

---

Subject: Re: Need help with an Iterative solution in IDL (relative newb question)  
Posted by [Chris\[6\]](#) on Thu, 14 Aug 2008 21:20:18 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Aug 14, 9:56 am, mbwel...@gmail.com wrote:

> On Aug 14, 11:50 am, Brian Larsen <balars...@gmail.com> wrote:

>

>

>

>> Matt,

>

>> this isn't anywhere near enough information to provide a coherent and  
>> meaningful answer.

>

>> - What exactly are you trying to do?

>> - What have you tried?

>> - What bits of code are working and not?

>

>> Cheers,

>

>> Brian

>

>> -----

>> Brian Larsen

>> Boston University

>> Center for Space Physics <http://people.bu.edu/balarsen/Home/IDL>

>

> Guess I should be more specific then :)

>

> Here is my code (non iterative):

> a= 3.6e007 ; area of region in meters^2

> o= (60\*!pi/180) ; fault dip angle in degrees

> c= 6e-003 ; scaling factor

> t= 50e003 ; elastic lithosphere thickness in meters

> v= (a\*t) ; volume of region in meters^3

> x= 5e003 ; depth of faulting in meters, 5-7km for normal

> faults, ~30km for thrust faults

>

> h= (x/sin(o)) ; depth of faulting in meters

> u= 3 ; fault aspect ratio: Length/Height(down dip)

> = 2 or 3

> kns=(sin(o)\*cos(o)/v) ; horizontal normal strain constant for small

> faults

> knl=(c\*cos(o)\*x^2/v/sin(o)) ; horizontal normal strain

> constant for large faults

> kvs=(-sin(o)\*cos(o)/v) ; vertical normal strain constant for small

> faults

> kvl=(-cos(o)/v) ; vertical normal strain constant for large

```

> faults
>
> ind_small = where(ar_plan[1,*] lt 2*x) ; select faults such that L
> < 2x
> ind_large = where(ar_plan[1,*] ge 2*x) ; select faults such that L> 2x
>
> ar_plan_small = ar_plan[:,ind_small] ; place in matrice with
> identifier
> ar_plan_large = ar_plan[:,ind_large] ; place in matrice with
> identifier
> lc_small= ar_plan_small[1,*] ; select only lengths to sum for
> small faults
> lc_large= ar_plan_large[1,*] ; select only lengths to sum for
> large faults
> tl_small = total(lc_small^3) ; sum lengths according to
> kostrov summation, small faults
> tl_large = total(lc_large) ; sum lengths according to kostrov
> summation, large faults
>
> ens= (kns*c/u)*tl_small ; horizontal normal strain
> for small faults
> enl= knl*tl_large ; horizontal normal strain for large
> faults
> e_t= ens+enl ; total horizontal normal strain
>
> I need to vary the parameters o,c,t,x and u with in a certain range
> (e.g. o= 50-80 degrees) in order to reproduce e_t (total horizontal
> normal strain) to within ~ +-10% and I need all the possible
> combination saved to an ascii file, or some other output. Where
> ar_plan is a FLOAT = Array[2, 129], different arrays have different
> dimensions and I have multiple arrays, but # of columns [2] should
> remain constant at this stage.
>
> I'm having some trouble getting started, but will probably have some
> issues in the implementation as well :)
>
> As an aside, I have another issue where, for example, ind_small = -1
> for no returned results instead of 0. This causes:
> % Attempt to subscript AR_PLAN with IND_SMALL is out of range and the
> program stops running.
> I would like this to run even with no returned results. Does anyone
> know how to do this?
>
> ~Matt

```

I think the main difficulty you are going to run into is that, with 5 independent variables, exhaustively searching the entire search space for solutions may not be feasible. The most straightforward approach, of

course, is to have five nested loops over each of your variables and checking to see if that combination of variables satisfies your constraint of reproducing  $e_t$ . However, even if you just tested 100 values for each variable, that would be  $10^{10}$  total steps in the loop. Furthermore, such an approach is extremely inefficient because it has no sense of 'how close' a given combination of variables are- it will spend the vast majority of the time checking ridiculous candidates.

There are a number of search algorithms that you could look into. Probably the easiest is some sort of monte carlo search like the following: Define a 'fitness function' for a combination of independent variables to be how far off the calculated  $e_t$  is from the goal  $e_t$ . You now want to minimize this error. Start with some random values for each of your variables, and use some local minimum finding algorithm (there is a built in amoeba function for 1 variable, but look into algorithms like steepest ascent hill climbing, downhill simplex, etc) to find a local error minimum. If the error is small enough, count that as an acceptable solution. If not, throw it away. Now start with new random values for the variables, and repeat. A book like Numerical Recipes by Press et al describes such algorithms.

The problem with this approach is that it is not guaranteed to find ALL acceptable combinations of values - that is only possible with an exhaustive search which is probably not feasible.

As for your problem of WHERE returning -1, use the count keyword in where. Then, test for whether or not that count is zero and, if it is, skip that case.

chris

---