Subject: "Numerical Recipes" implementation complaints
Posted by Wayne Landsman on Fri, 25 Apr 1997 07:00:00 GMT
View Forum Message <> Reply to Message

RSI is very careful when implementing the "Numerical Recipes in C" functions to not modify the original C code. This is a good idea in principle -- it means, for example, that users can use the NR book to examine the IDL source code. But I think that when the code requires a user-supplied function, then the required function should be IDL-like and not C-like. Below are two examples:

1. The intrinisc QSIMP() function will integrate a user-supplied function. This function must "accept a single *scalar* argument X and return a *scalar* result (!!!)." It goes against all my IDL training to write a function that does not accept a vector argument! More important, it means that the intrinisc QSIMP() function is usually slower -- sometimes much slower -- than a vectorized version coded in the IDL language (such as available at http://idlastro.gsfc.nasa.gov/ftp/pro/math/qsimp.pro). This is because, say, on the 16th iteration of the integration, the intrinsic QSIMP function must make 32768 calls to the user-supplied function, whereas a vectorized version makes one call with a 32768 element vector.

2. In IDL V5.0B5, the Numerical Recipes version of the Levenberg-Marquardt non-linear least squares fitting algorithm is introduced as the intrinsic function LMFIT(). This function seems to be more robust than CURFIT(), and I especially like the FITA keyword, which lets users designate parameters as either fixed or free. But LMFIT() wants the user-supplied function to return both the function value and derivative *in a single vector*. This differs from CURVEFIT which wants a user-supplied function to return the function value, and optionally the derivative, in parameters. The ability to make the derivative computation optional is what makes the CURVEFIT function IDL-like. Some consequences of the LMFIT() choice of fitting function are:

(1) If we want to compare the CURVEFIT() and LMFIT() fitting results, then we must write two different functions to do the same thing.

(2) The /NODERIVATIVE option -- to automatically compute derivatives numerically -- in CURVEFIT is not possible in LMFIT(). I usually use non-linear fitting to fit numerical models to data, and it is not possible to compute analytic derivatives. In CURVEFIT() I could just set the /NODERIVATIVE keyword, and let the calling program do the numerical derivatives, whereas for LMFIT() I must write the numerical derivative computation inside each function.

(3) If I want to use a function for other purposes than LMFIT, then I certainly do not want to have to compute the derivatives each time I need the function value. (Yes, I know I could add a /NODERIVATIVE keyword to the function, so that it would return garbage in the derivative indicies of the output vector, but this is not aesthetically very pleasing.)

Wayne Landsman landsman@mpb.gsfc.nasa.gov