Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by Heinz Stege on Thu, 13 Oct 2016 21:59:52 GMT
View Forum Message <> Reply to Message

On Thu, 13 Oct 2016 08:02:12 -0700 (PDT), Samantha Tushaus wrote:

> Interestingly, my method, the "dilute" method, and the no-ifs loop all take the same amount of time (63, 64, and 62 seconds, respectively).

Congratulation first, you have a very fast computer. My one needs
273 s if data is choosed by calculated by
  data=randomn(seed,3200,3248).

However your code can be optimized. The following needs about 18 s (on
my slow computer):

```
function sam_v02,data
;--------------------
;
temp=size(data,/dimensions)
nx=temp[0]
ny=temp[1]
;
low_ind_x=indgen(nx)-50
low_ind_x[0:49]=0
low_ind_y=indgen(ny)-50
low_ind_y[0:49]=0
hi_ind_x=indgen(nx)+50
hi_ind_x[-50:*]=nx-1
hi_ind_y=indgen(ny)+50
hi_ind_y[-50:*]=ny-1
;
data_type=size(data,/type)
data_max=make_array(nx,ny,type=data_type)
temp_max=make_array(ny,type=data_type)
FOR i=0,nx-1 DO BEGIN
  low_i=low_ind_x[i]
  hi_i=hi_ind_x[i]
  for j=0,ny-1 do temp_max[j]=max(data[low_i:hi_i,j])
  for j=0,ny-1 do $
    data_max[i,j]=max(temp_max[low_ind_y[j]:hi_ind_y[j]])
ENDFOR
;
return,data_max
end
```

First I removed the index calculation from the loop. But that made only a slow difference. Then I splitted the 2-dimensional calculation of the maximum into two 1-dimensional calculations (the two loops over j).

This calculation is more than 15 times faster than the original one.

Cheers, Heinz
