Subject: Multidimensional Interpolation
Posted by ian on Mon, 21 Jul 2003 15:40:58 GMT
View Forum Message <> Reply to Message

Hi,

I have created a 5 dimensional data cube (pressure, temperature, relative humidity, frequency, transmission) with a radiative transfer model. I have a user that will need to get transmission data for given values of the rest of the parameters, so I am currently planning to interpolate the cube to the input values of the user.

Does anyone know of any multi-dimensional interpolation routines (similar to spline) that would be able to perform this task?

-Ian

Subject: Re: Multidimensional Interpolation
Posted by JD Smith on Tue, 22 Jul 2003 16:20:03 GMT
View Forum Message <> Reply to Message

On Tue, 22 Jul 2003 05:39:49 -0700, Paul van Delst wrote:
> Haje Korth wrote:
>>
>> JD,
>> could you give me the source for your solution. Are you aware of a good
>> textbook on this matter?
>
> I would be interested also. I want to do some 2-D (or is it 3-D?)
> interpolation but using bicubic interpolation. My (feeble) initial
> attempts to figure out how have led me to my NR book, but the method of
> deriving the coefficients to do the interpolation is brazenly swept
> under the rug.

You'll notice I said "could be written", not "has been written" ;).
Anyway, I cobbled something together:

http://turtle.as.arizona.edu/idl/ninterpolate.pro

Beware of the restrictions listed. And Paul, this is *linear* interpolation only... it works well if your function is mostly linear in all directions at the point being interpolated. Also be warned that this is slow... about 40x slower at tri-linear than INTERPOLATE itself. Much of this slowness could be mitigated by pre-computing or saving the various index arrays used to get the right pattern signatures for a given dimension. Like INTERPOLATE, this uses fractional array indices as the target interpolation coordinate (e.g. [0..ndim1-1,0..ndim2-1,

... ,0..ndimn-1]). If you want INTERPOL-like functionality (each dimension corresponding to a specified regular or irregular grid of true data values), you'd first use VALUE_LOCATE to find a bracketing pair of indices in all directions, and then convert your point to a fractional array index using the grid values at the bracketing points.

As a side note, this routine also illustrates a remarkable point about just how obfuscated vectorized IDL code can be, when compared to a straightforward (but dog-slow, in IDL) looping method. Essentially, all I'm really doing is (in C):

```
long v0,x0,y0,z0;
float a,b,c,d;
float total=0.;
v0=floor(v); x0=floor(x); y0=floor(y); z0=floor(z);
a=v-v0; b=x-x0; c=y-y0; d=z-z0;
for(i=0;i<=1;i++) {
  for(j=0;j<=1;j++) {
    for(k=0;k<=1;k++) {
      for(l=0;l<=1;l++) {
 total+= (i?a:(1-a)) * (j?b:(1-b)) * (k?c:(1-c)) * (l?d:(1-d)) *
   data[v0+i,x0+j,y0+k,z0+l];
      }
    }
  }
}
```

Well, not exactly true, since I'm actually doing it for an arbitrary dimension, not just 4, but the mechanics are the same. All that REBIN stuff is just to generate an appropriate pattern of signatures that, for a given dimension, looks just like the above.

Note that n-cubic (or spline, etc.) interpolation could be had by this method too. For cubic, instead of using 2^n points in the interpolation, you use 4^n points, with weights that aren't linear, but instead look like:

```
w=1-2t^2+t^3     if t<1
w=4-8t+5t^2-t^3  if 1<=t<2
0                otherwise
```

where

t=abs(i-x)

i.e. the absolute fractional deviation of a given pixel being used from the interpolation point. In n dimensions, you have the sum of the 4^n data values, each weighted by the product of n such weights.

Expressed in this way, our linear interpolation is just:

w=t if t<=1
0 otherwise

Also a side note, and without too much technical detail, an excellent article by the German mathematics professor Helmut Dersch (author of the superb but industry-persecuted PanoTools panoramic photography toolset) regarding the real-world performance of different types of interpolation algorithms is available at:

http://www.path.unimelb.edu.au/~dersch/interpolator/interpol ator.html

JD
