Subject: Re: Vectorizing Code
Posted by Jeff Guerber on Thu, 28 Feb 2002 00:00:15 GMT
View Forum Message <> Reply to Message

On Wed, 27 Feb 2002, Steve Jones wrote:

> Is it possible to vectorize a simple double for-loop?
>
> for i=0,nstate-1 do begin
> for j=0,nvec-1 do begin
> sa(i,j)=10.^2*exp(-abs(i-j)*dz/h)
> endfor
> endfor
>
> I tend to write a large number of such loops and my indexes have been
> steadily increasing of late... Is there a faster alternative? Thanks
> in advance

I'm not sure about completely vectorizing this particular case, but two things that should help:

1) Exchange your i and j loops, since in IDL as in Fortran the leftmost index varies fastest. (See the current thread titled "IDL2MATLAB".)
2) Move the invariant terms outside the loops:

```
d=lonarr(nstate,nvec)
for j=0,nvec-1 do begin
  for i=0,nstate-1 do begin
    d(i,j) = i - j
  endfor
endfor
sa = 10.^2*exp(-abs(d)*dz/h)
```

(Hmmm. You may be able to create (using indgen and replicate) two appropriate nstate-by-nvec arrays, one for i and one for j, then subtract those... In this case, I'm not sure that would be faster, though.)

By the way, ^ has higher precedence than *, so I think you may want to say 10.^(2*exp(...)) [note the extra set of parentheses] instead. What you wrote is equivalent to 100.*exp(...).

Another tip: If your indices might exceed 32767, be sure to write your loops "for i=0L, ...", otherwise the index will be an int instead of a long. That one's bitten me, and it's painful!

Jeff Guerber

Subject: Re: Vectorizing Code
Posted by Kenneth P. Bowman on Thu, 28 Feb 2002 02:56:32 GMT
View Forum Message <> Reply to Message

As I was just reading the infamous REBIN tutorial today, I thought I
would take a shot. I *think* this does what you want.

```
sa = 10.^2*EXP(-ABS(REBIN(LINDGEN(nstate), nstate, nvec, /SAMPLE) - $
  REBIN(REFORM(LINDGEN(nvec), 1, nvec), nstate, nvec, /SAMPLE))*dz/h)
```

Ken

Subject: Re: Vectorizing Code
Posted by Pavel Romashkin on Thu, 28 Feb 2002 05:43:11 GMT
View Forum Message <> Reply to Message

From: "Kenneth P. Bowman" <kpb@null.com>

```
>  sa = 10.^2*EXP(-ABS(REBIN(LINDGEN(nstate), nstate, nvec, /SAMPLE) - $
>     REBIN(REFORM(LINDGEN(nvec), 1, nvec), nstate, nvec, /SAMPLE))*dz/h)
```

I am afraid I'd rather stick with the optimized loops than something that
takes me five minutes to figure out :-(
Pavel

Subject: Re: Vectorizing Code
Posted by marc schellens[1] on Thu, 28 Feb 2002 06:17:51 GMT
View Forum Message <> Reply to Message

```
>>  sa = 10.^2*EXP(-ABS(REBIN(LINDGEN(nstate), nstate, nvec, /SAMPLE) - $
>>     REBIN(REFORM(LINDGEN(nvec), 1, nvec), nstate, nvec, /SAMPLE))*dz/h)
>
>  I am afraid I'd rather stick with the optimized loops than something that
>  takes me five minutes to figure out :-(
>  Pavel
```

but unfortunately in IDL it may save you more than five
minutes at run-time if you do figure it out...
:-) marc

Subject: Re: Vectorizing Code
Posted by K. Bowman on Thu, 28 Feb 2002 15:43:53 GMT
View Forum Message <> Reply to Message

In article <a5kg1a$i7m$1@mwrns.noaa.gov>,
 "Pavel Romashkin" <pavel_romashkin@hotmail.com> wrote:

> From: "Kenneth P. Bowman" <kpb@null.com>
>
>> sa = 10.^2*EXP(-ABS(REBIN(LINDGEN(nstate), nstate, nvec, /SAMPLE) - $
>> REBIN(REFORM(LINDGEN(nvec), 1, nvec), nstate, nvec, /SAMPLE))*dz/h)
>
> I am afraid I'd rather stick with the optimized loops than something that
> takes me five minutes to figure out :-(
> Pavel

I used to feel as you, lost among the REBINs and HISTOGRAMS with
REVERSE_INDICES, but I have read the REBIN tutorial,
brothers and sisters, and I have seen the light!
(Admittedly, it was a 40 W appliance bulb.) It was understanding
the REFORM call that makes this a useful approach for me.

I was recently computing some conditional probability distributions,
and using REVERSE_INDICES sped things up by about a factor of 10
over repeated WHERE's.

To paraphrase Scott Turow, I'm just "learning to love IDL".

Ken

Subject: Re: Vectorizing Code
Posted by James Tappin on Thu, 28 Feb 2002 16:35:49 GMT
View Forum Message <> Reply to Message

Marc Schellens wrote:

>>> sa = 10.^2*EXP(-ABS(REBIN(LINDGEN(nstate), nstate, nvec, /SAMPLE) - $
>>> REBIN(REFORM(LINDGEN(nvec), 1, nvec), nstate, nvec, /SAMPLE))*dz/h)
>>
>> I am afraid I'd rather stick with the optimized loops than something that
>> takes me five minutes to figure out :-(
>> Pavel
>
> but unfortunately in IDL it may save you more than five
> minutes at run-time if you do figure it out...
> :-) marc
>

Loops aren't as bad as they used to be. Many years ago (IDL 2.x on VMS) I
wrote a very clever (IMHO) routine for time-averaging irregularly sampled
data which speeded things up about 10-fold. However recently (now with IDL

5.x on Linux and Solaris) it became clear that this same time averaging routine was a major bottleneck. Replacing it with a version with one more loop and many less temporary arrays speeded up 100-fold in some cases.

The moral: Nowadays loops may be slow but creating and destroying workspace arrays is often even slower.

James

--
 +----------------------+-----------------------------+-- -------+
James Tappin	School of Physics & Astronomy	O__
sjt@star.sr.bham.ac.uk	University of Birmingham	-- \/`
Ph: 0121-414-6462. Fax: 0121-414-3722	
 +---+-- -------+