Subject: Re: Optimizing a lookup table
Posted by Brian Larsen on Sat, 03 Mar 2007 17:40:20 GMT
View Forum Message <> Reply to Message

IP, :)

I always tend to lean toward the disk read as then you don't have any unwieldy issues with size and doubles and you can do structures. I find that using a good solution and just using it forever is easiest on my simple brain.

While it certainly won't work in all situations I often solve it this way.

```
if n_elements(lookup) eq 0 then restore, 'file.sav'

for i=0l, big-1 do begin
  ;; do something cool,
  cool = my_function(params, lookup)
endfor
```

This way you have to rewrite my_function to accept the lookup table as a param but you only have to do the read once, which is good because disk access is slow.

This is mainly my bias for almost never using system vars or common blocks, mainly because I hate having namespace issues that I wasn't expecting and it always seems to happen to me.

Brian

---------- ---------
Brian A. Larsen
Dept. of Physics
Space Science and Engineering Lab (SSEL)
Montana State University - Bozeman
Bozeman, MT 59717

On Mar 2, 6:15 pm, ianpaul.free...@gmail.com wrote:
> I would like to write a function that uses a look-up table. Now the
> easiest (coding-wise), is to just save the look-up table and have the
> function restore it. However, I plan to be calling this function in a

> loop, so this would result in numerous unnecessary disk-reads.
> Anyway, what is the most efficient way to get a lookup table into an
> IDL function?
>
> My thoughts so far:
> 1) Just print the look-up table to the screen, copy/paste into the
> function and add some brackets and commas, presto, variable is in the
> function and will stay loaded as long as the function is compiled.
> This will work for my present problem, but would be unwieldy for
> really large look-up tables and I worry about double-precision getting
> truncated on the print.
> 2) have a procedure that reads in the table and puts it in a common
> block, then just start my function with an if-statement to see if the
> common block exists and if not, call the reading procedure. My
> question here is, how can I check to see if a common block has already
> been created? I know I could call the common block maker outside the
> loop, but that seems lame and makes it more complicated if I want to
> share the code.
>
> Unless someone comes up with something really witty, I'll just use
> option 1. Just seems like this would be a common problem that
> someone's solved before.
>
> cheers,
> IP Freeley
