Subject: Re: Finding coefficients in multiple nonlinear equations using MPFIT
Posted by Craig Markwardt on Sat, 29 May 2010 06:33:20 GMT
View Forum Message <> Reply to Message

Greetings--

On May 29, 1:54 am, Junum <junshi...@gmail.com> wrote:
> Hi all,
>
> I got a problem that I have spent several days, but did not solve.
> I have two nonlinear equations.
> Equation 1 has 6 coefficients (c1 through c6) and Eq. 2 has 5
> equations (d1 through d5).
> Eq. 1 : c1 * A * B^c2 + c3*C^c4 + c5*exp(D^c6) = 0
> Eq. 2 : d1*(A/B)^d2 + d3*(1/C)^d4 = 0
> Actually, original equations are much complex.
>
> From lab experiments I have data sets of A, B, C, and D, more than
> 40000.
> My goal is to find optimum coefficients using these large data sets.
> I've tried to use MPFIT, but it seems that MPFIT has limit to define a
> function.
> What I did is
>
> expr = '['
> st = 31
> en = 44
> FOR j=st,en DO BEGIN
> s_L = STRING(FORMAT='(D9.3)',L[j]) + 'D'
> s_Lapp = STRING(FORMAT='(D9.3)',Lapp[j]) + 'D'
> s_Z = STRING(FORMAT='(D11.6)',Z[j]) + 'D'
> s_foc = STRING(FORMAT='(D8.3)',foc[j]) + 'D'
> s_rnd = STRING(FORMAT='(D8.3)',round[j]) + 'D'
> s_bg = STRING(FORMAT='(D11.5)',bg[j]) + 'D'
>
> exp1 = '(-1.*' + s_foc + ')'
> exp2 = '((ABS(' + s_L + '/' + s_rnd + '))^P[5])'
> exp3 = '(' + s_bg + ')'
> exp4 = '(-1.*ABS(P[1]*(' + exp3 + '^P[3])))'
> exp5 = '(ABS(' + s_Z + ' + ABS(' + s_Z + ')*P[2])^P[4])'
> exp6 = '(EXP(' + exp4 + ' * ' + exp5 + '))'
>
> expr1 = exp1 + ' + ' + exp2 + ' * ' + exp3 + ' * ' +
> exp6 + ' * P[0]'
>
> exp7 = '(-1.*(' + s_Lapp + '/' + s_L + '))'
> exp8 = '(' + s_Z + '/((' + s_L + '*'+ s_rnd + ')^(2.)))'
> exp9 = '(ABS(' + exp8 + '))'

```
>          exp10 = '( ' + exp8  + ' * P[7] )'
>          exp11 = '( (' + exp9 + ' + ' + exp10 +  ')^P[8] )'
>          exp12 = '( ((' + s_L + ')^P[10])*P[9] )'
>          exp13 = '( ' + exp11 + ' * ' + exp3 + ' )'
>
>          expr2 = exp7 + ' + ' + exp13 + ' * P[6] + ' + exp12
>
>          IF ( j eq st ) THEN BEGIN
>            expr  = expr + expr1 + ',  ' + expr2
>          ENDIF ELSE BEGIN
>            expr  = expr + ',  '  + expr1 + ', ' + expr2
>          ENDELSE
>
>     ENDFOR
>
>     expr = expr + '  ]'
>
>     fit_status = 0
>     num_iter = 0
>     resid = 0.D
>
>     st_gs = [-10., -10., -10., -10., -10., -10., -10., -10., -10.,
>  -10., -10. ]
>     P = mpfitexpr(expr, 1, 1, 0.1, st_gs, MAXITER=900)
>
>  14 (so 28 elements on defined function) is ok, but 15 (so 30 elements
>  on defined function) does not work.
>
>  How can I solve this problem?
>  Can any one help me?
```

You don't say how the code "does not work" so it is difficult to comment. The example is not complete, so I can't test it.

I think you need to define better what you are trying to do. I guess that you are trying to solve your two equations in a least squares sense using all the data jointly, but that is not clear.

Your equation has 11 coefficients, so I don't understand why 14 or 15 are important. If I misinterpreted your intent, and you want to solve each equation separately, then you can just run MPFIT*() 40000 times and achieve your goal. I don't know why you need to join them together into an expression. I doubt any IDL program will be able to solve an equation with 40000 parameters, if that is really what you are attempting.

MPFITEXPR() is not the ideal solution for your problem; it is better for quick jobs. MPFITEXPR has limitations because it needs to parse

the user expression many times. IDL cannot parse or evaluate infinitely long expressions. For solving equations, MPFIT() is the desirable method, especially since you have already expressed your equations as (function) = 0. That is exactly what MPFIT() solves.

If my initial suspicion was right, it should be as simple as something like this (untested),

```
 function myfunc, p, a=a, b=b, c=c, d=d
   return, [p[0]* a * b^p[1] + p[2]*c^p[3] + p[4]*exp(d^p[5]), $
           p[6] * (a/b)^p[7] + p[8]*(1/c)^p[9] ]
 end
```

The [...] notation will glue the two residual arrays together. MPFIT() will try to solve for parameters which drive all equations to zero.

```
p0 = [ ... initial guess of parameters ... ]
p = mpfit('myfunc', p0, functargs={a:a, b:b, c:c, d:d})
```

Your equations look a little troublesome. You may have to worry about whether they are linearly independent, and whether certain parameters will be strongly correlated. You also don't define whether the equations should be weighted equally or not. The above formulation gives equal weight.

Craig
