Subject: Re: Solution of Nonlinear System of Equations (BROYDEN etc.)
Posted by Craig Markwardt on Sun, 25 Aug 2002 13:34:14 GMT
View Forum Message <> Reply to Message

air_jlin@yahoo.com (Johnny Lin) writes:

> Hi all,
>
> I've this system of nonlinear equations that I want to solve
> iteratively and was planning on using BROYDEN to do it. What I'm
> wondering is how do I implement the function in BROYDEN so that I can
> change parameters on the fly. For instance, in the example in the
> help file, one of the equations is:
>
> 3x - COS(yz) - 1/2 = 0
>
> They then hard-wire the constants (3, 1/2, etc.) into BROYFUNC. But
> let's say I want to be able to change the 3x to ax where "a" is a
> constant I specify and change as the program runs. Is there a way to
> do this in a way the BROYDEN call will understand? I couldn't find
> the source code for BROYDEN to see if this is possible.

Hi Johnny--

One "solution" would be to use a common block to pass relevant parameters to your function. We all know that is a pretty bad solution, since it entails all of the ugliness associated with common blocks. The RSI-supplied numerical routines are pretty bad all around, when it comes to user-level flexibility.

One thing that people might not realize is that the MPFIT family of functions can also be used to solve systems of non-linear equations. Normally MPFIT is used to minimize the summed squared residuals in a curve fitting application. However it can also be used to minimize the error in a non-linear equation solution.

Here is an example using MPFITEXPR, but you could also use MPFIT directly. All you need to do is return the vector of non-linear function values. Here is an example based on the BROYDEN documentation. I have set X and Y to zero because the expression, EXPR, computes the residuals directly.

```
; Vector of non-linear functions to minimize
expr = '[3.0 * P[0] - COS(P[1]*P[2]) - 0.5, '+$
       ' P[0]^2 - 81.0*(P[1] + 0.1)^2 + SIN(P[2]) + 1.06, '+$
       ' EXP(-P[0]*P[1]) + 20.0 * P[2] + (10.0*!DPI - 3.0)/3.0]'
p0 = [-1.,1,2]
p = mpfitexpr(expr, 0, 0, 1, p0)
```

Now you may say to yourself, but Craig didn't answer my question! You wanted to know how to change the hard-coded values.

Possibility number 1 is simply to re-write the expression when you need to. This is much easier than trying to recompile an IDL function.

Possibility number 2 is to use FUNCTARGS to pass information to the expression. I have recently enhanced MPFITEXPR to allow this more usefully, and it is documented. Essentially, during execution of the expression, FUNCTARGS appears as a structure called PRIVATE which you can access in any way you wish.

Good luck!
Craig

--

---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------