Subject: Re: Setting up offsets array ...
Posted by John-David T. Smith on Wed, 18 Apr 2001 14:45:08 GMT
View Forum Message <> Reply to Message

Peter Thorne wrote:
>
> Dear all,
>
> after your excellent help last time I thought that I would test the
> waters again to see whether you can help me on my next problem. You'll
> all get an acknowledgement in my thesis when (if?) I eventually finish
> :-). This time I think the problem is really rather trivial I'm afraid
> (famous last words?) - sorry.
>
> The problem I now have is that I want to set up an offsets array to test
> the sensitivity to the choice of histogram origin of the solution in my
> last problem. Therefore, again one is working in space of unknown
> dimensionality of between 1 and 5, which is known within the program
> only upon call to it from a higher level program. What I want to be able
> to do is set up an offsets array so that I can systematically shift the
> origin over an entire space step in each dimension. For a 1-D problem
> this is simple:
>
> offsets=(findgen(11)*0.1)-0.5
>
> Which can then be used to force the offset of the 1-D histogram around
> the true origin by between -0.5 and 0.5 space units in the calling
> program to JD's excellent hist_nd. The problem is how to generalise this
> in the program to a potentially higher, and unknown (until call)
> dimensional space. The number of combinations of shifts will be 11^n,
> where n is the dimensionality of the histogram space. For a 2-D case the
> offsets array will look like:

> For each extra dimension another column would be added to the array and
> the number of rows increased by *11. Therefore the offsets array will
> always be fltarr(n,11^n) in size, it is the filling of this array with
> the values that is causing me my latest headache :-(
>
> Any ideas as to how to approach this problem will be very gratefully
> received.

```
ind=lindgen(11L^n,n)
fac=11L^(ind/11L^n)
offs=float((ind mod (11L*fac))/fac)/10.-.5
```

The basic idea is to use mod to wrap around faster on the rapidly
changing rows, and slower on the slowly changing rows, then bring
everything into the range [0-10] by long integer division. Insert the

usual temporary() baloney if you'd like a modest speedup with increased ugliness.

JD

---