Subject: Memory Allocation Problem
Posted by adhdunn on Tue, 14 Aug 2012 13:34:11 GMT
View Forum Message <> Reply to Message

Hello All,

I am working through a problem with memory allocation in IDL v8.2. I have read that there are some concerns with memory allotment in v8.0 and higher, but I find that there appears to not yet be a fix. Also, I am unable to make adjustments to my .ini file as has been suggested. Apparently I do not have the appropriate access to make changes to that file even though I am an administrator on my computer.

Either way, I am currently reading in a Landsat Image with the dimensions 7931 x 7001 and a NLDAS (GRIB) dataset with dimensions 224 x 464. This portion of the programming is attempting to subset the NLDAS image to the size of the Landsat image (despite the fact that based on dimensions it appears the NLDAS input is smaller but in fact covers the entire U.S., while the Landsat image is one scene in the south eastern part of the country-- have not quite figured out how that works yet). My program is halting at the command below which is at the very end of the posted code:

roi_raddata = INTERPOL(new_result_rad,tot_new_dims)

I will paste the other part of this section of the program below. I thought I might need to use the TEMPORARY command to free up some room but am not sure how to appropriately determine the point at which this should be placed. Any suggestions would be greatly appreciated as always!

Thank you!

Code lines and explanation:

; Extract latitude and longitude variables from the input Landsat and NLDAS data

```
latlon = FLTARR(nx,ny,2)
lat = FLTARR(nx,ny,1)
lon = FLTARR(nx,ny,1)


READU,4,latlon
CLOSE, 4

lat = latlon(*,*,0)
lon = latlon(*,*,1)

lat = REFORM(lat,[nx,ny])
lon = REFORM(lon,[nx,ny])


; Search for datapoints within the region of interest
```

```
z = 0

roi_lat_temp = FLTARR(nmx,1)
roi_lon_temp = FLTARR(nmx,1)
roi_raddata_temp = FLTARR(nmx,1)


roi_lat = FLTARR(nmx,nmy)
roi_lon = FLTARR(nmx,nmy)
roi_raddata = FLTARR(nmx,nmy)


FOR i = 0, nx-1 DO BEGIN
  FOR j = 0, ny - 1 DO BEGIN
  IF (lat[i,j] ge TM_lat_ll) and (lat[i,j] le TM_lat_ul) THEN BEGIN
  IF (lon[i,j] le TM_lon_ll) and (lon[i,j] ge TM_lon_ul) THEN BEGIN
       IF (raddata[i,j] eq 9.999E+20) THEN raddata[i,j] = !VALUES.F_NAN
       roi_lat_temp[z] = lat[i,j]
       roi_lon_temp[z] = lon[i,j]
       roi_raddata_temp[z] = raddata[i,j]

       z = z + 1

  ENDIF
 ENDIF
ENDFOR
ENDFOR

print, 'z = ',z

; This section of the code looks at the original temporary array, which has     ; padded zeros at the
end. This code removes the padded zeros at the end of the
; array and writes the values to a new array.  We will look at the values
; in the temporary lat array, since the radiation values might be
; all equal to zero, depending upon processing hour.

; For radiation data

index_rad = WHERE(roi_lat_temp, count1)

print, 'count1 = ',count1

IF (count1 NE 0) THEN BEGIN
  result_rad = roi_raddata_temp[index_rad]
ENDIF

; Do the same for latitude data to get the dimensions of the new 2-D array
```

```
index_lat = WHERE(roi_lat_temp, count2)


IF (count2 NE 0) THEN BEGIN
  result_lat = roi_lat_temp[index_lat]
ENDIF

;  Count the number of unique elements in roi_temp_lat to find the dimensions
;  of the new 2-D array.

count_lat = result_lat[UNIQ(result_lat, SORT(result_lat))]
print, 'count_lat = ',count_lat

sz_lat = SIZE(count_lat)
n_lat = N_ELEMENTS(count_lat)

;  Get new array dimensions

lx = z/n_lat
ly = z/lx

print, 'The dimensions of the new 2-D array are: ', lx,ly

; Reformat the original 1-D array to a 2-D array

new_result_rad = FLTARR(lx,ly)


new_result_rad = REFORM(result_rad,[lx,ly])


;  Interpolate to a bigger array

roi_raddata = INTERPOL(new_result_rad,tot_new_dims)
roi_raddata = REFORM(roi_raddata,[nmx,nmy,1])

;  Set up an array of nswrs and nlwrs to pass back to the main program.  This
;  array will roi_raddata(*,*,i)


print, 'returning to the main program'

CLOSE, /ALL

RETURN, roi_raddata

END
```