Subject: for loop is killing me
Posted by tarequeaziz on Tue, 06 Nov 2007 06:31:24 GMT
View Forum Message <> Reply to Message

Hello all,

help me! I am running a code 'infested' with for loops.And as u can
guess, its painfully slow. The part of my code with loop looks like
the following:

jump1:

;Setting up the Green functions array which will have the same number
of elements as of amp_vec

```
G=fltarr(n_elements(amp_vec),n_elements(amp_vec))
G_phi=fltarr(n_elements(amp_vec))
G_rp=fltarr(n_elements(amp_vec))
G_r=fltarr(n_elements(amp_vec))
G_in=fltarr(n_elements(amp_vec))
G_out=fltarr(n_elements(amp_vec))
G_p=fltarr(n_elements(amp_vec))
G_h=fltarr(n_elements(amp_vec))
gp=fltarr(n_elements(amp_vec))
gh=fltarr(n_elements(amp_vec))
G_in_msum = fltarr(n_elements(amp_vec))
G_out_msum = fltarr(n_elements(amp_vec))
G_in_phisum = fltarr(n_elements(amp_vec))
G_out_phisum = fltarr(n_elements(amp_vec))



if ignore_realdata eq 1. then begin
  amp_vec= abs(randomn(0.5,n_radial_points,/double))
  a=1.
 ;print,' amp_vec is',amp_vec
endif

do_u_like_to_start_with_rp_loop = 0 ; Set 1 if YES,0 if NO

if do_u_like_to_start_with_rp_loop eq 1. then BEGIN
  PRINT,' WE ARE USING rp PREFERED LOOP'
  goto,jump2

ENDIF
```

```
PRINT,' WE ARE USING r PREFERED LOOP'
  for i_r= 0,n_radial_points-1 do begin

   loop_time = systime(1)

   if ignore_realdata eq 1. then begin

    r_scld=( i_r*2.)/(n_radial_points -1)    ;---------> We took off
trap radius from here for scaling purpose,see note

       ;---------> Also note the use of factor 2. This is because, we
want to get values outside
                                                ;----------
>  the cylinder. But this is just for demonstration purpose!!!!
   endif else begin

    r_scld=( i_r)/(n_radial_points -1)  ; took off the factor 2

   endelse
   ;r_scld = 2.

   ;print,'r_scld value is ',r_scld

    for i_rp= 0,n_radial_points-1 do begin


      if ignore_realdata eq 1. then begin

       rp_mat = randomn(1,n_radial_points,/double)
          ;print,'rp_mat is',rp_mat

             rp_scld = rp_mat(i_rp)

      endif else begin

       ;rp_scld=( i_rp/((n_radial_points-1))) * float(amp_vec(i_rp))
       rp_scld= float(amp_vec(i_rp))
       ;print,' amp_vec is',amp_vec

      endelse




    ; rp_scld=( i_rp/((n_radial_points-1))) * float(amp_vec(i_rp))

     ;rp_mat = randomn(1,n_radial_points,/double)
    ; print,'rp_mat is',rp_mat
```

```
    ;rp_scld = rp_mat(i_rp)

    ;rp_scld = 0.5


    ;;print,'rp_scld for i_rp= ',i_rp,' is,  ',rp_scld

     if (r_scld gt rp_scld) then begin
      r_plus = r_scld
      r_minus = rp_scld
     endif else begin
      r_minus = r_scld
      r_plus = rp_scld
     endelse

    ;print,'r_plus value is ',r_plus
    ;print,'r_minus value is ',r_minus

     for i_m=1,20 do begin       ;n_radial_points do begin

     ;;print,'starting i_m value is ',i_m

     ;if  (n_radial_points le 5) then i_phi_max = 20 else i_phi_max =
n_radial_points
     ;print,'i phi  max is =  ',i_phi_max
     i_phi_max = 20.

       for i_phi=0,i_phi_max - 1 do begin
        ; print,'starting i_phi value is',i_phi

         count= i_phi + 1.
           ;print,'count is ',count
         phi=(count*2*!PI)/(i_phi_max)

        ; print,'phi value is= ',phi
        ; phi= 0.785398163  ; <----- In radian

        a= 1.  ; -----> Scaled trap radius

        gp(i_phi) = 2.0*(1./i_m)*(r_minus/r_plus)^i_m  *
cos(i_m*(phi))
        gh(i_phi) = 2.0*(1./i_m)*(r_minus * r_plus/a^2)^i_m  *
cos(i_m*(phi))

        ;print,'cos(i_m*phi) is',cos(i_m*phi)

       ;if  (i_r eq 0.)  &&  (i_rp eq 0.) && (i_m eq 1.)   then begin
```

```
         ;print,' the 1st gp is ',gp(i_phi)
         ;print,' the 1st gh is ',gh(i_phi)
        ;endif

          ;if  (i_r eq n_radial_points  - 1.)  &&  (i_rp eq
n_radial_points -1.) && (i_m eq n_radial_points -1.)   then begin

         ;print,' the Last gp is ',gp(i_phi)
         ;print,' the Last gh is ',gh(i_phi)

        ;endif

        endfor ; end of phi loop


       G_in_phisum [i_m -1] =  total(gp)  ; Add up the phi elements
for a specific m
       G_out_phisum [i_m -1] = total(gh)



      endfor  ; end of m-loop


     G_in_msum(i_rp) =  total(G_in_phisum)  ; Add up all m values for
a specific rp value
     G_out_msum(i_rp) = total(G_out_phisum)

     G_in(i_rp) = -alog((r_plus)^2.) + G_in_msum[i_rp]
     G_out(i_rp) = -alog((a^2./rp_scld)^2.) + G_out_msum[i_rp]


     ;print,'G_in is',G_in
     ;print,'G_out is',G_out

     G(i_r,i_rp)= G_in(i_rp) - G_out(i_rp) - alog((a/rp_scld)^2.)

      index=where(finite(G,/NaN) ,count)
      ;print,' index is ',index
        if (count ne 0) then G[index] = 0.
;      non_zero = where(G,count)
;      print,'non zero value is',non_zero
;
;      if count ne 0. then G = G[non_zero]


     ;print,' last log part',alog((a/rp_scld)^2.)
```

```
     ;print,' G_in - G_out is  ',G_in - G_out
       ;print,'G inside rp loop is  ',G(i_r,i_rp),' for i_r =',i_r,'
and i_rp =',i_rp


    endfor  ;end of rp loop


   ;G_p= G_in
   ;G_h= G_out
   ;print,'G_p is',G_p,'and G_h is',G_h
    ;G(i_r,i_rp)= G_p(i_rp) - G_h(i_rp) - alog((a/rp_scld)^2.)

   ;print,'rp_scld is ',rp_scld

  ; print,' G inside the r loop is',G

 ; if i_r eq (n_radial_points-1) then print,'Last r_scld value is=',
r_scld

 ;if i_r eq  ( (n_radial_points-1)/10.)  then print,'First 1/10th
r_scld value is=', r_scld
 ;if i_r eq  ( (n_radial_points-1)/2.)  then print,'First 1/2th
r_scld value is=', r_scld
 ;if i_r eq  ( (n_radial_points-1)*3./4.)  then print,'3/4 th r_scld
value is=', r_scld
 ;if i_r eq (n_radial_points-1) then print,'Last r_scld value is=',
r_scld

 if i_r eq (n_radial_points - 1.) then begin

 print,'The time it took to finish  ',i_r,'th loop is', systime(1) -
loop_time,'seconds'

 endif

  endfor ; end of r loop


  ;print,'here is the PROFILER report for r prefered loop'

        ;print,'G is  ',G       ;[n_radial_points - 1]


if ignore_realdata eq 1 then begin

     r_vec=(findgen(n_radial_points)/(n_radial_points))
    potential=fltarr(n_radial_points)
```

```
  prod = fltarr(n_radial_points)

endif else begin

  r_vec=(findgen(n_theta_points)/(n_theta_points))
     potential=fltarr(n_theta_points)
  prod = fltarr(n_theta_points)
endelse

if ignore_realdata eq 1 then p= n_radial_points else p= n_theta_points



  for k = 0,100 do begin ;p -1 do begin
    ;rp_mat = randomn(1,n_radial_points,/double)
    ;print,' amp_vec is',amp_vec
    ;print,'amp_vec is',amp_vec[k]
    ;print,'G[*,k] is',G[*,k]
     prod =amp_vec[k] * G(*,k)
     prod_invfft = abs( fft(prod,/inverse))
     potential[k] = int_tabulated(r_vec,prod_invfft)
    ;print,'potential is',potential
```

=====

So as u can see there are 4 for loops running. the m loop with running index i_m should be 200. But setting it to 200 slows down the program.

My supervisor said that, this should be a very fast way of calculating green function than the traditional way(which my group-mate is doing).

what I am doing wrong?
Any help will be HIGHLY appreciated!

Best,

Tareque

Subject: Re: for loop is killing me
Posted by Conor on Thu, 08 Nov 2007 15:26:27 GMT
View Forum Message <> Reply to Message

Okay, here's one more for you. Your m loop and phi loop can both be made for loop-less. Here's your warning though: I haven't tested this

code in the slightest. It's hard to test stuff when it is part of a much larger program, so I'll leave that to you. I'll just explain what this chunk of code is doing. So this replaces your m and your phi loops:

```
i_phi_max = 20.
phis = rebin(findgen(i_phi_max)+1,i_phi_max,i_phi_max)
i_ms = rebin(findgen(1,i_phi_max)+1,i_phi_max,i_phi_max)

gp = 2.0*(1./i_ms)*(r_minus/r_plus)^i_ms * cos(i_ms * phis*2*!PI/
i_phi_max )
gh = 2.0*(1./i_ms)*(r_minus*r_plus/a^2)^i_ms * cos(i_ms * phis*2*!PI/
i_phi_max )

G_in_phisum = total(gp,1)
G_out_phisum = total(gh,1)
```

The general idea here is to simply do all the calculations at once, combined into one big array operation. So for instance from looking at the phi loop you know that your phi's are going to go from 1-20 and are going to be used 20 times by the m-loop. Imagine this as a 20x20 array where each row contains the numbers 1-20. The second line (phis = rebin()) creates such a 20x20 array. Rebin is used to repeat the findgen() 20 times vertically.

Next consider your i_m's. These will also have the values 1-20 and will be used against the different values in the phis array. So you can consider it to be a 20x20 array where each column has the values 1-20. Once again, this array is created using rebin and findgen in a very similar way to how I created the phi array. Once this is accomplished you have two 20x20 arrays with all the 400 possible combinations of phi and m. Then, you perform your calculations exactly as you did in the for loop and in one go calculate all the possible results. Finally you use the dimension keyword to the total function to sum up along rows. Viola, no more for loops! To make sure things are clear, I want to explain this in a different way:

The idea is to do all your calculations at once. To make this happen I created a 20x20 array for the phi loop and a 20x20 array for the m loop. The idea is to pre-calculate all possible combinations of m and phi and calculate all the possible values at once. That's what the phis and ims arrays are for. Between them they contain all possible values for i_m and phi. Traveling along a row is the equivelent of looping over phi, and traveling along a column is the equivilent of looping over i_m. So for instance if, with your original set up, you were in the 3rd iteration of the m-loop and the 12th iteration of the phi loop the i_m variable would have the value of 3 and the phi

variable would have the value of 12. With the new setup if you printed out the values of i_ms[11,2] and phis[11,2] you would get the values 3 and 12! Hopefully this all makes sense.

Also, please note that there is a potential source of confusion between my code and yours because for your phi-loop you looped from phi=0,19 but then added one later. In my code this meant that the findgen() for phi had a +1 after it.

Subject: Re: for loop is killing me
Posted by Conor on Thu, 08 Nov 2007 19:26:37 GMT
View Forum Message <> Reply to Message

Okay, here's one more for you. Your m loop and phi loop can both be made for loop-less. Here's your warning though: I haven't tested this code in the slightest. It's hard to test stuff when it is part of a much larger program, so I'll leave that to you. I'll just explain what this chunk of code is doing. So this replaces your m and your phi loops:

```
i_phi_max = 20.
phis = rebin(findgen(i_phi_max)+1,i_phi_max,i_phi_max)
i_ms = rebin(findgen(1,i_phi_max)+1,i_phi_max,i_phi_max)

gp = 2.0*(1./i_ms)*(r_minus/r_plus)^i_ms * cos(i_ms * phis*2*!PI/
i_phi_max )
gh = 2.0*(1./i_ms)*(r_minus*r_plus/a^2)^i_ms * cos(i_ms * phis*2*!PI/
i_phi_max )

G_in_phisum = total(gp,1)
G_out_phisum = total(gh,1)
```

The general idea here is to simply do all the calculations at once, combined into one big array operation. So for instance from looking at the phi loop you know that your phi's are going to go from 1-20 and are going to be used 20 times by the m-loop. Imagine this as a 20x20 array where each row contains the numbers 1-20. The second line (phis = rebin()) creates such a 20x20 array. Rebin is used to repeat the findgen() 20 times vertically.

Next consider your i_m's. These will also have the values 1-20 and will be used against the different values in the phis array. So you can consider it to be a 20x20 array where each column has the values 1-20. Once again, this array is created using rebin and findgen in a very similar way to how I created the phi array. Once this is accomplished you have two 20x20 arrays with all the 400 possible

combinations of phi and m. Then, you perform your calculations exactly as you did in the for loop and in one go calculate all the possible results. Finally you use the dimension keyword to the total function to sum up along rows. Viola, no more for loops! To make sure things are clear, I want to explain this in a different way:

The idea is to do all your calculations at once. To make this happen I created a 20x20 array for the phi loop and a 20x20 array for the m loop. The idea is to pre-calculate all possible combinations of m and phi and calculate all the possible values at once. That's what the phis and ims arrays are for. Between them they contain all possible values for i_m and phi. Traveling along a row is the equivelent of looping over phi, and traveling along a column is the equivilent of looping over i_m. So for instance if, with your original set up, you were in the 3rd iteration of the m-loop and the 12th iteration of the phi loop the i_m variable would have the value of 3 and the phi variable would have the value of 12. With the new setup if you printed out the values of i_ms[11,2] and phis[11,2] you would get the values 3 and 12! Hopefully this all makes sense.

Also, please note that there is a potential source of confusion between my code and yours because for your phi-loop you looped from phi=0,19 but then added one later. In my code this meant that the findgen() for phi had a +1 after it.