Subject: Re: Expensive loops... can they be avoided?
Posted by Allan Whiteford on Tue, 26 Feb 2008 14:15:21 GMT
View Forum Message <> Reply to Message

Rainer,

Not using even a remotely similar algorithm (and one which might be
inadequate for your purposes) but how about:

```
xp=transpose(rebin(r,n_elements(r),n_elements(chi))) *     $
            rebin(cos(chi),n_elements(chi),n_elements(r))
yp=transpose(rebin(r,n_elements(r),n_elements(chi))) *
            rebin(sin(chi),n_elements(chi),n_elements(r)) $
triangulate,xp,yp,tr
b=trigrid(xp,yp,transpose(a),tr,xout=x,yout=y)
```

this should be faster than your previous solution but, as I said, might
not be suitable. You can also save xp, yp and tr between calls if your
"a" data is changing but everything else is staying the same which will
also speed things up a bit.

Also, this solution might not work. I've only really checked it for
syntax errors not for accuracy or it even being the correct thing to do.
I've only ever used triangulate and friends for display purposes so
can't claim to be an expert on how accurate "b" will end up - since you
were just picking the nearest point before I guess you're not overly
concerned with accuracy.

The "missing" keyword to TRIGRID might handle your environment_value case.

Sorry for not answering your question as such but I hope this helps or
at least points you in a helpful direction.

Thanks,

Allan

Rainer wrote:
> Hi!
>
> I need to render 2D slices cut from 3D spherical grid data. So far, I
> am using a brute force method which, using two FOR loops, is
> unfortunately rather slow (although working just fine). The problem
> and my approach boil down to this:
>
> -- A[i,j] is the input array, with r[i] and chi[j] being curvilinear
> coordinates (not necessarily uniformly spaced)
> -- B[k,l] is the output array with x[l] and y[l] being uniformly

```
>  spaced Cartesian coordinates (to be plotted with TVSCALE)
>
>  for k=0,nx-1 do begin
>     for l=0,ny-1 do begin
>
>        curr = sqrt(x[k]^2+y[l]^2)
>        curchi = atan(x[k],y[l])
>
>        if "curchi or curr are out of bounds" then begin
>           B[k,l] = environment_value
>           continue
>        endif
>
>        foo = min( r - curr ,nearestr,/absolute)
>        foo = min( chi - curchi ,nearestchi,/absolute)
>
>        B[k,l] = A[nearestr,nearestchi]
>
>     endfor
>  endfor
>
>
```

> The calculation of "curr" and "curchi" can easily be done outside the
> loops of course. But what can I do with the rest?
>
>
> Thanks,
> Rainer