Subject: Fitting plasma waveforms with 10^6 variable combos!
Posted by ted1508 on Wed, 07 Dec 2016 07:08:31 GMT
View Forum Message <> Reply to Message

Hello readers!

This is my first post here and a Hail Mary in a desperate attempt to speed up my IDL code...

I am presenting electric field antenna measurements from dust impacts onto the MAVEN spacecraft around Mars at the AGU conference next week. My code compares these waveforms (600 points) to a theoretical model, of which I have 5 variables, 4 with 10 possible points and 1 with 100 (the higher the resolution the variables the better!).

Problem is, 10^6 loops is a lot to go through. I tried doing it all in massive arrays, and its not much faster (IDL's memory allotment issue versus overhead with for loops)...

At this point I am thinking it would be faster to learn C and recode it from scratch then let this run for a week.

...

Part of the problem is that I am using this function (twice!) for each of the 600 time points:

```
function fun_ant,time,to,Q,deltat,C_ant,dtprime,tao_ant
  tprime=findgen(100)*dtprime
  one=  exp((-(time-tprime-to)^2)/(2*(deltat)^2))*exp(-(tprime)/tao_ ant)
  constant=(-Q/(C_ant*(sqrt(2*!pi)*(deltat))))
  equation=constant*Total(one)*dtprime
  return, equation
end
```

The variables which I will be looping through are time, deltat, tao_ant/bod, and Q_ant/bod (for plasma charge recollection to MAVEN's antenna and body respectively)

...

Right now the main procedure runs roughly as follows:

```
 for a = start_a,stop_a-1 do begin    ;;; loop through each waveform (there are 5000)

    input_max=max(mf_hits[a,*],mf_max_loc)    ;;; used to center waveforms later

    for ta = 1., 9. do begin                 ;;;; now all the combinations...
      tao_ant = 0.0001*double(ta)

      for tb = 1., 9. do begin
        tao_bod = 0.0001*double(tb)
        print,tb
        for dt = 1., 9. do begin
          deltat = 0.00001*double(dt)

          for qa = 0, 9. do begin
            q_ant = 0.000000000001*double(qa)

            for qb = 1, 9. do begin
              q_bod = 0.000000000001*double(qb)

              for n=0,n_elements(new_time)-1 do begin    ;;;; and the 600 time points.....
                newtime = new_time[n]   ;;; the function for this is defined above
                pot_ant[n] = fun_ant(newtime,to,q_ant,deltat,C_ant,dtprime,tao_ant)
                pot_bod[n] = fun_bod(newtime,to,q_bod,deltat,C_bod,dtprime,tao_bod)
              endfor

              pot = pot_ant - pot_bod    ;;; difference in potential between antenna and spacecraft

              Lag_mf = 25 ;;;; how much the 2 waveforms (mf_hits and pot) can shift

               similarity=double(max(c_correlate(mf_hits[a,mf_max_loc-103:m
f_max_loc+500],pot[0:603],lag_mf)))

              if similarity GT best_sim then begin
                ;;;; save the best variable values
              endif

            endfor    ; qb
          endfor    ; qa
        endfor    ; dt
      endfor    ; tb
    endfor    ; ta
  endfor   ; a

...
```

So theres my problem: I can choose between massive arrays or massive number of for loops, either way resulting in several days of computation.

For those thinking I could step through the values iteratively, there are local 'potential wells' where the similarity value will be higher than the neighbors but still will not be the best fit.

Does anyone have an idea if this code could be sped up by an order of magnitude?

Appreciate all your thoughts!
-Tenacious Ted

Subject: Re: Fitting plasma waveforms with 10^6 variable combos!
Posted by natha on Wed, 07 Dec 2016 13:24:09 GMT
View Forum Message <> Reply to Message

I suggest you to parallelize your code using the CPU_Process_Manager library
https://github.com/bernatp3rs/idl_cpu_pm

It will be fast and easy to adapt your code to the library. A couple of suggestions more:

- Change your multiplications to something like:
  for ta = 0.0001D, 0.0009D, 0.0001D do begin ;; I think it should work
- It seems that IDL is much faster when you use for ... do for ... do for .. instead of for ..... do begin for .... do begin
  check http://www.idlcoyote.com/tips/forloops2.html for more details
- delete the line "newtime = new_time[n]". it is not neeeded, pass newtime[n] to the functions fun_ant and fun_bod
- Put "Lag_mf = 25" outside of all loops
- Use the keyword /double in c_correlate instead of using the function double() afterwards
- Bring this line "mf_hits[a,mf_max_loc-103:mf_max_loc+500]" to the first loop, something like mf_hits_cur=mf_hits[a,mf_max_loc-103:mf_max_loc+500] and then use this temporary variable mf_hits_cur in the c_correlate function

I think that's all for now. I think that my last suggestion is the most important. Your code will be improved for sure!
Cheers,
nata

Subject: Re: Fitting plasma waveforms with 10^6 variable combos!
Posted by Dick Jackson on Wed, 07 Dec 2016 17:43:10 GMT
View Forum Message <> Reply to Message

Hi,

I would first be sure to check if any of the optimization/minimization routines in IDL might be sufficient for your search space…

http://www.harrisgeospatial.com/docs/mathematics_funt_list.h tml ("Optimization")
https://www.physics.wisc.edu/~craigm/idl/fitting.html ("TNMIN" is dependable for many applications)
http://www.ncnr.nist.gov/staff/dimeo/idl_programs.html (a couple of "global optimization" approaches I have not tried)

But if you really have to search the entire space, I think I've found 38% in time savings…

nata had some good tips, and I think there are some tweaks you can make to the function(s) being called in the inner loop (each called 196,830,000,000 times for your whole project, if I understand correctly) that will make some difference. (question: is there really a "fun_bod" as well as the "fun_ant"?):

Original:

```
function fun_ant,time,to,Q,deltat,C_ant,dtprime,tao_ant
   tprime=findgen(100)*dtprime
   one=  exp((-(time-tprime-to)^2)/(2*(deltat)^2))*exp(-(tprime)/tao_ ant)
   constant=(-Q/(C_ant*(sqrt(2*!pi)*(deltat))))
   equation=constant*Total(one)*dtprime
   return, equation
end

function fun_ant2,time,to,Q,deltat,C_ant,dtprime,tao_ant
   tprime=findgen(100)*dtprime

; Recording test run time for several tests
; (note: time testing is not an exact science, and your mileage may vary)

; Original:
;   one=  exp((-(time-tprime-to)^2)/(2*(deltat)^2))*exp(-(tprime)/tao_ ant);0.124

; Algebra helps: exp(x)*exp(y) = exp(x+y)
;   one= exp( (-(time-tprime-to)^2)/(2*(deltat)^2) - (tprime)/tao_ant)   ;0.092

; Factoring and reordering to put scalar operations first:
;   one= exp( -0.5D / deltat^2 * (time-tprime-to)^2 - (tprime)/tao_ant)  ;0.079

; Sometimes using x*x instead of x^2 helps. This didn't give a big improvement,
; but it didn't hurt
   ttt = time-tprime-to
   one= exp( ((-0.5D / (deltat*deltat)) * (ttt*ttt)) - (tprime/tao_ant));0.077

   ; Constant is Sqrt(2*!DPi)
   constant=(-Q/(C_ant*(2.5066282746310002D * deltat)))
   equation=constant*Total(one)*dtprime
   return, equation
end

pro test_fun_ant

newtime = 0.0001D
to = 0.0001D
dtprime = 0.0000001D
C_ant = 0.01D

tao_ant = 0.0001D
tao_bod = 0.0001D
deltat = 0.00001D
```

```
q_ant = 0.000000000001D
q_bod = 0.000000000001D

nRuns = 20L
nCalls = 10000L

best_fun_ant = 1D9
FOR j=1, nRuns DO BEGIN
  TIC
  FOR i=1,nCalls DO x1=fun_ant(newtime,to,q_ant,deltat,C_ant,dtprime,tao_ant)
  t = TOC()
  IF t LT best_fun_ant THEN best_fun_ant = t
ENDFOR ; j over nRuns

best_fun_ant2 = 1D9
FOR j=1, nRuns DO BEGIN
  TIC
  FOR i=1,nCalls DO x2=fun_ant2(newtime,to,q_ant,deltat,C_ant,dtprime,tao_ant)
  t = TOC()
  IF t LT best_fun_ant2 THEN best_fun_ant2 = t
ENDFOR ; j over nRuns

Help, best_fun_ant, best_fun_ant2, x1, x2, x2-x1

END
```

I hope this helps… and of course, while the exhaustive search is running, you can experiment with the optimizers! :-)

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada --- http://www.d-jackson.com