Subject: Searching for fast linear interpolation routine
Posted by Roger J. Dejus on Fri, 04 Apr 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Did someone write a fast linear interpolation routine for irregular one
dimensional arrays (monotonically ascending or descending abscissas)
similar in functionality to the INTERPOL.PRO routine from RSI?

The enclosed test code takes about 1.6 s on a DEC alpha (old = model
3000/400). In reality, the arrays I'm using are of course much larger
and the bottleneck is really in INTERPOL.PRO.

Regards, Roger Dejus. (dejus@aps.anl.gov).

```
PRO test6,x,y,xi,yi

x  = findgen(1001)/1000.0*2.0*!pi
y  = sin(x)
xi = findgen(10001)/10000.0*2.0*!pi

t  = systime(1)
yi = interpol(y,x,xi)
print,systime(1)-t

END ; test6
```

Subject: Re: Searching for fast linear interpolation routine
Posted by Christian Marquardt on Sat, 05 Apr 1997 08:00:00 GMT
View Forum Message <> Reply to Message

This is a multi-part message in MIME format.

--------------614427971AAD9DC964589125
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Hello,

Liam Gumley wrote:
>
> Roger J. Dejus wrote:
>> Did someone write a fast linear interpolation routine for irregular one
>> dimensional arrays (monotonically ascending or descending abscissas)
>> similar in functionality to the INTERPOL.PRO routine from RSI?
>
> If you can find a way to use the built-in routine INTERPOLATE, you will

> see a dramatic speed increase - it is very fast on 1D, 2D or 3D arrays.
>
> Cheers,
> Liam.

Recently, Paul Richiazzi posted a solution to this problem to the
newsgroup; I'll attach his posting...

Hope this helps,

Chris Marquardt.

--------------614427971AAD9DC964589125
Content-Type: text/plain; charset=us-ascii; name="findex.email"
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="findex.email"

Path: fu-berlin.de!news.apfel.de!news.maxwell.syr.edu!news.bc.net!
info.ucla.edu!nnrp.info.ucla.edu!ucsbuxb.ucsb.edu!ucsbuxb.uc sb.edu!paul
From: paul@orion.crseo.ucsb.edu (Paul Ricchiazzi)
Newsgroups: comp.lang.idl-pvwave
Subject: A faster way to INTERPOL
Date: 21 Feb 1997 21:30:44 GMT
Organization: University of California, Santa Barbara
Lines: 130
Distribution: global
Message-ID: <PAUL.97Feb21133044@orion.crseo.ucsb.edu>
NNTP-Posting-Host: orion.crseo.ucsb.edu

I find that using INTERPOL to do 1-d interpolations on large irregular
grids can be extremely time consuming. The problem with INTERPOL is
that it uses a linear search to find where a given field point fits
into the irregular grid. Below you'll find my solution to the
problem. The procedure FINDEX uses a binary search to obtain a
"floating point index" which can be used with INTERPOLATE. I have
found that the FINDEX + INTERPOLATE method can be up to 70 times
faster than using INTERPOL. I am donating this procedure to the IDL
community in hopes of saving untold millions of machine cycles that
would otherwise have been wasted in futile linear searches. But
seriously, give it a try and let me know if it breaks.

Regards,

Paul Ricchiazzi

---------8<---------8<---------8<---------8<---------8<---------8 <---------8<

```
function findex,u,v
;+
; ROUTINE:  findex
;
; PURPOSE:  Compute "floating point index" into a table using binary
;           search.  The resulting output may be used with INTERPOLATE.
;
; USEAGE:   result = findex(u,v)
;
; INPUT:
;   u       a monitically increasing or decreasing 1-D grid
;   v       a scalor, or array of values
;
; OUTPUT:
;   result  Floating point index. Integer part of RESULT(i) gives
;           the index into to U such that V(i) is between
;           U(RESULT(i)) and U(RESULT(i)+1).  The fractional part
;           is the weighting factor
;
;                         V(i)-U(RESULT(i))
;                         ---------------------
;                     U(RESULT(i)+1)-U(RESULT(i))
;
;
; DISCUSSION:
;           This routine is used to expedite one dimensional
;           interpolation on irregular 1-d grids.  Using this routine
;           with INTERPOLATE is much faster then IDL's INTERPOL
;           procedure because it uses a binary instead of linear
;           search algorithm.  The speedup is even more dramatic when
;           the same independent variable (V) and grid (U) are used
;           for several dependent variable interpolations.
;
;
; EXAMPLE:
;
;; In this example I found the FINDEX + INTERPOLATE combination
;; to be about 60 times faster then INTERPOL.
;
;  u=randomu(iseed,200000) & u=u(sort(u))
;  v=randomu(iseed,10)     & v=v(sort(v))
;  y=randomu(iseed,200000) & y=y(sort(y))
;
;  t=systime(1) & y1=interpolate(y,findex(u,v)) & print,systime(1)-t
;  t=systime(1) & y2=interpol(y,u,v)            & print,systime(1)-t
;  print,f='(3(a,10f7.4/))','findex:   ',y1,'interpol: ',y2,'diff:     ',y1-y2
;
```

```
; AUTHOR:   Paul Ricchiazzi                     21 Feb 97
;           Institute for Computational Earth System Science
;           University of California, Santa Barbara
;           paul@icess.ucsb.edu
;
; REVISIONS:
;
;-
;
nu=n_elements(u)
nv=n_elements(v)

us=u-shift(u,+1)
us=us(1:*)
umx=max(us,min=umn)
if umx gt 0 and umn lt 0 then message,'u must be monotonic'
if umx gt 0 then inc=1 else inc=0

maxcomp=fix(alog(float(nu))/alog(2.)+.5)

; maxcomp = maximum number of binary search iteratios

jlim=lonarr(2,nv)
jlim(0,*)=0         ; array of lower limits
jlim(1,*)=nu-1      ; array of upper limits

iter=0
repeat begin
  jj=(jlim(0,*)+jlim(1,*))/2
  ii=where(v ge u(jj),n) & if n gt 0 then jlim(1-inc,ii)=jj(ii)
  ii=where(v lt u(jj),n) & if n gt 0 then jlim(inc,ii)=jj(ii)
  jdif=max(jlim(1,*)-jlim(0,*))
  if iter gt maxcomp then begin
    print,maxcomp,iter, jdif
    message,'binary search failed'
  endif
  iter=iter+1
endrep until jdif eq 1

w=v-v
 w(*)=(v-u(jlim(0,*)))/(u(jlim(0,*)+1)-u(jlim(0,*)))+jlim(0,* )

return,w
end
```

--

Paul Ricchiazzi
Institute for Computational Earth System Science (ICESS)
University of California, Santa Barbara

email: paul@icess.ucsb.edu

--
```
 ----------------------------------------------------------- ---------------
  Christian Marquardt

  Meteorologisches Institut der   |  tel.: (+49) 30-838-71170
  Freien Universitaet Berlin      |  fax.: (+49) 30-838-71167
  Carl-Heinrich-Becker-Weg 6-10   |  email: marq@strat01.met.fu-berlin.de
  D-12165 Berlin                  |
 ----------------------------------------------------------- ---------------
```

--------------614427971AAD9DC964589125--

Subject: Re: Searching for fast linear interpolation routine
Posted by Liam Gumley on Sat, 05 Apr 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Roger J. Dejus wrote:
> Did someone write a fast linear interpolation routine for irregular one
> dimensional arrays (monotonically ascending or descending abscissas)
> similar in functionality to the INTERPOL.PRO routine from RSI?

If you can find a way to use the built-in routine INTERPOLATE, you will
see a dramatic speed increase - it is very fast on 1D, 2D or 3D arrays.

Cheers,
Liam.

Subject: Re: Searching for fast linear interpolation routine
Posted by Christian Soeller on Sun, 06 Apr 1997 08:00:00 GMT

View Forum Message <> Reply to Message

Christian Marquardt <marq@strat01.met.fu-berlin.de> writes:

>
> I find that using INTERPOL to do 1-d interpolations on large irregular
> grids can be extremely time consuming. The problem with INTERPOL is
> that it uses a linear search to find where a given field point fits
> into the irregular grid. Below you'll find my solution to the
> problem. The procedure FINDEX uses a binary search to obtain a
> "floating point index" which can be used with INTERPOLATE. I have
> found that the FINDEX + INTERPOLATE method can be up to 70 times
> faster than using INTERPOL. I am donating this procedure to the IDL
> community in hopes of saving untold millions of machine cycles that
> would otherwise have been wasted in futile linear searches. But
> seriously, give it a try and let me know if it breaks.

If the indices you are calculating interpolates at are often close to each
other you can even speed up the binary search some further by using the last
calculated index as a starting guess for the next, see "How to search an
ordered table" in "Numerical Recipes" (now available as an online web
document), especially the hunt routine. I have implemented that method
in a 'call external' like way in another data language (PDL = Perl
Data Language == free(!)) and get for the equivalent of the published
'test6' example an execution time of 0.016s compared to 2.09s to the
IDL interpolate example. So more than a factor of 100. If you go for
the ultimate speed than a 'call external' (or if you prefer
'linkimage') implementation using hunt to find the indices should be
the way to go.

Christian

-- --------
Christian Soeller mailto: csoelle@sghms.ac.uk
St. Georges Hospital Medical School Dept. of Pharmacology
Cranmer Terrace London SW17 0RE

Subject: Re: Searching for fast linear interpolation routine
Posted by David Crain on Mon, 07 Apr 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Christian Marquardt wrote:
>
> Hello,
>
> Liam Gumley wrote:
>>

>> Roger J. Dejus wrote:
>>> Did someone write a fast linear interpolation routine for irregular one
>>> dimensional arrays (monotonically ascending or descending abscissas)
>>> similar in functionality to the INTERPOL.PRO routine from RSI?
>>
>> If you can find a way to use the built-in routine INTERPOLATE, you will
>> see a dramatic speed increase - it is very fast on 1D, 2D or 3D arrays.
>>
>> Cheers,
>> Liam.
>
> Recently, Paul Richiazzi posted a solution to this problem to the
> newsgroup; I'll attach his posting...
>
> Hope this helps,
>
>   Chris Marquardt.
>
>   -- ---
> Path: fu-berlin.de!news.apfel.de!news.maxwell.syr.edu!news.bc.net!
info.ucla.edu!nnrp.info.ucla.edu!ucsbuxb.ucsb.edu!ucsbuxb.uc sb.edu!paul
> From: paul@orion.crseo.ucsb.edu (Paul Ricchiazzi)
> Newsgroups: comp.lang.idl-pvwave
> Subject: A faster way to INTERPOL
> Date: 21 Feb 1997 21:30:44 GMT
> Organization: University of California, Santa Barbara
> Lines: 130
> Distribution: global
> Message-ID: <PAUL.97Feb21133044@orion.crseo.ucsb.edu>
> NNTP-Posting-Host: orion.crseo.ucsb.edu
>
> I find that using INTERPOL to do 1-d interpolations on large irregular
> grids can be extremely time consuming. The problem with INTERPOL is
> that it uses a linear search to find where a given field point fits
> into the irregular grid. Below you'll find my solution to the
> problem. The procedure FINDEX uses a binary search to obtain a
> "floating point index" which can be used with INTERPOLATE. I have
> found that the FINDEX + INTERPOLATE method can be up to 70 times
> faster than using INTERPOL. I am donating this procedure to the IDL
> community in hopes of saving untold millions of machine cycles that
> would otherwise have been wasted in futile linear searches. But
> seriously, give it a try and let me know if it breaks.
>
> Regards,
>
> Paul Ricchiazzi
>
> ---------8<---------8<---------8<---------8<---------8<---------8 <---------8<

```
>
>  function findex,u,v
>  ;+
>  ; ROUTINE:  findex
>  ;
>  ; PURPOSE:  Compute "floating point index" into a table using binary
>  ;           search.  The resulting output may be used with INTERPOLATE.
>  ;
>  ; USEAGE:   result = findex(u,v)
>  ;
>  ; INPUT:
>  ;   u       a monitically increasing or decreasing 1-D grid
>  ;   v       a scalor, or array of values
>  ;
>  ; OUTPUT:
>  ;   result  Floating point index. Integer part of RESULT(i) gives
>  ;           the index into to U such that V(i) is between
>  ;           U(RESULT(i)) and U(RESULT(i)+1).  The fractional part
>  ;           is the weighting factor
>  ;
>  ;                          V(i)-U(RESULT(i))
>  ;                          ---------------------
>  ;                       U(RESULT(i)+1)-U(RESULT(i))
>  ;
>  ;
>  ; DISCUSSION:
>  ;           This routine is used to expedite one dimensional
>  ;           interpolation on irregular 1-d grids.  Using this routine
>  ;           with INTERPOLATE is much faster then IDL's INTERPOL
>  ;           procedure because it uses a binary instead of linear
>  ;           search algorithm.  The speedup is even more dramatic when
>  ;           the same independent variable (V) and grid (U) are used
>  ;           for several dependent variable interpolations.
>  ;
>  ;
>  ; EXAMPLE:
>  ;
>  ;; In this example I found the FINDEX + INTERPOLATE combination
>  ;; to be about 60 times faster then INTERPOL.
>  ;
>  ;  u=randomu(iseed,200000) & u=u(sort(u))
>  ;  v=randomu(iseed,10)     & v=v(sort(v))
>  ;  y=randomu(iseed,200000) & y=y(sort(y))
>  ;
>  ;  t=systime(1) & y1=interpolate(y,findex(u,v)) & print,systime(1)-t
>  ;  t=systime(1) & y2=interpol(y,u,v)            & print,systime(1)-t
>  ;  print,f='(3(a,10f7.4/))','findex:   ',y1,'interpol: ',y2,'diff:     ',y1-y2
>  ;
> ; AUTHOR:   Paul Ricchiazzi                         21 Feb 97
> ;           Institute for Computational Earth System Science
> ;           University of California, Santa Barbara
> ;           paul@icess.ucsb.edu
> ;
> ; REVISIONS:
> ;
> ;-
> ;
> nu=n_elements(u)
> nv=n_elements(v)
>
> us=u-shift(u,+1)
> us=us(1:*)
> umx=max(us,min=umn)
> if umx gt 0 and umn lt 0 then message,'u must be monotonic'
> if umx gt 0 then inc=1 else inc=0
>
> maxcomp=fix(alog(float(nu))/alog(2.)+.5)
>
> ; maxcomp = maximum number of binary search iteratios
>
> jlim=lonarr(2,nv)
> jlim(0,*)=0          ; array of lower limits
> jlim(1,*)=nu-1       ; array of upper limits
>
> iter=0
> repeat begin
>   jj=(jlim(0,*)+jlim(1,*))/2
>   ii=where(v ge u(jj),n) & if n gt 0 then jlim(1-inc,ii)=jj(ii)
>   ii=where(v lt u(jj),n) & if n gt 0 then jlim(inc,ii)=jj(ii)
>   jdif=max(jlim(1,*)-jlim(0,*))
>   if iter gt maxcomp then begin
>     print,maxcomp,iter, jdif
>     message,'binary search failed'
>   endif
>   iter=iter+1
> endrep until jdif eq 1
>
> w=v-v
>  w(*)=(v-u(jlim(0,*)))/(u(jlim(0,*)+1)-u(jlim(0,*)))+jlim(0,* )
>
> return,w
> end
>
> --
>
> ____________________________________________________________
```

> Paul Ricchiazzi
> Institute for Computational Earth System Science (ICESS)
> University of California, Santa Barbara
>
> email: paul@icess.ucsb.edu
> ____ ______________
>
> --
> -- ---------------
> Christian Marquardt
>
> Meteorologisches Institut der | tel.: (+49) 30-838-71170
> Freien Universitaet Berlin | fax.: (+49) 30-838-71167
> Carl-Heinrich-Becker-Weg 6-10 | email: marq@strat01.met.fu-berlin.de
> D-12165 Berlin |
> -- ---------------
Yes, more than a factor of 2 quicker using the same test6.pro described above!

David Crain

Subject: Re: Searching for fast linear interpolation routine
Posted by Roger J. Dejus on Mon, 07 Apr 1997 07:00:00 GMT
View Forum Message <> Reply to Message

In response to suggestions for a fast linear interpolation routine:

1) Liam Gumley wrote:
> If you can find a way to use the built-in routine INTERPOLATE, you
> will see a dramatic speed increase - it is very fast on 1D, 2D or 3D
> arrays.

-- Yes, I can use INTERPOLATE rather than INTERPOL and there was a noticable speed increase. Note, the time will now be determined by the time spent in the routine preparing to call INTERPOLATE (see 2) and 3) below).

2) Wayne Landsman wrote:
> The INTERPOL function consists of two steps: (1) finding the effective
> index of the interpolation value (e.g. it is located between indicies
> 12 and 13 in the abscissa), and (2) performing the interpolation. I
> know of at least 3 ways in which the speed of the INTERPOL function
> can be improved:
>
> (1) INTERPOL uses a incremental search algorithm to find the
> effective index, whereas the quickest way to search a monotonic array

> is a binary search (divide and conquer).
> (2) the effective index search is not vectorized
> (3) the intrinsic INTERPOLATE function (available since V2.2) is not
> used to do the interpolation
>
> The program LINTERP in the IDL Astronomy Library incorporates these
> three improvements and I get a factor of four improvement in speed on
> my Ultra-2.
>
> http://idlastro.gsfc.nasa.gov/ftp/pro/math/linterp.pro
>
> LINTERP calls the program TABINV to find the effective index
>
> http://idlastro.gsfc.nasa.gov/ftp/pro/math/tabinv.pro
>
> These procedures also call ISARRAY (from the JHUAPL library) and
> ZPARCHECK
>
> http://idlastro.gsfc.nasa.gov/ftp/pro/jhuapl/isarray.pro
> http://idlastro.gsfc.nasa.gov/ftp/pro/misc/zparcheck.pro

-- I agree, and I also got a factor of four speed increase on my DEC
Alpha using LINTERP (and TABINV, ISARRAY, and ZPARCHECK).

3) Chris Marquardt wrote:
> Recently, Paul Richiazzi posted a solution to this problem to the
> newsgroup; I'll attach his posting...
>
> Hope this helps,
>
> Chris Marquardt.
>

-- The solution from Paul Richiazzi (using the routine called FINDEX) is
similar to the solution by Wayne Landsman (FINDEX replaces the call to
TABINV above), although slower by a factor of two (for my sample
problem). The FINDEX routine will show a speed increase by about 25% by
switching indices so that the array JLIM=LONARR(2,NV) is replaced by
JLIM=LONARR(NV,2). Remember, IDL (and FORTRAN) are column oriented =
first index varies the fastest.

4) Christian Soeller wrote:
> If the indices you are calculating interpolates at are often close to each
> other you can even speed up the binary search some further by using the last
> calculated index as a starting guess for the next, see "How to search an
> ordered table" in "Numerical Recipes" (now available as an online web
> document), especially the hunt routine. I have implemented that method

> in a 'call external' like way in another data language (PDL = Perl
> Data Language == free(!)) and get for the equivalent of the published
> 'test6' example an execution time of 0.016s compared to 2.09s to the
> IDL interpolate example. So more than a factor of 100. If you go for
> the ultimate speed than a 'call external' (or if you prefer
> 'linkimage') implementation using hunt to find the indices should be
> the way to go.
>
> Christian
>

-- Yes, the starting index (guess) should be the previous index when calculating indices which are close. I don't know whether this can be vectorized though. Also, I would like to see this implemented in an IDL routine rather than calling an external program (which I'm familiar with but it is more straightforward to call an IDL program and it causes less problems when making portable codes). I'm also questioning the quoted time above of 0.016 s which is approx. the time spent in the routine INTERPOLATE without first calling an index searching routine (such as TABINV or FINDEX above).

Please feel free to comment on the above.
Thanks, Roger. (dejus@aps.anl.gov).

Subject: Re: Searching for fast linear interpolation routine
Posted by paul on Wed, 16 Apr 1997 07:00:00 GMT

View Forum Message <> Reply to Message

I just wanted to point out that the version of FINDEX posted here a few weeks ago can be sped up by about 25% by reversing the indexing order of the variable JLIM. That is, by replacing JLIM=LONARR(2,NV) with JLIM=LONARR(NV,2). This is because IDL, (like FORTRAN) is column oriented -- first index varies the fastest. Thanks to Roger Dejus for this suggestion.

--

Paul Ricchiazzi
Institute for Computational Earth System Science (ICESS)
University of California, Santa Barbara

email: paul@icess.ucsb.edu