Subject: Re: Removing if then else loop for efficiency
Posted by penteado on Sun, 10 Jan 2010 14:42:43 GMT
View Forum Message <> Reply to Message

On Jan 10, 10:06 am, Tom Ashbee <tlash...@googlemail.com> wrote:
>
> It's way too slow at the moment, but I gather I can remove one of the
> for loops and the if then else loop but I have no idea how!

It is impossible to tell without knowing what happens in the lines you omitted.

if..then..else is not a loop, it is a conditional construct. Loops are control structures that (potentially) make some piece of code get executed multiple times, such as for, while and repeat.

Subject: Re: Removing if then else loop for efficiency
Posted by Tom Ashbee on Sun, 10 Jan 2010 14:46:44 GMT
View Forum Message <> Reply to Message

On Jan 10, 12:06 pm, Tom Ashbee <tlash...@googlemail.com> wrote:
> Hello,
>
> apologies for my newb-ness, but I was hoping I could get some help
> with increasing the efficiency of the following program:
>
> At the moment it is like this
>
> for i=0, N-1 do begin
> for j=0, N-1 do begin
> if i ne j then begin
> ;stuff
> endif else begin
> ;more stuff
> endif
> endfor
> endfor
>
> It's way too slow at the moment, but I gather I can remove one of the
> for loops and the if then else loop but I have no idea how!
>
> Any help would be much appreciated and I can post the full program too
> if it helps. FTR it's the RHS of a Hamiltonian.
>
> Thanks again

OK here's the whole program:

```
function velocities, t, xyvec

N = 50
R = 5.0

xvec = xyvec(0: N-1)
yvec = xyvec(N: (N*2)-1)

dxvecdt = dblarr(N)
dyvecdt = dblarr(N)

for i = 0, N-1 do begin

   dxvecdt(i) = 0.0d
   dyvecdt(i) = 0.0d

   for j = 0, N-1 do begin

      gam = gamma(N, 2.0d, 10.0d)

      if i ne j then begin

      dxvecdt(i) = dxvecdt(i) + ( gam(j) /(2.0d*!pi))*(yvec(j)-yvec
(i))/((xvec(i)-xvec(j))^2+(yvec(i)-yvec(j))^2)  -   ( gam(j) /(2.0d*!
 pi))*((yvec(j)*R^2)/((xvec(j))^2+(yvec(j))^2)-yvec(i))/((xve c(i)-(xvec
 (j)*R^2)/((xvec(j))^2+(yvec(j))^2))^2+(yvec(i)-(yvec(j)*R^2) /((xvec(j))
^2+(yvec(j))^2))^2)

      dyvecdt(i) = dyvecdt(i) + ( gam(j) /(2.0d*!pi))*(xvec(i)-xvec
(j))/((xvec(i)-xvec(j))^2+(yvec(i)-yvec(j))^2)   -  ( gam(j) /(2.0d*!
 pi))*(xvec(i)-(xvec(j)*R^2)/((xvec(j))^2+(yvec(j))^2))/((xve c(i)-(xvec
 (j)*R^2)/((xvec(j))^2+(yvec(j))^2))^2+(yvec(i)-(yvec(j)*R^2) /((xvec(j))
^2+(yvec(j))^2))^2)

       endif else begin

       dxvecdt(i) =  dxvecdt(i) - ( gam(j) /(2.0d*!pi))*((yvec(j)*R^2)/
 ((xvec(j))^2+(yvec(j))^2)-yvec(i))/((xvec(i)-(xvec(j)*R^2)/( (xvec(j))
 ^2+(yvec(j))^2))^2+(yvec(i)-(yvec(j)*R^2)/((xvec(j))^2+(yvec (j))^2))
^2)

       dyvecdt(i) = dyvecdt(i) - ( gam(j) /(2.0d*!pi))*(xvec(i)-(xvec
 (j)*R^2)/((xvec(j))^2+(yvec(j))^2))/((xvec(i)-(xvec(j)*R^2)/ ((xvec(j))
 ^2+(yvec(j))^2))^2+(yvec(i)-(yvec(j)*R^2)/((xvec(j))^2+(yvec (j))^2))
^2)

       endelse
```

```
    endfor

endfor

z = [dxvecdt, dyvecdt]

return, z

end
```

Subject: Re: Removing if then else loop for efficiency
Posted by penteado on Sun, 10 Jan 2010 18:25:16 GMT
View Forum Message <> Reply to Message

I think this does the same as your code, but it does not use any loops, and it should be much faster and easier to read:

```
function velocities,t,xyvec
compile_opt idl2
;Constants
N=50
R=5d0
;Unpack the input
xvec=xyvec[0:N-1]
yvec=xyvec[N:(N*2)-1]
;Temporary arrays for x[j], x[i], y[j], y[i]
xj=rebin(xvec,N,N)
xi=transpose(xj)
yj=rebin(yvec,N,N)
yi=transpose(yj)
;Repeated terms in the expressions
tmp1=(xi-xj)^2+(yi-yj)^2
tmp2=R^2/(xj^2+yj^2)
tmp3=(xi-xj*tmp2)^2+(yi-yj*tmp2)^2
;Terms of dxdt,dydt present everywhere
dxdt=-(yi-yj*tmp2)/tmp3
dydt=(xi-xj*tmp2)/tmp3
;Terms present only out of the diagonal
tmp4=1d0-identity(N) ;this is 0 in the diagonal, 1 out of it
dxdt+=((yj-yi)/tmp1)*tmp4
dydt-=((xi-xj)/tmp1)*tmp4
;Put the gamma factor
gamm=rebin(gamma(N,2.0d,10.0d)/(2d0*!dpi),N,N) ;this does not seem to
be IDL's gamma function
dxdt*=gamm
dydt*=gamm
```

```
;Sum over the rows
dxvecdt=total(dxdt,1)
dyvecdt=total(dydt,1)
;Pack the results
z=[dxvecdt,dyvecdt]
return,z
end
```

You should check that I did not misidentify anything, which would not have been difficult in such convoluted expressions.

Other points to note:

1) Do not use () for array indexes. Use [] instead. That makes it unambiguous that it is an array index, and not a function call.

2) When using doubles, as you did, use !dpi instead of !pi.

3) Your function has an argument t that is not used anywhere in it. I left it there, so that the argument order does not change.

Subject: Re: Removing if then else loop for efficiency
Posted by Tom Ashbee on Tue, 12 Jan 2010 13:09:13 GMT
View Forum Message <> Reply to Message

On Jan 10, 6:25 pm, pp <pp.pente...@gmail.com> wrote:

```
>  I think this does the same as your code, but it does not use any
>  loops, and it should be much faster and easier to read:
>
>  function velocities,t,xyvec
>  compile_opt idl2
>  ;Constants
>  N=50
>  R=5d0
>  ;Unpack the input
>  xvec=xyvec[0:N-1]
>  yvec=xyvec[N:(N*2)-1]
>  ;Temporary arrays for x[j], x[i], y[j], y[i]
>  xj=rebin(xvec,N,N)
>  xi=transpose(xj)
>  yj=rebin(yvec,N,N)
>  yi=transpose(yj)
>  ;Repeated terms in the expressions
>  tmp1=(xi-xj)^2+(yi-yj)^2
>  tmp2=R^2/(xj^2+yj^2)
>  tmp3=(xi-xj*tmp2)^2+(yi-yj*tmp2)^2
>  ;Terms of dxdt,dydt present everywhere
```

> dxdt=-(yi-yj*tmp2)/tmp3
> dydt=(xi-xj*tmp2)/tmp3
> ;Terms present only out of the diagonal
> tmp4=1d0-identity(N) ;this is 0 in the diagonal, 1 out of it
> dxdt+=((yj-yi)/tmp1)*tmp4
> dydt-=((xi-xj)/tmp1)*tmp4
> ;Put the gamma factor
> gamm=rebin(gamma(N,2.0d,10.0d)/(2d0*!dpi),N,N) ;this does not seem to
> be IDL's gamma function
> dxdt*=gamm
> dydt*=gamm
> ;Sum over the rows
> dxvecdt=total(dxdt,1)
> dyvecdt=total(dydt,1)
> ;Pack the results
> z=[dxvecdt,dyvecdt]
> return,z
> end
>
> You should check that I did not misidentify anything, which would not
> have been difficult in such convoluted expressions.
>
> Other points to note:
>
> 1) Do not use () for array indexes. Use [] instead. That makes it
> unambiguous that it is an array index, and not a function call.
>
> 2) When using doubles, as you did, use !dpi instead of !pi.
>
> 3) Your function has an argument t that is not used anywhere in it. I
> left it there, so that the argument order does not change.

Hi,

thanks a lot for this; it was very insightful and helpful.
Unfortunately it's just giving NaNs for z at the moment but I'm
working on debugging it.

Thanks again

Subject: Re: Removing if then else loop for efficiency
Posted by penteado on Tue, 12 Jan 2010 16:04:18 GMT
View Forum Message <> Reply to Message

On Jan 12, 11:09 am, Tom Ashbee <tlash...@googlemail.com> wrote:
> On Jan 10, 6:25 pm, pp <pp.pente...@gmail.com> wrote:
>

>
>
>> I think this does the same as your code, but it does not use any
>> loops, and it should be much faster and easier to read:
>
>> function velocities,t,xyvec
>> compile_opt idl2
>> ;Constants
>> N=50
>> R=5d0
>> ;Unpack the input
>> xvec=xyvec[0:N-1]
>> yvec=xyvec[N:(N*2)-1]
>> ;Temporary arrays for x[j], x[i], y[j], y[i]
>> xj=rebin(xvec,N,N)
>> xi=transpose(xj)
>> yj=rebin(yvec,N,N)
>> yi=transpose(yj)
>> ;Repeated terms in the expressions
>> tmp1=(xi-xj)^2+(yi-yj)^2
>> tmp2=R^2/(xj^2+yj^2)
>> tmp3=(xi-xj*tmp2)^2+(yi-yj*tmp2)^2
>> ;Terms of dxdt,dydt present everywhere
>> dxdt=-(yi-yj*tmp2)/tmp3
>> dydt=(xi-xj*tmp2)/tmp3
>> ;Terms present only out of the diagonal
>> tmp4=1d0-identity(N) ;this is 0 in the diagonal, 1 out of it
>> dxdt+=((yj-yi)/tmp1)*tmp4
>> dydt-=((xi-xj)/tmp1)*tmp4
>> ;Put the gamma factor
>> gamm=rebin(gamma(N,2.0d,10.0d)/(2d0*!dpi),N,N) ;this does not seem to
>> be IDL's gamma function
>> dxdt*=gamm
>> dydt*=gamm
>> ;Sum over the rows
>> dxvecdt=total(dxdt,1)
>> dyvecdt=total(dydt,1)
>> ;Pack the results
>> z=[dxvecdt,dyvecdt]
>> return,z
>> end
>
>> You should check that I did not misidentify anything, which would not
>> have been difficult in such convoluted expressions.
>
>> Other points to note:
>
>> 1) Do not use () for array indexes. Use [] instead. That makes it

>> unambiguous that it is an array index, and not a function call.
>
>> 2) When using doubles, as you did, use !dpi instead of !pi.
>
>> 3) Your function has an argument t that is not used anywhere in it. I
>> left it there, so that the argument order does not change.
>
> Hi,
>
> thanks a lot for this; it was very insightful and helpful.
> Unfortunately it's just giving NaNs for z at the moment but I'm
> working on debugging it.

Now that you mention it, I see a reason for the NaNs. The lines

```
tmp4=1d0-identity(N) ;this is 0 in the diagonal, 1 out of it
dxdt+=((yj-yi)/tmp1)*tmp4
dydt-=((xi-xj)/tmp1)*tmp4
```

were intended to add to dxdt only in the off-diagonal elements, by multiplying the diagonal elements of ((yj-yi)/tmp1) by 0 (same for dydt). But these diagonal elements are some non finite value (some form of NaN or Infinity), so their product with 0 is not 0, it is some NaN.

One way to get around this is to replace those 7 lines with:

```
;Terms of dxdt,dydt present only out of the diagonal
dxdt=((yj-yi)/tmp1)
dydt=-((xi-xj)/tmp1)
;Reset to 0 the diagonal elements
dxdt[0:N*N-1:N+1]=0d0
dydt[0:N*N-1:N+1]=0d0
;Terms of dxdt,dydt present everywhere
dxdt-=(yi-yj*tmp2)/tmp3
dydt+=(xi-xj*tmp2)/tmp3
```