seconds, and the time between tangent line slopes is also exactly 0.2 seconds. This will become very important later as we get to the different derivations for our B estimator. Remember, once we derive a good estimator for B, finding A and C becomes easy because we can rely on standard linear curve fitting techniques.

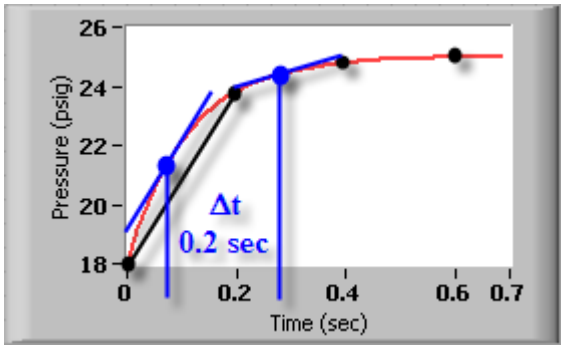

Figure 9: Tangent Point Δt = Sample Δt

Problems 2, 3, & 4: Small changes become a large problem. A/D Limits, Noise, and Small changes between samples:

If we have far more than three data points, it might be reasonable to assume that a good estimator for B would be the average of individual estimators for B *(from equation 28).*

$$B = \sum ln(Y'_i/Y'_{i+1}) \,/\, \mathbf{n}(t_i - t_{i+1}) \qquad (29)$$

But, this has many problems when we use real world data *(see figure 10).*

1. *Negative Slope Ratios.* Noise often causes some of the individual slope ratios to be negative. Then we have to throw out these negative ratios because you cannot take the logarithm of a negative number. All slopes must be either less than zero or all greater than zero. Noise often causes this to be violated.

2. *Middle terms are thrown out.* Remember, the sum of logarithms is equal to the logarithm of the product of the terms.

$$ln(Y_1) + ln(Y_2) + ln(Y_3) = ln[\ (Y_1)\ (Y_2)\ (Y_3)\ ] \qquad (30)$$

Therefore, the sum of the logarithms of ratios is equal to the logarithm of the product of the ratios

$$\sum ln(Y'_i/Y'_{i+1}) \quad = \quad ln[(Y'_i/Y'_{i+1})\ (Y'_{i+1}/Y'_{i+2})\ldots\ (Y'_{n-1}/Y'_n)] \qquad (31)$$

which reduces to

$$\sum ln(Y'_i/Y'_{i+1}) \ = \ ln(Y'_i/Y'_n) \qquad (32)$$

but this shows that if we estimate B this way, we ignore all estimated slope ratios between the first and last slopes [*with the time between them* $n(t_i - t_{i+1}) = t_i - t_n$]. This makes the B estimator totally dependant on only the first and last computed slopes. If there was no noise in your data, this would work. But remember, without noise any three evenly spaced points would work.

3. *Noise greater than the real change.* When B < 0, and |Bt| is large (when t >> 0), the data is almost exactly equal to C. The data is not changing very much at this point in time. But, the noise in the data is usually still the same, and this noise will cause the secant line slope to be very close to zero *(and sometimes less than zero)*. When you divide by almost zero (Y'_{i+1}), the B estimator blows up. When the secant line's slope is close to zero, it gives too much weight for these sample points.

Now let us use an example to understand the problems that noise can cause. Suppose we are measuring pressure downstream of some valve we just opened *(zero to 50 psig; therefore 1% of full scale would be 0.5 psig).* We are measuring the pressure 10 times per second. Sometimes the pressure we are measuring increases less than one percent of full scale *(0.5 psig)* **between two adjacent data points** *(fairly typical).* Then suppose our noise *(physical, electrical … etc.)* is sometimes as great as one-half of one percent of full scale (0.25 psig). Now when we calculate the slope changes between adjacent data points *(blue dots on the right side of Figure 10)*, we can end up dividing by zero. In other words, before we add in the noise, with $\Delta t_1 = \Delta t_2$,

Y_i = 22.0 psig
Y_{i+1} = 22.0 + 0.6 change = 22.6 psig
Y_{i+2} = 22.6 + 0.4 change = 23.0 psig
Noise = 0.25 psig

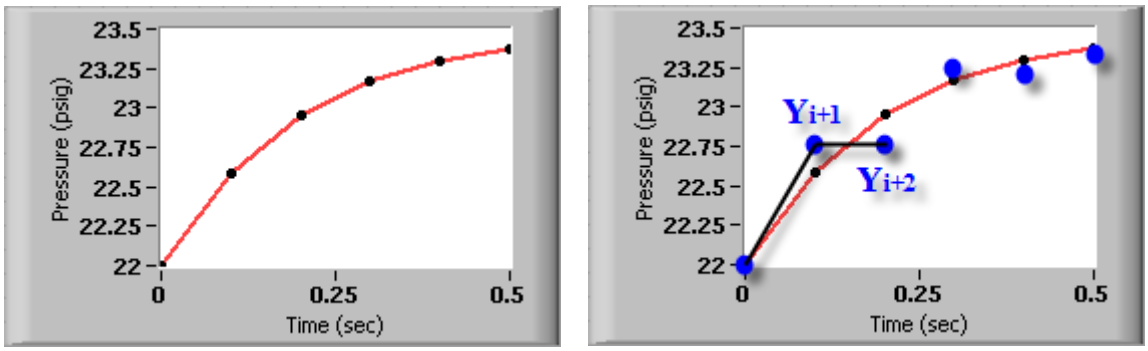

Figure 10: How Noise Affects Our Slopes.
If the slope is zero, this method fails.
The black dots are the sample points if there was no noise.
The blue dots are the actual samples points recorded – including noise.

Then our real world measured values (they include noise) for Y_i , Y_{i+1} , and Y_{i+2} could be,

Y_i = 22.0 psig
Y_{i+1} = 22.0 + 0.6 change + 0.15 noise = 22.75 psig
Y_{i+2} = 22.6 + 0.4 change - 0.25 noise = 22.75 psig

When we compute the slope ratios, this leads to dividing by zero

$Y'_1 = [Y_i - Y_{i+1} + \text{noise}] / \Delta t_1 = [22.0 - 22.75] / \Delta t_1 = -0.75 / \Delta t_1$
$Y'_2 = [Y_{i+1} - Y_{i+2} + \text{noise}] / \Delta t_2 = [22.75 - 22.75] / \Delta t_1 = (0.0) / \Delta t_2 =$ **0.0**

Therefore,

$Y'_1 / Y'_2 = (-0.75) / (0.0) =$ undefined, or sometimes called infinity.

Also, if the second measurement (Y_{i+1}) has a slightly more positive noise of 0.2 psig, and the second slope estimator has the previous example's -0.25 psig noise, then

Y_i = 22.0 psig
Y_{i+1} = 22.0 + 0.6 change + 0.2 noise = 22.8 psig
Y_{i+2} = 22.6 + 0.4 change - 0.25 noise = 22.75 psig.

This leads to a negative ratio between two adjacent slopes.

$Y'_1 = [Y_i - Y_{i+1} + \text{noise}] / \Delta t_1 = [22.0 - 22.8] / \Delta t_1 = -0.8 / \Delta t_1$
$Y'_2 = [Y_{i+1} - Y_{i+2} + \text{noise}] / \Delta t_2 = [22.8 - 22.75] / \Delta t_1 = 0.05 / \Delta t_2 = 0.05$

Now we end up with a slope ratio that is negative,

$Y'_1 / Y'_2 = (-0.8) / (0.05) = -16.0$.

But if the slope ratio is negative, we cannot compute the $\mathcal{h}$ of this ratio.

In real world applications, this is often worse. If we sample too quickly, we do not have large enough changes between sample points to overcome this noise problem. If we sample slowly, we eventually have the same problem. When the measured value gets close to its final value *(for typical systems B<0),* it is not changing enough between samples to overcome this noise problem. *(Even if the noise is less than one hundredth of one percent of full scale)*

This noise has always been a problem with discrete calculus. Perhaps that is why PhDs in mathematics are still working on the best estimators for finite derivatives[10]. They have derived Taylor series expansions[7] for correcting the derivative estimators at a point, but the noise in real world data will still interfere with deriving these slope ratios. See the paper titled "Derivative Approximations on Time Scales."[10]

Now, let us revisit the problem of using the secant line as an estimator for the real slope. We will find the curious solution talked about on page 15 (*"Problem #1"*) very useful. The mean value theorem of derivatives (for continuous functions) says the secant line's slope between any two points (Y_i and Y_{i+1}) is equal to the tangent line at some location between Y_i and Y_{i+1} (see Figures 7-9). Therefore, when we take the discrete derivative at some point (Y_i), the actual slope we calculate is the secant line's slope. But this secant line's slope is not the real slope at the point where we took the derivative. Yet, this secant line slope is the real slope at some point between Y_i and Y_{i+1}. One might think the tangent point would be half way between Y_i and Y_{i+1}. Then we could average the forward and backward derivatives *(as typically used in discrete calculus)* to estimate the real slope at any point on the exponential curve. But, the real slope is not at the halfway point.

Remember, it was noticed that for the **same exponential function**, that this tangent line matched the real slope at the **same fractional distance (F) between** the two data points used for the secant line *('F' is a constant for a given exponential curve)*. Therefore the real Δt_t, between two tangent lines *(derived from the slope of the secant lines)*, is equal to the Δt between the data points used to derive the secant line slopes. If we combine this with the following derivation, that the change in slope ratios is only related to B and Δt (Equation 28), then we can say that it does not matter that the secant line is a poor estimator for the slope at a given point. The change in slope ratios is only dependant on B and the time between points. **The true slope will change the same fractional amount *(our ratios)* for a fixed Δt, no matter where on this curve you start.**

Therefore, we can use equation 28, or use more than three points as shown next. In order to take advantage of all our good data points, we will have to do some kind of averaging. But should we use averages in evenly spaced regions? Before we optimize for noise, let us derive a simple linear technique.

Overlapping Linear Fit Method.

Secant Line, Linear Fit, and the Mean Value Theorem. A linear fit of an exponential curve is like the secant line we talked about earlier. We know, from the mean value theorem of derivatives (for continuous functions), that any linear curve fit of an exponential curve will give us a slope that is equal to the real slope of the exponential

curve at some point along the real curve. This is true because we can apply the mean value theorem of derivatives for any continuous function. Since all exponential curves would have only one point where the slope of this linear fit would be the real slope of the curve, then we can use this to estimate B. We will take the linear curve fit of two different regions of the data. This works because:

1. The change in slope is only dependant on B and the change in time between the slopes. In other words, for a given exponential curve, the change in slope/Δt is constant. (see equation 28)

2. B is dependent on the change in slope and the change in time. Our measured data gives us the change in slopes, but we also need to know the change in time. But we know that the measured time between the secant lines is also the correct time between where the secant lines are the real slope.

Therefore, our estimator for B becomes a simple linear fit of two different regions. The two different regions can even overlap each other. The change in slope of these regions and the time between these regions give us a good estimator for B.

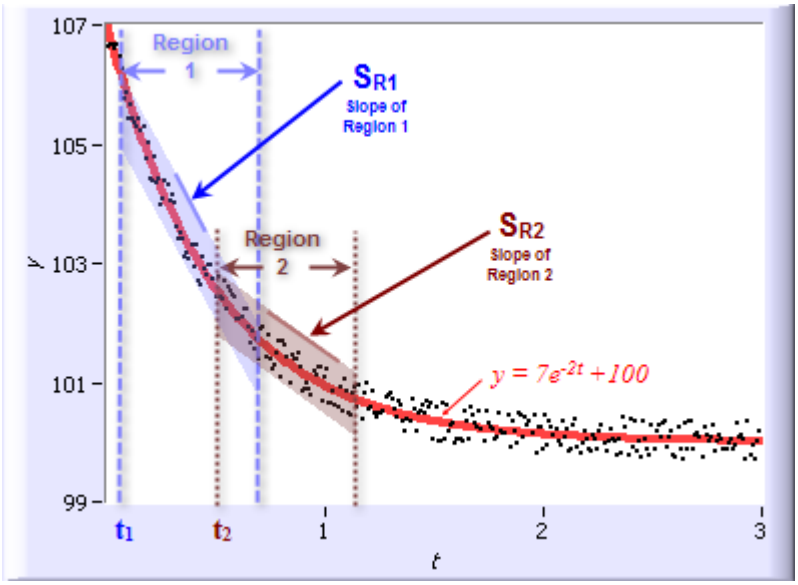

Figure 11: Overlapping Regions & Linear Fit Method

Let S_{R1} be the slope calculated for region 1. Let S_{R2} be the slope calculated for region 2. Then the change in slope can be estimated by S_{R1}/S_{R2}. The time used to cause this change in slope is equal to the time between the linear fit regions. Therefore:

$$B = ln(S_{R1}/S_{R2}) / (t_1 - t_2) \qquad (33)$$

Steps In Overlapping Linear Fit Method. Here is a more direct way to derive B that will show how we can minimize the effects of noise by using overlapping regions of data. This method relies on previously discussed estimators of slopes using exponential curves (see equations 23-28).

Let us start with any four data points on the curve (no noise involved in this four point example). We will call these four points 1, 2, 3, and 4. Notice that they are not equally spaced, but that the times between the points have a pattern where $t_2 - t_1 = t_4 - t_3$.

Therefore we can say $t_3 - t_1 = t_4 - t_2$. Overlapping regions (3-1 and 4-2) are $t_2 - t_1$ (Δt) apart. This will be useful in the following derivation. Let:

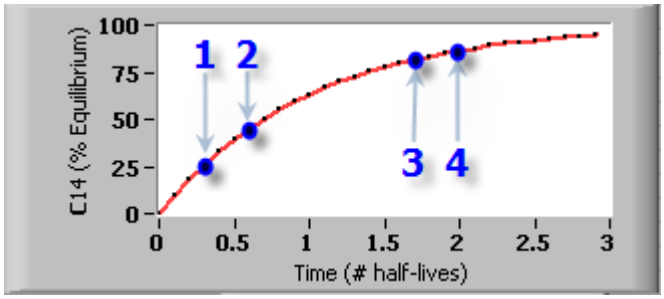

Figure 12: Overlapping – Pairs of Points From 2 Regions

$$Y(t_1) = Ae^{Bt_1} + C \tag{34}$$

$$Y(t_3) - Y(t_1) = Y_3 - Y_1 = (Ae^{Bt_3} + C) - (Ae^{Bt_1} + C) = A(e^{Bt_3} - e^{Bt_1}) \tag{35}$$

$$Y(t_4) - Y(t_2) = Y_4 - Y_2 = (Ae^{Bt_4} + C) - (Ae^{Bt_2} + C) = A(e^{Bt_4} - e^{Bt_2}) \tag{36}$$

Therefore:

$$\frac{Y_3 - Y_1}{Y_4 - Y_2} = \frac{A(e^{Bt_3} - e^{Bt_1})}{A(e^{Bt_4} - e^{Bt_2})} = \frac{e^{Bt_3}(1 - e^{B(t_1 - t_3)})}{e^{Bt_4}(1 - e^{B(t_2 - t_4)})} \tag{37}$$

If we let $t_1 - t_3 = t_2 - t_4$, then

$$\frac{Y_3 - Y_1}{Y_4 - Y_2} = \frac{e^{Bt_3}(1 - e^{B(t_1 - t_3)})}{e^{Bt_4}(1 - e^{B(t_2 - t_4)})} = \frac{e^{Bt_3}}{e^{Bt_4}} = e^{B(t_3 - t_4)} \tag{38}$$

Now, it is easy to derive B

$$ln[(Y_3 - Y_1) / (Y_4 - Y_2)] = B(t_3 - t_4) \tag{39}$$

$$B = \frac{ln[(Y_3 - Y_1) / (Y_4 - Y_2)]}{(t_3 - t_4)} \tag{40}$$

Why is this method more reliable? It minimizes the affects of noise. If there was no noise, then any of the techniques mentioned in this paper would work equally well. Without noise, we could use any three data points (equally spaced) to compute the A, B, and C coefficients perfectly. Now we will try to understand the effect of noise on our estimators.

Let us try to look at this curve fitting from a graphical point. Look at the equations we are assuming for our curve fit, and think about how this looks on a graph.

$$Y = Ae^{Bt} + C \tag{41}$$

$$Y(t_i) = Ae^{Bt_i} + C \qquad (42)$$

This technique works because we can isolate some behavior (that we can measure) that is only caused by our unknown B. We subtracted two measurements to get rid of C. In other words, this "difference" measurement is independent of C. We often then divide by (another "difference" measurement) to get A to drop out. We are then left with some function (a ratio of differences) that only depends on B. Once we solve for B, we can easily solve for A. Then we can even more easily solve for C.

Sometimes The Error In Estimating the B Coefficient Is Not Critical for C. *Often times the small error in our estimate for B has very little effect on our final error in estimating C. This is because the final error for C depends on the combined estimates for B and A. Our estimate for A uses B and the real data. This seems to cause the error in our estimate for A to sort of "make up for" some of the error in C, that is due to our error in estimating B. This is very useful, since many applications only want the final settled value C (they do not depend on the A and B coefficients).*

But, the ratio of "differences" that we use for computing "B" has problems if two things occur simultaneously. 1) Our data is noisy and 2) if one (or both) of the "difference" measurements is in the part of the curve where the noise is almost as great as this "difference" measurement (Fig 13, "2nd Diff, very little change"). (This "difference" measurement is the change in Y if there was zero noise.) This occurs because our estimate for B relies on two good measurements of change (difference measurements).

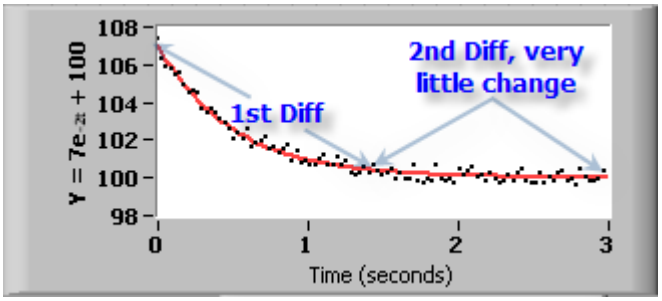

Figure 13: Small Changes In Slope Are A Problem

Example – see Fig 13: Suppose the noise is greater than the smaller of the two difference measurements (i.e., where the slope is getting closer to zero). Then our estimate for the real difference is greatly affected by the noise. If noise causes this estimate to be off by 10%, then our final estimate for B will be off by much greater than 10% if the slope ratio is close to one, and less than 10% error if the slope ratio is much greater than 3 (remember, our estimate for B has to take the logarithm of these slope ratios).

Suppose we have not traveled very far; so the change in slope is only 10% (our first change was -2.2 per second, and our last change is -2.0). Then our slope ratio without any noise is only -2.2/-2.0 = 1.1. If our second difference measurement (-2.0) is off by 10%, then it could be -1.8 or -2.2. If it is -2.2, then our error in estimating B would be 100% (the logarithm of 1.1 is 0.095, and the logarithm of 1.0 = 0). If the error causes the second measurement to be -1.8, then the final error in estimating B will be 111% off.

Therefore, we need to measure these changes in slope over regions that cover large enough periods of time (or any X axis variable) so that the change in slope is maximized, but not so large a period of time that one of the difference measurements (a single slope

measurement) is greatly affected by noise. Therefore curve fit the exponential curve before it gets to the "settled" region where the noise is affecting the slope estimator.

Generalization & summary of overlapping linear fit method. Use the observation that the ratio of the changes in Y_i can be made dependant only on B. Example:

$$Y_i - Y_{i+1} = Ae^{Bt_i} - Ae^{Bt_{i+1}} = A(e^{Bt_i} - e^{Bt_{i+1}})$$

Therefore

$$(Y_k - Y_{k+\Delta t_1}) / (Y_i - Y_{i+\Delta t_2}) = (e^{Bt_k} - e^{Bt_{k+\Delta t1}}) / (e^{Bt_i} - e^{Bt_{i+\Delta t2}}) = e^{Bt_k}(1 - e^{B\Delta t_1}) / e^{Bt_i}(1 - e^{B\Delta t_2})$$

If we choose $\Delta t_1 = \Delta t_2$, then the equation simplifies to

$$(Y_k - Y_{k+\Delta t_1}) / (Y_i - Y_{i+\Delta t_1}) = e^{Bt_k}(1 - e^{B\Delta t_1}) / e^{Bt_i}(1 - e^{B\Delta t_1}) = e^{Bt_k} / e^{Bt_i} = e^{B(t_k - t_i)}$$

Which we can use to solve for B

$$B = ln[(Y_k - Y_{k+\Delta t_1}) / (Y_i - Y_{i+\Delta t_1})] / (t_k - t_i)$$

This derivation says the ratio of the changes in Y is totally dependant on B and Δt. It is the same answer we got using discrete calculus. This answer is not limited to the assumptions behind discrete calculus. But it still uses the mean value theorem of derivatives. Remember, if we let $\Delta t_1 = \Delta t_2$ the solution becomes easy. **This allows us to use overlapping regions (S_{R1} & S_{R2} in fig. 14) to measure the change in slopes.** This allows us to use regions of data where Y is changing substantially more than the noise. This avoids one of the major limitations of this new method. (The part of the curve that changes the most has the best signal to noise ratio for the slope estimator.)

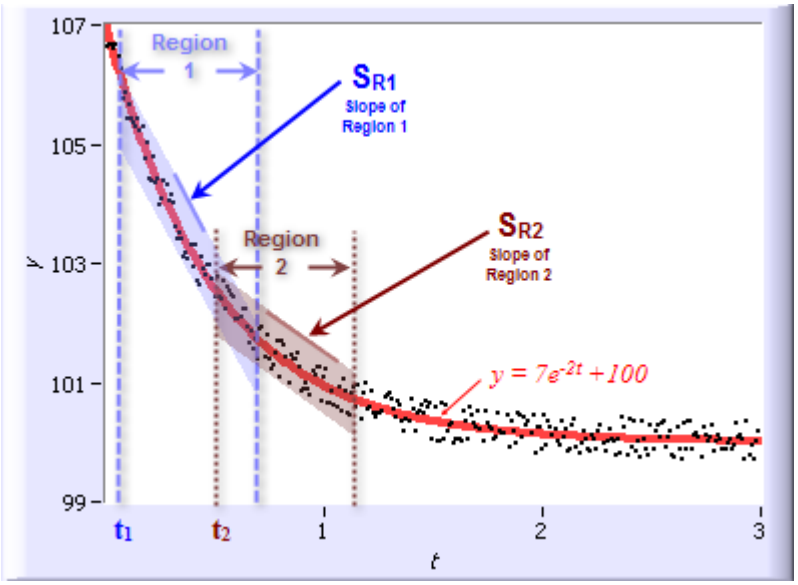

Figure 14: Overlapping Method – Fig 11 repeated

Once we know B, standard statistical methods can be used for A and C in equation 1. Since we now have an estimate for e^{Bt} for every value of "t_i", then let $K_i = e^{Bt_i}$, and equation 1 simplifies to

$$Y(t_i) = AK_i + C \qquad (43)$$

Using standard linear regression, we can use the Sum of the Square of the Errors (SSE) to derive an estimator for A. We will skip the step by step derivation since this is nothing new. (Any statistics book with linear curve fitting covers its derivation.)

$$\mathbf{A} = [\, n\sum \mathbf{K}_i \mathbf{Y}_i - \sum \mathbf{K}_i \sum \mathbf{Y}_i \,] \,/\, [\, n\sum \mathbf{K}_i^2 - (\sum \mathbf{K}_i)^2 \,] \qquad (44)$$

And it follows that

$$\mathbf{C} = [\, \sum \mathbf{Y}_i - \mathbf{B}\sum \mathbf{K}_i \,] \,/\, \mathbf{n} \qquad (45)$$

There are more ways to derive estimates for A and C, once we have an estimator for B. But this new technique is unique in its method for deriving an estimator for B (speed, stability, determinant processing time, and the removal of limits). See the benefits and limitations of using this new method on the following pages.

Benefits

1. Faster. This is a much faster algorithm (50x to 500x faster). When the comparison was made using National Instruments LabVIEW, this method averaged 500 times faster than the non-linear technique that comes with the LabVIEW analysis package (actually 510x to 658x). To be fair to LabVIEW, the nonlinear algorithm was generic and probably not as fast as it could be, if specifically designed for curve fitting an exponential. But if this paper's method(s) was specifically designed into LabVIEW, it also would be even faster. Even if the LabVIEW algorithm was designed specifically to fit an exponential, it would still take many iterations. The new method in this paper uses the equivalent of one (or two) iterations.

2. No bad guesses. There is no chance of making a bad first guess as sometimes happens in non-linear techniques. Sometimes the non-linear techniques "blow up" when they start with a bad guess.

3. Real-Time. Being faster would allow the curve fit to be used in real-time applications where non-linear methods take too much processing time. Many real-time applications require a consistent repeatable processing time from fit to fit, or at least a known processing time. Iterative techniques vary greatly in the processing time they consume. See #4.

4. Consistent. This method takes the same amount of processing time *(for the same number of data points)*, no matter what the error distribution is. Non-linear techniques sometimes settle quickly, or slowly depending on the data and initial guesses.

5. Y can cross Zero. Even if C = 0 and we use the standard curve fitting linearization method, all the data samples have to be greater than or equal to zero *(or transformed to greater than zero)*. This method does not have any such limitation.

6. PID set-point control. Some PID applications have an exponential relationship between time and the control variable *(especially after a sudden change in the set-point)*. As the control variable approaches the set-point, the algorithm could predict the final settled value and let the curve "smoothly" settle to the set-point. Therefore smoother control. Example, a sudden change in a system setting or set-point causes some control variables to change exponentially. Eventually the control variables will settle to some value. But, will that value be equal to the set-point? This method will predict the final settling point and the system can make the small corrections long before the control variable settles. It can even choose the amount of overshoot or minimize the rise time. Being able to predict the final settled value, without using as much processing time and using a known processing time, opens up other possibilities for PID control. PID control does not like timing loops that vary. (Fuzzy Logic can also benefit, since it is even more sensitive to the variations in its control loop timing.)

7. Control systems prediction. Long before the data has settled to some C = Y_n value, the algorithm can quickly detect changes in the A, B, and C to warn the operator that some other system change has occurred.

8. Seed for non-linear. For even more accurate curve fits, use this method to derive a first guess for each of the parameters A, B, C. This avoids any possibility of making a bad first guess, and therefore prevents large errors that would make a control system "blow up". If the appropriate non-linear technique is used, we can avoid violating the assumptions usually made in statistical curve fits[3]. See the non-Gaussian discussion. The non-linear technique will run much faster (far fewer iterations) if the first guess is already very close.

9. Detect overall instability quickly, and find hidden instability regions. This method can give us a quick estimator for B. If the B value is positive, then the system is unstable. (If B is positive, e^{Bt} will eventually become too large, no matter how small of a positive B we have.) We may also detect regions of instability. We may have never known that parts of our control system had unstable regions in areas where we previously thought the system was stable. Example: suppose when we change our control variable 10 percent of full scale, the system response changes in an unstable way (B > 0), but only for the first 10% of its change. Then the system has some internal response that "dampens" out the unstable portion AFTER the system response changes more than 15%. Therefore detecting the unstable region may help designers find those physical areas of their system that are causing instabilities at other set-points. Removing these instability areas may let the system designers run their control system at a more efficient level. Figure 15 shows that the unstable region stops after a 40% change in the control variable's response. If the plot showed this change after 10% or 20% it would be difficult for you to detect the unstable region on this plot. But, the PID controller, with this 'B' estimator, could possibly detect these small unstable regions that might be missed by the human eye.

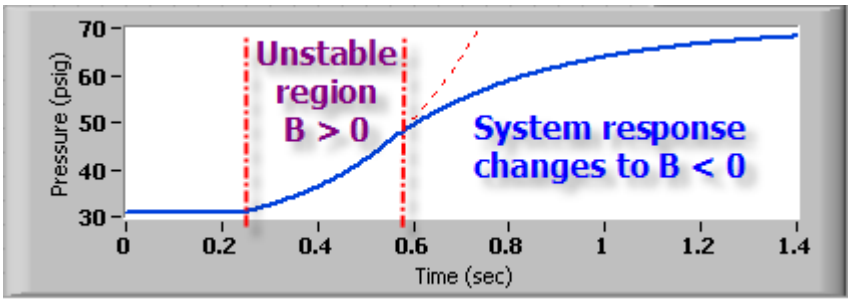

Figure 15: Detect Unstable Regions

10. Detecting Set-Point Instability May Lead To Greater Efficiency. As you change the set-point, you may reach an unstable point. It would be nice to know exactly where the process variable

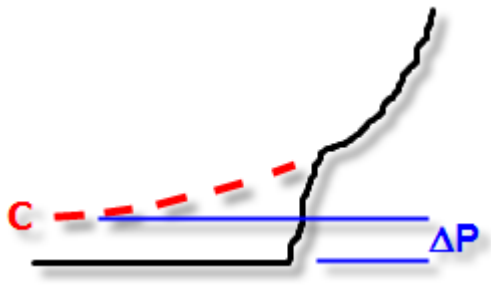

Figure 16: Where Would Instability Start?

"started" to go unstable. Then you could control your system to stay just below this unstable point (if B>0 then the function blows up or reaches system instability). It may be useful to know what set-point value that this blow up would have started from. The control system could have stayed just below this critical set-point and gained another ΔP increase in the process variable (Figure 16). Therefore, this method could help "tune" the feedback system between the unstable limits.

11. Even Obscure Applications. This method can also be used in radiometric dating because the final value is not zero in some radiometric methods. Here are two examples. Both examples reference figure 17. **Example #1.** Most people have heard of Carbon 14 dating. One of the major assumptions, behind the longer dates, is that the amount of C_{14} in the atmosphere is constant *(more precisely stated as: the ratio between C_{14} and C_{12} is assumed to have reached equilibrium millions of years ago.)*. Willard Libby, the winner of the Nobel Prize for his discovery of C_{14} dating, tested this assumption. His assumption was that the amount of C_{14} being produced in the upper atmosphere has been equal to the amount of C_{14} turning back into N_{14}, for millions of years. He was surprised at his results. *(We're not in equilibrium yet: C_{14} is still being produced faster than it is decaying.)* Therefore, this new curve fitting technique could also be applied to dating the maximum age of the atmosphere. Note: C_{14} in the atmosphere would be constant after about five C_{14} half-lives *(after subtracting changes due to nuclear explosions and industrial burning of carbon fuels, by using samples that were before these effects)*. In other words, if the Earth's atmosphere is younger than two half-lives, then this explains why C_{14} is still being produced much faster than it is decaying. **Example #2.** Accumulation of helium in the upper atmosphere: Since the Earth's crust contains many radioactive minerals that decay using beta decay, then these beta decay products should accumulate exponentially over time. This beta decay quickly picks up electrons and becomes a helium atom. This helium is escaping from the Earth's crust at a known rate. These high amounts of beta decay are required by evolutionists to be a heat source for the Earth. (Otherwise the Earth would be much cooler in 4.6 billion years.) The loss of helium in the upper atmosphere is approximated by an increasing exponential curve. This would cause a buildup of helium (Figure 17) over the billions of years speculated for the Earth's age. As a side note, the measured amount of helium is 2000 times smaller than it should be, if the evolutionists' source of heat and billions of years are correct. Therefore the estimator for time in equation 1 is limited to much less than 4.6 billion years.

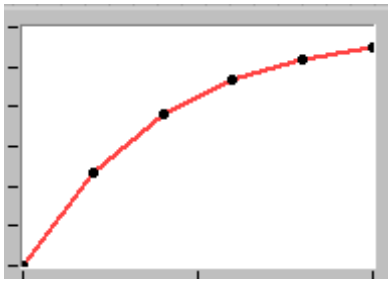

Figure 17: Exponential Build-up

12. Moving Exponential Curve Fit. Since this method is so much faster, we could curve fit data in real-time and see if the process we are measuring changes to a different exponential decay curve. We can see if the estimates for A, B, and C remain reasonably constant. Perhaps when some processes get past a "threshold",

they change to a different exponential decay curve. This method could detect the change and measure it in real time. This moving curve fit should see its estimates (for A, B, and C) converging as more data is accumulated to use in the curve fit. The estimate for a value generally has a standard deviation equal to the standard deviation of a single point, divided by the square root of the number of samples used in the estimate. The standard deviation in our estimates (for A, B, and C) should go down as we get more and more data. Be careful about this assumption for slopes in the region where the data is not changing very much.

Limitations:
What are some of the limitations of this new method?

1. This new method requires getting an estimate for B, before estimating A or C. Therefore the estimates for A and C are not independent of B. In other words, the error in the estimate for B biases the A estimate. But these errors do not add. The combined error is actually less when used to estimate C. The errors in A and B seem to "correct each other" to give a better estimate for C.

2. This method still has the bias due to using the sum of the square of the errors (SSE) method. This bias causes a non-Gaussian distribution of the errors. Remember, the SSE method assumes a Gaussian distribution of the errors. This bias error is usually so small that it is ignored.

3. Taking the logarithm of the slope ratios also causes the distribution of the errors to move further away from a Gaussian distribution as assumed in the SSE method.

4. You must have enough real change in your data, to get a good estimate for the change in the slope from one region to the next. You must also choose linear fit regions where this slope estimate is not greatly affected by noise. Therefore both regions must be in a part of the exponential curve where the data is changing significantly. My first estimate would be to use overlapping regions where the change in one region is about half the change in the other region.

First Order Differential Equations. Exponential relationships often come from the solution to a first order differential equation of the form

$$dY/dx = -kY \tag{46}$$

Notice that this differential equation only gives you the relationship of the change in Y as a function of the present value of Y. Many processes can be modeled as a first order differential equation. Radiometric dating techniques often rely on deriving an estimator

for time using exponential decay. Many times the solution does not have C equal to zero. The solution to this first order differential equation is the familiar exponential equation:

$$Y = Ae^{-kt} \tag{47}$$

If all the data values are greater than zero, then this new method or the standard exponential curve fitting algorithm can be used. (There is no disadvantage in using this new method.) It requires about the same amount of processing time as the standard exponential curve fitting method. But if any of the data values are less than zero while other data values are greater than zero, then the standard method cannot be used (equations 2-18).

Some physical models fit this type of equation (47). Example: many population growth curves have this exponential relationship.[9] The growth in our human population grows roughly exponentially. Note: Humans have not reached their Malthusian food supply limit in most areas. So take the historical human population growth rate (about 0.5% to 1% per year) and figure out a rough estimate for how long ago the supposed "African Eve" lived. (If we use 0.5% growth per year for 4500 years we end up with today's population of 6 billion people.) Then take a ten times smaller growth rate and estimate again. (If we use 0.05% for 45,000 years we get the same 6 billion people.) We can also derive an exponential equation for the number of people who have lived and died if humans have been here for even one million years. Even a growth rate ten times smaller than today's rate, would give far too large of an answer for how many humans have lived and died on this earth. (20,000 generations – 40 years per generation - with only 1% growth per generation would give you far more humans than the number of atoms in the universe.) Notice that these uniformitarianism assumptions do not work if they assume millions of years for humans on this planet. This just shows us that exponential models may give us results that many do not want to accept. Picking and choosing when to use exponential models can be difficult. As humans we can be biased as to when and where we apply exponential curve fits to our data.

In summary, this new method can be the general method for solving any exponential equation. The stable region does not have to be zero (C = 0), and the data points do not have to all be positive, or all be negative. Many areas of life have data where the exponential equation models the results we see. Even the general geometric growth function can be reduced to an exponential curve where this method can be applied.

Methods that almost work as well

1. Break up the data into three equal regions *(same number of equally spaced data points in each region)*. Then simply take each region and add up the values (R_1, R_2, R_3). Then compute B using

   $B = -\ln[(R_1 - R_2) / (R_2 - R_3)] / (n/3)\Delta t$ where n = total number of equally spaced data points, and Δt is the time between data points.

   This is a very fast algorithm. No curve fitting sum of the square of the errors is needed to estimate B. Then use this estimate for B and utilize the previous methods for A and C. As long as two of these regions are not where the curve has already settled, this works quite well. It could also be used as a quick estimator for the non-linear method to avoid bad initial guesses.

   This also gives the exact value for B *(zero error)*, if there is no noise. It also works with any three equally spaced data values. You could take the first, middle and ending points and use them as a good estimator for B. This would be even faster. Since you would be using the first, middle, and ending points, you would avoid much of the problems associated with noise (assuming the signal to noise ratio is high enough).

2. Use the average of the forward and backward derivatives.

   The discrete first derivative can be estimated by taking the average of the forward and backward derivatives. This would require four regions (equally spaced apart in time). The estimate for the slope at region 2 would be the average of the forward and backward slopes on each side of region 2

   $dY_2/_{dt} = [(R_2-R_1)/\Delta t + (R_3-R_2)/\Delta t] / 2 =$ Average of forward & backward derivatives.

   $= (R_3-R_1) / 2\Delta t$

   The estimate for the slope at region 3 would be the average of the forward and backward slopes on each side of region 3

   $dY_3/_{dt} = [(R_3-R_2)/\Delta t + (R_4-R_3)/\Delta t] / 2$

   $= (R_4-R_2) / 2\Delta t$

   Then we use the same estimate for B as found in equation 28

   $B = \ln(dY_2/_{dt} / dY_3/_{dt}) / \Delta t$

3. Derive B from the Ratio of the 2nd derivative to the 1st derivative.

   Divide the second derivative by the first derivative: Perhaps we could also solve for B since $dY/_{dt} = ABe^{Bt}$ (Equation 20), we can also see that the second derivative is

   $dY^2/_{dt}{}^2 = AB^2e^{Bt}$

   Therefore $B = (AB^2e^{Bt}) / ABe^{Bt} = (dY^2/_{dt}{}^2)/ dY/_{dt}$

   Since the estimator for the second derivative and the first derivative have different time shifts for their estimate of Δt, this is only a good estimator if the data is still more than 30% away from the *"settled"* value *(the data has only changed less than 70% of what its total change will be: if B < 0)*. Even then, this should only be used to give the non-linear technique a good initial guess for B. See "Finite Difference" methods for some better estimators for the first and second derivatives. But, even higher derivatives will have the change in slopes calculated at a different time than the first derivative. *Remember B determines where between data points the real tangent line slope is equal to the secant line between the points.* If the data is still 70% away from the final settled value *(the data has only changed 30% of what its total change will be)*, this method gives a good estimate for B that is very close to the previous method. Therefore, you could limit the data used to get a good fast estimate for B.

4. A & C Without SSE. Using the same fast method in #1 for B, we could also compute A & C without using the sum of the square of the errors (SSE). SSE still takes more processing time than the following method *(but SSE is still much faster than the non-linear techniques)*. These fast methods give poor estimates for A and C because the point on the curve (usually time) where the curve equals the average value is not in the middle and is a function of number of points, B, and A. But they are an even quicker way to make initial estimates for A and C *(to use in the standard non-linear iterative techniques)*. This would do two things:

   a. Avoid the rare unpredictable large errors associated with non-linear techniques *(due to a bad first guess)* by using this fast method as the initial guesses for A, B, and C.

   b. Use less iterations because this initial guess is much closer to the final estimates for A, B and C.

   This would give an estimate for the non-linear initial guesses faster than the main methods discussed in this paper *(faster because it avoids having to compute all the intermediate terms for the SSE)*. Here is the faster method for A and C. Since we have already computed the values for R_1, R_2, and R_3, use these values to estimate A & C (like page 30, Method 1). We do not need all three values. We only need two. But, the best two regions to use for A are the two adjacent regions, which differ the most. In other words, compute $D_{12} = R_1 - R_2$, and $D_{23} = R_2, - R_3$.

If $D_{12} > D_{23}$, then use R_1 and R_2 to estimate A. Otherwise use R_2 and R_3. Then use the opposite pair or opposite region to estimate C after you derive A. Example, suppose $D_{12} > D_{23}$. Then let

$K_1 = e^{Bt_1}$ t_1 is estimated by using the middle of region one.
$K_2 = e^{Bt_2}$
$K_3 = e^{Bt_3}$
$r_1 = R_1/n_1$ which is the average value in region one
$r_2 = R_2/n_2$
$r_3 = R_3/n_3$

where R_1 is the sum of all the data points in region 1, and
$n_1 = n_2 = n_3$ = number of data points in one region
Then we can estimate A and C by using

$r_1 = AK_1 + C$
$r_2 = AK_2 + C$
$r_1 - r_2 = A(K_1 - K_2)$
$A = (r_1 - r_2) / (K_1 - K_2)$
$r_3 = AK_3 + C$ opposite region
$C = AK_3 - r_3$

This faster method also avoids many of the problems associated with noise. Since each region is averaged, the noise effect is greatly reduced. The reduction in noise usually follows the function of the square root of the number of samples. In other words, if you have 100 samples for each region, the noise will be reduced by a factor of 10.

There are many more ways to derive estimates for B using the rules of calculus or other mathematical techniques.

Technical References

1. **Weisstein, Eric W.** *"Least Squares Fitting."* From *MathWorld*--A Wolfram Web Resource. "The sum of the squares of the offsets is used instead of the offset absolute values because this allows the residuals to be treated as a continuous differentiable quantity. However, because squares of the offsets are used, outlying points can have a disproportionate effect on the fit, a property which may or may not be desirable depending on the problem at hand."
http://mathworld.wolfram.com/LeastSquaresFitting.html
LAST MODIFIED: September 16, 2002

2. **H.J. Motulsky and A Christopoulos**. *"Fitting models to biological data using linear and nonlinear regression. A practical guide to curve fitting."* 2003, page 95, GraphPad Software Inc., San Diego CA, http://www.graphpad.com/curvefit/how_nonlin_works.htm

3. **Ledvi, Marko** *"Curve Fitting Made Easy"*, From April/May 2003 *"The Industrial Physicist"*, pg 24, "Hence, log(Y) is a linear function of X. This means that one could try to fit the logarithm of Y data to a linear function of X data to find A and X0. This approach was widely used before the age of computers. There are problems with it, however, including the fact that the transformed data usually do not satisfy the assumptions of linear regression. Linear regression assumes that the scatter of points around the line follows a Gaussian distribution, and that the standard deviation is the same at every value of the independent variable. To find the values of the model's parameters that yield the curve closest to the data points, one must define a function that measures the closeness between the data and the model. This function depends on the method used to do the fitting, and the function is typically chosen as the sum of the squares of the vertical deviations from each data point to the curve. (The deviations are first squared and then added up to avoid cancellations between positive and negative values.) This method assumes that the measurement errors are independent and normally distributed with constant standard deviation." Also available at: http://www.aip.org/tip/INPHFA/vol-9/iss-2/p24.html

4. ibid, pg 24, "**Nonlinear models**. In most cases, the relationship between measured values and measurement variables is nonlinear. Nonlinear curve fitting also seeks to find those parameter values that minimize the deviations between the observed *y* values and the expected *y* values. In nonlinear models, however, one usually cannot solve the equation for the parameters. Various iterative procedures are used instead. Nonlinear fitting always starts with an initial guess of parameter values. One usually looks at the general shape of the model curve and compares it with the shape of the data points."

5. **Pezzullo , John C**., Interactive Statistics page, "But we often encounter functions that cannot be linearized by any such tricks, a simple example being exponential decay that levels off to some unknown value: *y=a*Exp(-b*x)+c*. Applying a logarithmic transformation in this case produces *Log(y-c)=a'-b*x*. This linearizes *b*, but now *c* appears inside the logarithm; either way, we're stuck with an intrinsically nonlinear parameter estimation problem, which is considerably more difficult than linear curve-fitting." http://www.statpages.org/nonlin.html

6. ibid, **Background Info (just what is nonlinear curve-fitting, anyway?):** Simple **linear** curve fitting deals with functions that are **linear in the parameters**, even though they may be **nonlinear in the variables**. For example, a parabola *y=a+b*x+c*x*x* is a **nonlinear function of x** (because of the *x*-squared term), but fitting a parabola to a set of data is a relatively simple **linear curve-fitting problem** because the parameters enter into the formula as simple multipliers of terms that are added together. Another example of a linear curve-fitting problem is *y= a+b*Log(x)+c/x*; the terms involve nonlinear functions of the independent variable *x*, but the parameters enter into the formula in a simple, linear way.

   Unfortunately, many functions that arise in real world situations are **nonlinear in the parameters**, like the curve for exponential decay *y=a*Exp(-b*x)*, where *b* is "wrapped up" inside the exponential

function. Some nonlinear functions can be linearized by transforming the independent and/or dependent variables. The exponential decay curve, for example, can be linearized by taking logarithms: *Log(y)=a'-b*x* . The *a'* parameter in this new equation is the logarithm of *a* in the original equation,so once *a'* has been determined by a simple linear curve-fit, we can just take its antilog to get *a*.

But we often encounter functions that cannot be linearized by any such tricks, a simple example being exponential decay that levels off to some unknown value: *y=a*Exp(-b*x)+c*. Applying a logarithmic transformation in this case produces *Log(y-c)=a'-b*x*. This linearizes *b*, but now *c* appears inside the logarithm; either way, we're stuck with an intrinsically nonlinear parameter estimation problem, which is considerably more difficult than linear curve-fitting. That's the situation this web page was designed to handle.

For a more in-depth treatment of this topic, check out Dr. Harvey Motulsky's new web site: Curvefit.com -- a complete guide to nonlinear regression. Most of the information here is excerpted from *Analyzing Data with GraphPad Prism*, a book that accompanies the program *GraphPad Prism*. You can download this book as a pdf file.

7. **Wikipedia**® , Wikimedia Foundation, Inc., **Calculus of Finite Differences**, "A **finite difference** is a mathematical expression of the form $f(x + b) - f(x + a)$. If a finite difference is divided by $b - a$, one gets an expression similar to a differential quotient, except that it uses finite quantities instead of infinitesimal ones. The approximation of derivatives by finite differences plays a central role in finite difference methods for the numerical solution of partial differential equations. … Another important aspect is that finite differences approach differential quotients as *h* goes to zero. Thus, we can use finite differences to approximate derivatives. This is often used in numerical analysis, especially in numerical ordinary differential equations and numerical partial differential equations, which aim at the numerical solution of ordinary and partial differential equations respectively. The resulting methods are called *finite difference methods*." The full text is available at: http://en.wikipedia.org/wiki/Finite_difference

8. **D.E.Dirkse**, Castricum the Netherlands. http://home.hccnet.nl/david.dirkse/math/exp/expfunc.html
This looks like the private web site of Mr. Dirkse. His approximation for 'b' will fail many times due to noise in real world data. This is because you cannot take the logarithm of a negative number and his approximation uses the ratio of **adjacent slopes** to derive a first guess for his 'b'. This is probably why no company I have found uses his method as a first guess for 'b'. An excerpt from his website follows:
This article describes the exponential curve fitting method implemented in Graphics-Explorer.….Given a set of points (x_i , y_i) , (i = 1..n) in a plane, the user may select an exponential function of type

$y = ab^x$
$y = ae^{bx}$
$y = ab^x + c$
$y = ae^{bx} + c$

crossing these points with a,b,c calculated for minimal deviation. Finding a,b,c requires the following steps:

1. find an **approximation of b**
2. using result of 1. find an approximation of a
3. using result of 2. find an approximation of c
4. fine tune c
5. rebuild function to $y - c = ae^{bx}$
6. rebuild function to $\ln(y - c) = \ln(a) + bx$
7. temporarily replace $\ln(y - c)$ by y , $\ln(a)$ by a
8. use least-squares method to find best fitting polynomial degree 1, $y = a + bx$
9. enter function in Graphics-Explorer editwindow, translate and plot

… The idea is to differentiate $y = ae^{bx} + c$. In the derivative $y' = abe^{bx}$,c is eliminated. We guess, that the slope dy/dx between two adjacent points is most accurate **in the middle** between the x-coordinates, so x is modified accordingly.
Approximation of b
Working with the derivative we can write the system of equations:

$$y_1' = abe^{bx}{}_1$$
$$\cdots\cdots\cdots\cdots$$
$$y_{n-1}' = abe^{bx}{}_{n-1}$$

for 2 adjacent points i and j = i+1 in the dxy table:

$$\frac{y'_j}{y'_i} = \frac{a\,b\,e^{b\,x_j}}{a\,b\,e^{b\,x_i}}$$

$$ln\left(\frac{y'_j}{y'_i}\right) = b\,(x_j - x_i)$$

$$b = \frac{ln\left(\frac{y'_j}{y'_i}\right)}{x_j - x_i}$$

(end of excerpt) An **average value of b is calculated**. *(See "*Middle terms are thrown out*", see equations 30 through 32 of this new technology report for a discussion on why this is not really an average. Also see page 15, "Problem #1", for a discussion on why the middle between two data points is not the correct estimator for the time when the secant slope is the real slope.)*

9. **StatSoft**: http://www.statsoft.com/textbook/stnonlin.html Intrinsically Nonlinear Regression Models Some regression models which cannot be transformed into linear ones, can only be estimated via *Nonlinear Estimation*. In the growth rate example above, we purposely "forgot" about the random error in the dependent variable. Of course, the growth rate is affected by very many other variables (other than time), and we can expect a considerable amount of random (*residual*) fluctuation around the fitted line. If we add this *error* or residual variability to the model, we could rewrite it as follows:

   Growth = $\exp(-b_1*\text{Age}) + \text{error}$

   Additive error. In this model we assume that the error variability is independent of age, that is, that the amount of residual error variability is the same at any age. Because the error term in this model is additive, you can no longer linearize this model by taking the logarithm of both sides. If for a given data set, you were to log-transform variable *Growth* anyway and fit the simple linear model, then you would find that the residuals from the analysis would no longer be evenly distributed over the range of variable Age; and thus, the standard linear regression analysis (via *Multiple Regression*) would no longer be appropriate. Therefore, the only way to estimate the parameters for this model is via *Nonlinear Estimation*.

10. **Travis Bolden -** Morehouse College, **Beverly Gonzalez -** University of Illinois at Urbana-Champaign**, Rebecca Parker -** University of Georgia at Athens**,** July 26, 2002**,** *Derivative Approximations on Time Scales*, http://jwilson.coe.uga.edu/emt668/emat4680.2000/parker.rebecca/Official%20Webpage/VIGRE/VIGRE/ThePaper.pdf