Subject: Re: Best way to generate arrays of coordinates for hypersurface calculations?
Posted by James[2] on Wed, 07 Apr 2010 23:20:35 GMT
View Forum Message <> Reply to Message

I've been working with this problem some more, and I found some dimensional juggling info on David Fanning's website that helped me make these arrays in a more straightforward, and flexible, way. I just create a one-dimensional vector for each axis and pad it out with a bunch of empty dimensions using REFORM. Then, I can stretch out the 1D vector using REBIN to occupy the entire multidimensional space.

I wrote a program to make these coordinate arrays, which I've posted here: http://pastebin.com/KVew8M0g . This program includes error checking on the max/min arguments and type argument. The "meat" is actually quite brief, however:

```
function coordinates, mins, maxes, type
   ...error checking...

   sizes = maxes - mins + 1
   out = ptrarr(dims)

   for i=0, dims[0]-1 do begin
      ;make a 1D vector containing the indices for this dimension
      out[i] = ptr_new(make_array(sizes[i], /index, type=type))
      ;shift index vector it contains values from min to max,
inclusive
      *out[i] +=+ mins[i]
      ;make a vector to feed into reform for dimensional juggling
      rvec = replicate(1,i+1)
      ;place the index vector into the proper dimension
      rvec[i] = sizes[i]
      ;replicate the index vector across all dimensions
      *out[i] = rebin(reform(*out[i], rvec), sizes)
   endfor
   return, out
end
```

I'd like to add a few more features: mainly, the ability to make floating point arrays with a range that is smaller than the number of elements, like a 100x100x100 unit cube for instance. I think this would make the function quite a bit more complicated, but also a lot more complete. I'm also still uneasy about calculating functions this way. It seems like a huge waste of memory, but I can't think of any other way to do it without using many loops!

James Preiss