Subject: Re: Interpolate whole array instead of looping through elements
Posted by Markus Schmassmann on Thu, 13 Apr 2017 12:21:48 GMT
View Forum Message <> Reply to Message

On 04/13/2017 01:50 AM, liam.steele@gmx.co.uk wrote:
> thanks for the replies everyone,
>
> Well it is a little bit more complicated than I originally said. I
> made the original question simpler to avoid confusing things!
>
> Basically, temp is of size [280,280,60,720]. What I am actually
> doing is getting the average temperatures in a radially symmetric crater. Take
> the following image as an example:
> https://s22.postimg.org/6r50fbrld/bilinear.jpg
>
> Imagine the black grid is the lat-lon temperature field at a certain
> level and time (i.e. temp[*,*,0,0]). Once I have this 2D field, I need
> to calculate how the average temperature varies from the centre of the
> crater to the edge. So I define a line (shown in red, with the black
> dots the locations I want values at), and for each black dot I use the
> bilinear function to get a value. I then rotate the red line a bit more
> and do the calculation again, and repeat. On each line there are about
> 100 points.
>
> Once a full circle of rotations is complete, the average temps from
> the centre to the edge of the crater are found. But only for one time
> and one level. At the moment I'm rotating the line by 5 degrees. So each
> time and each level of data has 36 rotations with each rotation having
> 100 points on the line to use the bilinear function on. So, it's
> something like:
>
```
> for iangle = 0, 35 do begin
>    for ipoint = 0, 99 do begin
>
>      ; Find ival and jval of the point we want to interpolate to
>      ival = ....
>      jval = ....
>
>      for itime = 0, 719 do begin
>        for ilev = 0, 59 do begin
>          out_vals[ipoint,ilev,itime] = out_vals[ipoint,ilev,itime] + $
>              bilinear(temp[*,*,ilev,itime],ival,jval)/36
>        endfor
>      endfor
>
>    endfor
> endfor
>
```

> And it goes really rather slowly. Looping through just iangle,
> ipointand itime takes 131 seconds (using the TIC,TOC functions). This then
> needs multiplied by 60 to loop through each atmospheric level, so it
> takes more than two hours in total. And this is just for one lot of
> data. At the moment I have 50 or so of these to calculate, so that's
> almost 5 days of IDL calculation!
>
> I was thinking there was maybe something that could be done where
> theiangle and ipoint loops still occur (as they have to, in order to find
> the i and j indices for the bilinear interpolation), but then
> interpolation could occur for all itime and ilev values at once in some
> speedy IDL vectorized way (since they are using the same indices). But
> maybe not! Maybe I need to find something other than IDL that might be
> quicker. Or just accept it is going to take a while to calculate!
>
> Apologies if none of this makes sense!

```
ivals=139.5+1.395*rebin(findgen(1,100),[36,100],/sample) $
      *rebin(sin(!pi/18*findgen(36,1)),[36,100],/sample)
jvals=139.5+1.395*rebin(findgen(1,100),[36,100],/sample) $
      *rebin(cos(!pi/18*findgen(36,1)),[36,100],/sample)
```

one simple thing for pure increased speed:

```
for iangle = 0, 35 do for ipoint = 0, 99 do for itime = 0, 719 do $
    for ilev = 0, 59 do out_vals[ipoint,ilev,itime] = $
       out_vals[ipoint,ilev,itime] + bilinear(temp[*,*,ilev,itime], $
       ivals[iangle,ipoint],jval[iangle,ipoint])/36
```

but better is to vectorize:

```
ivals2=rebin(ivals,[36,100,720],/sample)
ivals2=rebin(jvals,[36,100,720],/sample)
tvals2=rebin(findgen(1,1,720),[36,100,720],/sample)
out_vals=fltarr(100,720,60)
for ilev=0,59 do out_vals[0,0,ilev]=mean(interpolate( $
    reform(temp[*,*,ilev,*]), ivals2, jvals2, tvals2),dim=1 )
out_vals=transpose(out_vals,[0,2,1])
```

and if that is not fast enough, do the interpolation manually:

```
 wFF=rebin(floor(ivals)+280l*floor(jvals),[36,100,60,720],/sa mple)+ $
    rebin(280l^2*lindgen(1,1,60,720),[36,100,60,720],/sample)
wFC=rebin(floor(ivals)+280l* ceil(jvals),[36,100,60,720],/sample)+ $
    rebin(280l^2*lindgen(1,1,60,720),[36,100,60,720],/sample)
wCF=rebin( ceil(ivals)+280l*floor(jvals),[36,100,60,720],/sample)+ $
    rebin(280l^2*lindgen(1,1,60,720),[36,100,60,720],/sample)
wCC=rebin( ceil(ivals)+280l* ceil(jvals),[36,100,60,720],/sample)+ $
```

```
    rebin(280l^2*lindgen(1,1,60,720),[36,100,60,720],/sample)
weightFF=rebin( (1+(ivals mod 1))*(1+(jvals mod 1)), $
    [36,100,60,720],/sample)
weightFC=rebin(-(1+(ivals mod 1))*   (jvals mod 1) , $
    [36,100,60,720],/sample)
weightCF=rebin(-   (ivals mod 1) *(1+(jvals mod 1)), $
    [36,100,60,720],/sample)
weightCC=rebin(    (ivals mod 1) *   (jvals mod 1) , $
    [36,100,60,720],/sample)
 out_vals=mean(weightFF*temp[wFF]+weightFC*temp[wFC]+weightCF
*temp[wCF]+weightCC*temp[wCC],dim=1)
```

of which only the last line has to be repeated for every data set.

I haven't debugged anything, so some corrections might be necessary.

Good luck, Markus