As an example consider a fit to an exponential where the x values range from 100 to 101. We temporarily offset the x values by 100, perform the fit and then restore the x values by adding 100. When we did the fit, rather than fitting to k0+k1*exp(-k2*x) we really did the fit to c0+c1*exp(-c2*(x-100)). A little rearrangement and we have c0+c1*exp(-c2*x)*exp(c2*100). Comparing these expressions, we see that k0= c0, k1= c1*exp(c2*100) and k2= c2.

A better solution to the problem of fitting exponentials with large X offsets is to use the built-in exp_XOffset and dblexp_XOffset fit functions. These fit functions automatically incorporate the X shifting; see **Built-in Curve Fitting Functions** on page III-206 for details.

The same problem can occur when fitting to high-degree polynomials. In this case, the algebra required to transform the solution coefficients back to unoffset X values is nontrivial. It would be better to simply redefine your problem in terms of offset X value.

Curve Fitting Troubleshooting

If you are getting unsatisfactory results from curve fitting you should try the following before giving up.

Make sure your data is valid. It should not be all one value. It should bear some resemblance to the function that you're trying to fit it to.

If the fit is iterative try different initial guesses. Some fit functions require initial guesses that are very close to the correct solution.

If you are fitting to a user-defined function, check the following:

- Your coefficients wave must have exactly the same number of points as the number of coefficients that you actually use in your function unless you hold constant the unused coefficients.
- Your initial guesses should not be zero unless the expected range is near 1.0 or you have specified an epsilon wave. See **The Epsilon Wave** on page III-267 for details.
- Ensure that your function is working properly. Try plotting it over a representative domain.
- Examine your function to ensure all your coefficients are distinguishable. For example in the fragment (k0+k1)*x, k0 and k1 are indistinguishable. If this situation is detected, the history will contain the message: "Warning: These parameters may be linearly dependent:" followed by a line listing the two parameters that were detected as being indistinguishable.
- Because the derivatives for a user-defined fit function are calculated numerically, if the function depends only weakly on a coefficient, the derivatives may appear to be zero. The solution is to create an epsilon wave and set its values large enough to give a nonzero difference in the function output. See **The Epsilon Wave** on page III-267 for details.
- A variation the previous problem is a function that changes in a step-wise fashion, or is "noisy" because an approximation is used that is good to only a limited precision. Again, create an epsilon wave and set the values large enough to give nonzero differences that are of consistent sign.
- Verify that each of your coefficients has an effect on the function. In some cases, a coefficient may have an effect over a limited range of X values. If your data do not sample that range adequately the fit may not work well, or may give a singular matrix error.
- Make sure that the optimal value of your coefficients is not infinity (it takes a long time to increment to infinity).
- Check to see if your function could possibly return NaN or INF for any value of the coefficients. You might be able to add constraints to prevent this from happening. You will see warnings if a singular matrix error resulted from NaN or INF values returned by the fitting function.
- Use a double-precision coefficient wave. Curve fitting is numerically demanding and usually works best if all computations are done using double precision numbers. Using a single-precision coefficient wave often causes failures due to numeric truncation. Because the **Make** operation defaults to single precision, you must use the /D flag when creating your coefficient wave.
- If you use intermediate waves in your user-defined fit function, make sure they are all double precision. Because the **Make** operation defaults to single precision, you must use the /D flag when creating your coefficient wave.