Subject: Re: Need help with an Iterative solution in IDL (relative newb question)
Posted by Brian Larsen on Thu, 14 Aug 2008 18:50:30 GMT
View Forum Message <> Reply to Message

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

Subject: Re: Need help with an Iterative solution in IDL (relative newb question)
Posted by mbweller on Thu, 14 Aug 2008 19:56:43 GMT
View Forum Message <> Reply to Message

On Aug 14, 11:50 am, Brian Larsen <balar...@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 Physicshttp://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
identifer
ar_plan_large = ar_plan[*,ind_large]       ; place in matrice with
identifer
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 combintation 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

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 <balar...@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 Physicshttp://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
> identifer
> ar_plan_large = ar_plan[*,ind_large] ; place in matrice with
> identifer
> 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
> combintation 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 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

Subject: Re: Need help with an Iterative solution in IDL (relative newb question)
Posted by mbweller on Fri, 15 Aug 2008 00:45:09 GMT
View Forum Message <> Reply to Message

On Aug 14, 2:20 pm, Chris <beaum...@ifa.hawaii.edu> wrote:
> On Aug 14, 9:56 am, mbwel...@gmail.com wrote:
>
>
>
>> On Aug 14, 11:50 am, Brian Larsen <balar...@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 Physicshttp://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
>> identifer
>> ar_plan_large = ar_plan[*,ind_large] ; place in matrice with
>> identifer
>> 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
>> combintation 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 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

I'm trying to fix the where statement returning -1, here is what I've
tried thus far:

```
ind_small = where(ar_plan[1,*] lt 2*x,count)    ; select faults such
that L < 2x
if count ge 0 then ar_plan_small=ar_plan[*,ind_small] else
ar_plan_small=0
ar_plan_small
```

but I'm still getting the same error, I'm sure I have the syntax
wrong. Unfortunately I'm not quite at the level to trouble shoot this

myself, confidently.

I have ordered the book suggested, I would imagine that it would come in handy very soon, but for the shear learning experience of it I would like to try it in IDL first (plus research waits for no amazon order). I can limit the increments for each variable to make it more manageable (less than 10^10 total steps), I just need some help and/or examples to illustrate how to create five nested loops for each variable, with each bounded condition and set increment that satisfy e_t that are recorded to an ASCII file. e.g. o = 50-80, del o = 5; t=5-100, del t = 10; etc...

Thanks,
~Matt

Subject: Re: Need help with an Iterative solution in IDL (relative newb question)
Posted by Chris[6] on Fri, 15 Aug 2008 02:25:43 GMT
View Forum Message <> Reply to Message

On Aug 14, 2:45 pm, mbwel...@gmail.com wrote:
> On Aug 14, 2:20 pm, Chris <beaum...@ifa.hawaii.edu> wrote:
>
>
>
>> On Aug 14, 9:56 am, mbwel...@gmail.com wrote:
>
>>> On Aug 14, 11:50 am, Brian Larsen <balar...@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 Physicshttp://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
>>> identifer
>>> ar_plan_large = ar_plan[*,ind_large] ; place in matrice with
>>> identifer
>>> 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

>>> combintation 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 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
>
> I'm trying to fix the where statement returning -1, here is what I've
> tried thus far:
> ind_small = where(ar_plan[1,*] lt 2*x,count) ; select faults such
> that L < 2x
> if count ge 0 then ar_plan_small=ar_plan[*,ind_small] else
> ar_plan_small=0
> ar_plan_small
> but I'm still getting the same error, I'm sure I have the syntax
> wrong. Unfortunately I'm not quite at the level to trouble shoot this
> myself, confidently.
>
> I have ordered the book suggested, I would imagine that it would come
> in handy very soon, but for the shear learning experience of it I
> would like to try it in IDL first (plus research waits for no amazon
> order). I can limit the increments for each variable to make it more
> manageable (less than 10^10 total steps), I just need some help and/or
> examples to illustrate how to create five nested loops for each
> variable, with each bounded condition and set increment that satisfy
> e_t that are recorded to an ASCII file. e.g. o = 50-80, del o = 5;
> t=5-100, del t = 10; etc...
>
> Thanks,
> ~Matt

The where problem probably comes from the fact that you are selecting
indices from the sub-array ar_plan[1,*] but indexing the array
ar_plan[*,indsmall]. In other words, you select ROWS of interest (IDL
is column major, so array[i,j] is the ith column, jth row) and then
index those COLUMNS. If there are more rows than columns, you may get
an 'array index out of bounds' error. If you are still having issues,
try including the output of the following lines in your next post:

help,ar_plan
help,count
print,max(ind_small)
print,min(ind_small)

Also remember that IDL is zero-indexed so, if you are trying to access
the first column of something, you would use ar_plan[0,*] and not
ar_plan[1,*]

A clunky nested for loop for three variables looks something like this

openw,1,'output.dat'; this opens a file for writing
for a=alow, ahigh, astep do begin

```
   for b=blow, bhigh, bstep do begin
     for c=clow, chigh, cstep do begin
          if (f(a,b,c) ge goal-error) && (f(a,b,c) le goal+error)
then begin
             printf,1,a,b,c,format='(3f9.3)' ; records variables to
three decimal places
          endif
     endfor
   endfor
endfor

close, 1 ;close the file
```

here, f(a,b,c) is whatever combination of a b and c that's meant to reproduce the number goal to within the number error. the lows and highs are your lower and upper bondaries for a,b, and c, and the steps are how much to increment each time.

Please let me stress that this is not only an inefficient algorithm (it wastes time checking hopeless candidates), but one for which IDL will run very slowly (IDL hates extensive looping). Posting it here actually makes me feel a little dirty. I hope David Fanning doesn't see it...

chris

Subject: Re: Need help with an Iterative solution in IDL (relative newb question)
Posted by David Fanning on Fri, 15 Aug 2008 02:55:02 GMT
View Forum Message <> Reply to Message

Chris writes:

> Posting it here
> actually makes me feel a little dirty. I hope David Fanning doesn't
> see it...

I saw it, but since this case looks hopeless anyway, this is unlikely to be the cause of death. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Need help with an Iterative solution in IDL (relative newb question)
Posted by mbweller on Fri, 15 Aug 2008 04:54:45 GMT
View Forum Message <> Reply to Message

On Aug 14, 7:25 pm, Chris <beaum...@ifa.hawaii.edu> wrote:
> On Aug 14, 2:45 pm, mbwel...@gmail.com wrote:
>
>
>
>> On Aug 14, 2:20 pm, Chris <beaum...@ifa.hawaii.edu> wrote:
>
>>> On Aug 14, 9:56 am, mbwel...@gmail.com wrote:
>
>>>> On Aug 14, 11:50 am, Brian Larsen <balar...@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 Physicshttp://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
>>>> identifer
>>>> ar_plan_large = ar_plan[*,ind_large] ; place in matrice with
>>>> identifer
>>>> 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
>>>> combintation 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 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
>
>> I'm trying to fix the where statement returning -1, here is what I've
>> tried thus far:
>> ind_small = where(ar_plan[1,*] lt 2*x,count) ; select faults such
>> that L < 2x
>> if count ge 0 then ar_plan_small=ar_plan[*,ind_small] else
>> ar_plan_small=0
>> ar_plan_small
>> but I'm still getting the same error, I'm sure I have the syntax
>> wrong. Unfortunately I'm not quite at the level to trouble shoot this
>> myself, confidently.
>
>> I have ordered the book suggested, I would imagine that it would come

>> in handy very soon, but for the shear learning experience of it I
>> would like to try it in IDL first (plus research waits for no amazon
>> order). I can limit the increments for each variable to make it more
>> manageable (less than 10^10 total steps), I just need some help and/or
>> examples to illustrate how to create five nested loops for each
>> variable, with each bounded condition and set increment that satisfy
>> e_t that are recorded to an ASCII file. e.g. o = 50-80, del o = 5;
>> t=5-100, del t = 10; etc...
>
>> Thanks,
>> ~Matt
>
> The where problem probably comes from the fact that you are selecting
> indices from the sub-array ar_plan[1,*] but indexing the array
> ar_plan[*,indsmall]. In other words, you select ROWS of interest (IDL
> is column major, so array[i,j] is the ith column, jth row) and then
> index those COLUMNS. If there are more rows than columns, you may get
> an 'array index out of bounds' error. If you are still having issues,
> try including the output of the following lines in your next post:
>
> help,ar_plan
> help,count
> print,max(ind_small)
> print,min(ind_small)
>
> Also remember that IDL is zero-indexed so, if you are trying to access
> the first column of something, you would use ar_plan[0,*] and not
> ar_plan[1,*]
>
> A clunky nested for loop for three variables looks something like this
>
> openw,1,'output.dat'; this opens a file for writing
> for a=alow, ahigh, astep do begin
> for b=blow, bhigh, bstep do begin
> for c=clow, chigh, cstep do begin
> if (f(a,b,c) ge goal-error) && (f(a,b,c) le goal+error)
> then begin
> printf,1,a,b,c,format='(3f9.3)' ; records variables to
> three decimal places
> endif
> endfor
> endfor
> endfor
>
> close, 1 ;close the file
>
> here, f(a,b,c) is whatever combination of a b and c that's meant to
> reproduce the number goal to within the number error. the lows and

> highs are your lower and upper bondaries for a,b, and c, and the steps
> are how much to increment each time.
>
> Please let me stress that this is not only an inefficient algorithm
> (it wastes time checking hopeless candidates), but one for which IDL
> will run very slowly (IDL hates extensive looping). Posting it here
> actually makes me feel a little dirty. I hope David Fanning doesn't
> see it...
>
> chris

Holy Crap, you mean I have the right syntax!?!?!? :)

The data is always (at this point) going to have the form of [2,*]
*=30-18,000. It sounds form your last post Chris that I'm always going
to have trouble since the rows are always going to exceed the columns.
Just in case though, here is the info you requested along with the
code that's not working again:

```
ind_small = where(ar_plan[1,*] lt 2*x,count)
if count ge 0 then ar_plan_small=ar_plan[*,ind_small] else
ar_plan_small=0
ar_plan_small

IDL> help,ar_plan
AR_PLAN         FLOAT     = Array[2, 129]
IDL> help,count
COUNT           LONG      =           0
IDL> print,max(ind_small)
         -1
IDL> print,min(ind_small)
         -1
```

I put this at the end of the program, but I receive compilation errors
on the if, the end if and the final endfor statements. enl is a
function of otx and I tried (on the off chance) enl(o,t,x). I'm trying
to understand what the problem is, hopefully I'm not wasting too much
of your time :) Really though, I do appreciate the help.

```
openw,1,'g:\mars_tectonics\idl_programs\test.dat'; this opens a file
for writing
for o=50,80,5 do begin
   for t=10,100,5 do begin
     for x=5,14,1 do begin
          if (enl ge 0.06) && (enl le 0.06)
then begin
             printf,1,a,b,c,format='(3f9.3)' ; records variables to
```

three decimal places
          endif
     endfor
   endfor
endfor
close, 1 ;close the file

Subject: Re: Need help with an Iterative solution in IDL (relative newb question)
Posted by Chris[6] on Fri, 15 Aug 2008 07:54:05 GMT
View Forum Message <> Reply to Message

On Aug 14, 6:54 pm, mbwel...@gmail.com wrote:
> On Aug 14, 7:25 pm, Chris <beaum...@ifa.hawaii.edu> wrote:
>
>> On Aug 14, 2:45 pm, mbwel...@gmail.com wrote:
>
>>> On Aug 14, 2:20 pm, Chris <beaum...@ifa.hawaii.edu> wrote:
>
>>>> On Aug 14, 9:56 am, mbwel...@gmail.com wrote:
>
>>>> > On Aug 14, 11:50 am, Brian Larsen <balar...@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 Physicshttp://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
>>>> > identifer
>>>> > ar_plan_large = ar_plan[*,ind_large] ; place in matrice with
>>>> > identifer
>>>> > 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
>>>> > combintation 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 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
>
>>> I'm trying to fix the where statement returning -1, here is what I've
>>> tried thus far:
>>> ind_small = where(ar_plan[1,*] lt 2*x,count)   ; select faults such

>>> that L < 2x
>>> if count ge 0 then ar_plan_small=ar_plan[*,ind_small] else
>>> ar_plan_small=0
>>> ar_plan_small
>>> but I'm still getting the same error, I'm sure I have the syntax
>>> wrong. Unfortunately I'm not quite at the level to trouble shoot this
>>> myself, confidently.
>
>>> I have ordered the book suggested, I would imagine that it would come
>>> in handy very soon, but for the shear learning experience of it I
>>> would like to try it in IDL first (plus research waits for no amazon
>>> order). I can limit the increments for each variable to make it more
>>> manageable (less than 10^10 total steps), I just need some help and/or
>>> examples to illustrate how to create five nested loops for each
>>> variable, with each bounded condition and set increment that satisfy
>>> e_t that are recorded to an ASCII file. e.g. o = 50-80, del o = 5;
>>> t=5-100, del t = 10; etc...
>
>>> Thanks,
>>> ~Matt
>
>> The where problem probably comes from the fact that you are selecting
>> indices from the sub-array ar_plan[1,*] but indexing the array
>> ar_plan[*,indsmall]. In other words, you select ROWS of interest (IDL
>> is column major, so array[i,j] is the ith column, jth row) and then
>> index those COLUMNS. If there are more rows than columns, you may get
>> an 'array index out of bounds' error. If you are still having issues,
>> try including the output of the following lines in your next post:
>
>> help,ar_plan
>> help,count
>> print,max(ind_small)
>> print,min(ind_small)
>
>> Also remember that IDL is zero-indexed so, if you are trying to access
>> the first column of something, you would use ar_plan[0,*] and not
>> ar_plan[1,*]
>
>> A clunky nested for loop for three variables looks something like this
>
>> openw,1,'output.dat'; this opens a file for writing
>> for a=alow, ahigh, astep do begin
>> for b=blow, bhigh, bstep do begin
>> for c=clow, chigh, cstep do begin
>> if (f(a,b,c) ge goal-error) && (f(a,b,c) le goal+error)
>> then begin
>> printf,1,a,b,c,format='(3f9.3)' ; records variables to
>> three decimal places

>>   endif
>>   endfor
>>   endfor
>> endfor
>
>> close, 1 ;close the file
>
>> here, f(a,b,c) is whatever combination of a b and c that's meant to
>> reproduce the number goal to within the number error. the lows and
>> highs are your lower and upper bondaries for a,b, and c, and the steps
>> are how much to increment each time.
>
>> Please let me stress that this is not only an inefficient algorithm
>> (it wastes time checking hopeless candidates), but one for which IDL
>> will run very slowly (IDL hates extensive looping). Posting it here
>> actually makes me feel a little dirty. I hope David Fanning doesn't
>> see it...
>
>> chris
>
> Holy Crap, you mean I have the right syntax!?!?!? :)
>
> The data is always (at this point) going to have the form of [2,*]
> *=30-18,000. It sounds form your last post Chris that I'm always going
> to have trouble since the rows are always going to exceed the columns.
> Just in case though, here is the info you requested along with the
> code that's not working again:
>
> ind_small = where(ar_plan[1,*] lt 2*x,count)
> if count ge 0 then ar_plan_small=ar_plan[*,ind_small] else
> ar_plan_small=0
> ar_plan_small
>
> IDL> help,ar_plan
> AR_PLAN   FLOAT   = Array[2, 129]
> IDL> help,count
> COUNT   LONG   =   0
> IDL> print,max(ind_small)
>   -1
> IDL> print,min(ind_small)
>   -1
>
> I put this at the end of the program, but I receive compilation errors
> on the if, the end if and the final endfor statements. enl is a
> function of otx and I tried (on the off chance) enl(o,t,x). I'm trying
> to understand what the problem is, hopefully I'm not wasting too much
> of your time :) Really though, I do appreciate the help.
>

```
>  openw,1,'g:\mars_tectonics\idl_programs\test.dat'; this opens a file
>  for writing
>  for o=50,80,5 do begin
>     for t=10,100,5 do begin
>       for x=5,14,1 do begin
>            if (enl ge 0.06) && (enl le 0.06)
>  then begin
>               printf,1,a,b,c,format='(3f9.3)' ; records variables to
>  three decimal places
>            endif
>       endfor
>     endfor
>  endfor
>  close, 1 ;close the file
```

Yeah, you have some problems:)

First, you that count is zero, meaning that there are no values which match your search within WHERE. So you shouldn't even be trying to index anything with the output of where (-1). Your test is 'if count ge 0', means 'do this if the count is greater than OR EQUAL TO 0'. You want 'if count gt 0' (if count is strictly greater than 0). I would also think more carefully if your where code is doing what you think it will- this switching between array[1,*] and array[ind,*] sounds wrong.

The other errors may very well be occuring if the text is formatted in your file like it is on my screen. The comment 'this opens a file for writing' spills over to a new line right at the word 'for.' IDL doesn't see a semicolon, so it interprets FOR as the beginning of a for loop. this would explain the complilation error at the last for (it's looking for one more 'endfor'). If 'then begin' really is on a new line, it shouldn't be. if the 'records variables to three decimal places' spills onto a new line, it shouldn't.

unfortunately, I think you are battling a lot of syntax problems related to unfamiliarity with IDL. If that is the case, I think what you are trying to code is a bit ambitious - it will have algorithm implementation problems of its own. I would recommend using a program like Mathematica or Matlab if they are available to you- they have built in routines to do multi-dimensional minimum finding (like NMinimize, FindRoot, FindMinimum, etc in Mathematica). Plus, Mathematica doesn't compile and can be executed line-by-line, so you can 'interact' with that particular 'data language' more easily. If you are learning a language from scratch for this problem, Mathematica will be much faster.

chris

Subject: Re: Need help with an Iterative solution in IDL (relative newb question)
Posted by Jean H. on Fri, 15 Aug 2008 16:44:53 GMT
View Forum Message <> Reply to Message

> ind_small = where(ar_plan[1,*] lt 2*x,count)   ; select faults such
> if count ge 0 then ar_plan_small=ar_plan[*,ind_small] else

> but I'm still getting the same error, I'm sure I have the syntax
> wrong. Unfortunately I'm not quite at the level to trouble shoot this
> myself, confidently.

well, this is a very easy problem indeed, that every beginner can solve.
Read your code and think of what it does.

1) where(..., count). So, if you have 1 valid subscript, what should
the value of count be? What if you have NO valid subscript? Could
'count' be negative?

2) if count ge 0. So you deal with 0 or positive values. Again, what
does a count of 0 mean?

3) ar_plan[*,ind] What would it do if count = 0 (and therefore ind = -1)

Jean