Subject: Re: Trouble with MPFITFUN
Posted by Craig Markwardt on Thu, 12 Apr 2012 12:35:06 GMT
View Forum Message <> Reply to Message

On Thursday, April 12, 2012 4:48:44 AM UTC-4, Helder wrote:
> On Thursday, April 12, 2012 7:10:08 AM UTC+2, Craig Markwardt wrote:
>> On Wednesday, April 11, 2012 12:34:23 PM UTC-4, Helder wrote:
>>> Hi,
>>> I've been spending a bit too much time on this and I am wondering what is going wrong here.
>>> I'm trying to fit using a step function broadened by a Gaussian.
>>> The fitting function is:
>>>
>>> FUNCTION GaussStep, X, P
>>> ;Calculate the broadening of a step function with:
>>> ;P[0] = step position
>>> ;P[1] = left value
>>> ;P[2] = right value
>>> ;P[3] = step width
>>> PRINT, P
>>> P[0] = (P[0] > MIN(X)) < MAX(X)
>>> Y = DBLARR(N_ELEMENTS(X))
>>> LowIndeces = WHERE(X LT P[0], CountLow, COMPLEMENT = HighIndeces,
NCOMPLEMENT=CountHigh)
>>> IF CountLow GT 0 THEN Y[LowIndeces] = P[1]
>>> IF CountHigh GT 0 THEN Y[HighIndeces] = P[2]
>>> Sigma=P[3]
>>> nPts=10*Sigma+1.0
>>> kernel=DINDGEN(nPts)-(nPts-1)/2.0
>>> kernel=EXP(-kernel^2/(2.*sigma^2))
>>> kernel/=TOTAL(kernel,/DOUBLE)
>>> yconvol = CONVOL(Y,kernel,/EDGE_TRUNCATE)
>>> RETURN, yconvol
>>> END
>>>
>>> To test MPFITFUN I use the following code:
>>> PRO TestFit
>>> xData = DINDGEN(201)
>>> yData = DBLARR(201)+RANDOMU(SEED,201,/DOUBLE)*0.2-0.1
>>> yData[150:200] += 1.0D
>>> StParam = [148D,MIN(yData),MAX(yData),3D]
>>> DataErr = DBLARR(N_ELEMENTS(xData))+0.2D
>>> Results = MPFITFUN('GaussStep', xData,yData, DataErr, StParam, STATUS=status, /quiet)
>>> PLOT, xData, yData
>>> OPLOT, xData, GaussStep(xData,Results), COLOR = 255L
>>> PRINT, 'Final Parameters = ', Results
>>> PRINT, 'Start Parameters = ', StParam
>>> END
>>>

>>> The output shows all the calls of the fitting function. And I find that at the end there is always NO change in the first parameter. Here is an example of the output:
>>>
>>> 148.00000 -0.099990073 1.0994661 3.0000000
>>> 148.00000 -0.099990073 1.0994661 3.0000000
>>> 148.00000 -0.099990071 1.0994661 3.0000000
>>> 148.00000 -0.099990073 1.0994661 3.0000000
>>> 148.00000 -0.099990073 1.0994661 3.0000000
>>> 148.00000 0.0073445709 1.0082363 2.3488363
>>> 148.00000 0.0073445709 1.0082363 2.3488363
>>> 148.00000 0.0073445710 1.0082363 2.3488363
>>> ...
>>> 148.00000 -0.0039705287 0.99188729 2.0999998
>>> 148.00000 -0.0039705257 0.99188729 2.1000000
>>> 148.00000 -0.0039705254 0.99188729 2.1000000
>>> 148.00000 -0.0039705254 0.99188729 2.1000000
>>> Final Parameters = 148.00000 -0.0039705254 0.99188729 2.1000000
>>> Start Parameters = 148.00000 -0.095071379 1.0978406 3.0000000
>>>
>>> Throughout all the fitting procedure the first parameter has never been changed.
>>>
>>> Am I doing something terribly wrong? I generally have no estimates for the errors in the data, therefore I used 0.1. In the example data this is easy to calculate, but the fitting has to be applied to the most different data sets.
>>>
>>> I also tried playing with the XTOL parameter without any success.
>>>
>>> Any tips are appreciated.
>>>
>>> Many thanks,
>>> Helder
>>>
>>> PS: I tried lots of different initial conditions, I tried using "parinfo.fixed" to block the other parameters, ... but at the end I never get any change in P[0]... sigh..
>>>
>>> PSS: The function GaussStep is working fine... I can replot the data in the correct way by moving the parameters by hand.
>>
>> You are getting closer to the right track.
>>
>> If I were you, I would avoid complicated invocations of CONVOL. It looks like you can compute your "smoothed step function" exactly, by using the ERF (formerly ERRORF) function. I've used that before with success.
>>
>> ERF is much better than your convolution because it actually integrates the gaussian, rather than assuming that sampling a gaussian at a few discrete points is sufficient to integrate it.
>>
>> You might also want to play with using PARINFO, and setting the .STEP or .RELSTEP fields.

The fitter can get stuck if your peak position and/or step position is between data samples. Set the parameter step size to something close to your data grid sample size.
>>
>> Best wishes,
>> Craig Markwardt
>
> Thanks Craig,
> your tip was very useful. I never thought about the ERF!
> Here is the function I am now using:
...
> So far it worked fine!
> Thanks again.
> I will see what I can do with the Parinfo parameters. I'm currently using the procedure to show live fitting whilst moving a cross section on an image and it seems stable enough... Fits edges also where one would not see one.

I'm glad that it's working. The reason I suggested PARINFO is that the fitter can get stuck. If it tries to make the step too narrow (P[2] -> 0) during its iterations, then it will have a hard time getting out of that condition in later iterations. A step function with width 0.0001x(grid spacing) gives essentially the same chi-square as a step function with 0.00010001x(grid spacing), so the partial derivative goes to zero, or is noisy.

By setting up PARINFO.STEP, you can make sure that MPFIT takes large enough steps for partial derivatives.

Craig
