Subject: Interpolate with findex no longer faster than interpol?
Posted by Trae on Sat, 11 Aug 2007 17:16:13 GMT
View Forum Message <> Reply to Message

I found out something interesting about IDL interpolation schemes recently that I thought might be of interest to others.

The experiment:
I ran two versions of a script to test interpolation schemes. One version has a well refined initial grid and a sparse grid to interpolate onto. The second version has a sparse initial grid and a well refined abscissa to interpolate onto. Both of these tests are based on the test in the header of findex.pro.

The script tested the time in took to run 3 different interpolation schemes and the accuracy of each. 1) Using interpolate with findex, 2) Using interpol alone, 3) Using a highly optimized, and efficient C routine that is called via call_external.

(The code follows at the bottom, minus the calls to the C routine which is specialized for a specific computer. Note that the sorting of the test data is handled differently in the two codes. This was done for comparisons to specific calculations I was doing. You may want to change.)

The results:
To my surprise the straight call to interpol alone won hands down every time! The header for the findex routine states that using findex in conjunction with interpolate can yield a factor of 60 increase of efficiency over using interpol alone. A factor of 70 is stated on David Fanning's amazingly helpful website. However, this now longer seems to be the case.

With a well refined grid initial grid the time savings was a factor of ~180 if I just used interpol instead of interpol with findex. With a sparse initial grid the time savings was a factor of ~5.5. The results with the C code were similar but the C code was worse than either of the IDL interpolators.

For the C code I think the explanation is that the computational overhead of using call_external offsets any other efficiencies. For why interpol now works faster than interpolate with findex, I simply don't know. Findex has a repeat loop in it. Could it be that any efficiency incurred by findex is offset by calling another function and a repeat loop in that function?

Anyway, this discovery has saved me many hours of computing time. Hopefully, it will help others. Please test this and let me know your

results. I find this to be very curious.

Cheers,
-Trae

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;.r findex_test
;Old x
u=randomu(iseed,200000) & u=u[sort(u)]
;New x
v=randomu(iseed,10)     & v=v[sort(v)]
;Old y
y=randomu(iseed,200000) & y=y[sort(y)]

t=systime(1) & y1=interpolate(y,findex(u,v)) & findex_time=systime(1)-
t
print,'Findex time',findex_time
t=systime(1) & y2=interpol(y,u,v) & no_findex_time=systime(1)-t
print,'No Findex time',no_findex_time

print,''
;print,f='(3(a,10f7.4/))',$
;      'findex:     ',y1,$
;      'interpol:   ',y2,$
;      'diff(y1-y2):',y1-y2
;print,''
print, 'With Findex/Without Findex time ratio:', findex_time/
no_findex_time


plot, u,y

oplot, v, y1, psym=1
oplot, v, y2, psym=2

end ;Of findex_test
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;.r findex_test3a
;; In this example I found the FINDEX + INTERPOLATE combination
;; to be about 60 times faster then INTERPOL.
;
;Old x
u=double([0.0, 1.0,2.0, 3.0,4.0, 5.0])
;New x
v=randomu(iseed,20000)*5d ;& v=v[sort(v)]

;Old y

y=sin((!dpi/2.)*(U/5.))*exp((2.5-u))
```

```
t=systime(1) & y1=interpolate(y,findex(u,v)) & findex_time=systime(1)-
t
print,'Findex time',findex_time
t=systime(1) & y2=interpol(y,u,v) & no_findex_time=systime(1)-t
print,'No Findex time',no_findex_time

print, 'With Findex/Without Findex time ratio:', findex_time/
no_findex_time

plot, u,y, psym=2

i=sort(v)
oplot, v[i], y1[i], line=1
oplot, v[i], y2[i], line=2
;oplot, v[i], y3[i], line=3

end; findex_test3a
```