Subject: Another "IDL way to do this" question
Posted by Fabzou on Tue, 08 Nov 2011 14:50:23 GMT
View Forum Message <> Reply to Message

Dear IDLers,

I have to interpolate modelled fields from eta to pressure coordinates, which is a relatively easy thing to do when using a simple linear approach. Since my 3D grid is irregular "only" on the third dimension (the fourth dimension being time), well I did not find any other solution than looping on the three regular dimensions and use INTERPOL for each vertical column.(interpol_3d in the code posted below)

This works fine, but is not very very fast.

Then I tried to loop on the time dimension outside the "main loop" (interpol_3d_alt in the code posted below), but results are only slightly better, as shown if I run the test program compare_interp:

% Compiled module: INTERPOL_3D.
% Compiled module: INTERPOL_3D_ALT.
% Compiled module: COMPARE_INTERP.
IDL> compare_interp
Method1: 19.713714
Method2: 18.455919

Do you have an idea how to make this faster? I thought about other types of 3D interpolation, but they all seem so "overkill" for such a simple problem...

Thanks a lot

Fab

```
;+
; :Description:
;    Interpolates model data vertically (height or pressure levels).
;
; :Params:
;    varin: in, float, required
;          Data on model levels that will be
;          interpolated[nx,ny,nz,(nt)]
;    z_in: in, float, required
;         Array of vertical coordinate to interpolate into.
;         This must either be pressure/height.
;         Dimensions must be the same as those of `varin`.
;    loc_param: in, float, required
```

```
;              the locatations to interpolate to
;               (typically pressure or height)
;
; :Returns:
;     Data interpolated to horizontal plane(s), with the third dimension
;     being equal to the number of elements in `loc_param`
;
;-
function interpol_3d, varin, z_in, loc_param

   ; Set Up environnement
   COMPILE_OPT idl2

   dims = SIZE(varin, /DIMENSIONS)
   nd = N_ELEMENTS(dims)
   nlocs = N_ELEMENTS(loc_param)

   if nd eq 3 then begin ; no time dimension
     out_var = FLTARR(dims[0], dims[1], nlocs)
     for i=0, dims[0]-1 do begin
       for j=0, dims[1]-1 do begin
         _z = z_in[i,j,*]
         out_var[i,j,*] = INTERPOL(varin[i,j,*],_z,loc_param)
         p = where(loc_param gt max(_z) or loc_param lt min(_z), cnt)
         if cnt ne 0 then out_var[i,j,p] = !VALUES.F_NAN
       endfor
     endfor
   endif else if nd eq 4 then begin
     out_var = FLTARR(dims[0], dims[1], nlocs, dims[3])
     for i=0, dims[0]-1 do begin
       for j=0, dims[1]-1 do begin
         for t=0, dims[3]-1 do begin
           _z = z_in[i,j,*,t]
           out_var[i,j,*,t] = INTERPOL(varin[i,j,*,t],_z,loc_param)
           p = where(loc_param gt max(_z) or loc_param lt min(_z), cnt)
           if cnt ne 0 then  out_var[i,j,p,t] = !VALUES.F_NAN
         endfor
       endfor
     endfor
   endif else Message, 'VARIN dimensions should be 3 or 4'

   return, out_var

end

function interpol_3d_alt, varin, z_in, loc_param

   ; Set Up environnement
```

```
  COMPILE_OPT idl2

  dims = SIZE(varin, /DIMENSIONS)
  nd = N_ELEMENTS(dims)
  nlocs = N_ELEMENTS(loc_param)

  if nd eq 3 then begin ; no time dimension
    out_var = FLTARR(dims[0], dims[1], nlocs)
    for i=0, dims[0]-1 do begin
      for j=0, dims[1]-1 do begin
        _z = z_in[i,j,*]
        out_var[i,j,*] = INTERPOL(varin[i,j,*],_z,loc_param)
        p = where(loc_param gt max(_z) or loc_param lt min(_z), cnt)
        if cnt ne 0 then out_var[i,j,p] = !VALUES.F_NAN
      endfor
    endfor
  endif else if nd eq 4 then begin
    out_var = FLTARR(dims[0], dims[1], nlocs, dims[3])
    for t=0, dims[3]-1 do $
        out_var[*,*,*,t]= interpol_3d_alt(reform(varin[*,*,*,t]), $
          reform(z_in[*,*,*,t]), loc_param)
  endif else Message, 'VARIN dimensions should be 3 or 4'

  return, out_var

end

pro compare_interp

  varin = FLTARR(200,200,27,24)
  p = LINDGEN(200,200,27,24) * 1.

  pressure_levels = [850., 700., 500., 300.]
  t1 = SYSTIME(/SECONDS)
  one = interpol_3d(varin, p, pressure_levels)
  print, 'Method1: ' + str_equiv(SYSTIME(/SECONDS) - t1)
  t1 = SYSTIME(/SECONDS)
  two = interpol_3d_alt(varin, p, pressure_levels)
  print, 'Method2: ' + str_equiv(SYSTIME(/SECONDS) - t1)

end
```