Subject: Re: Matrix filling methods?
Posted by sjt on Fri, 19 Aug 1994 10:33:38 GMT
View Forum Message <> Reply to Message

Andy Nicholas (nicholas@dsuap1) wrote:
:
: I'm trying to speed up some code and hence I am trying
: to get rid of some FOR loops. I need to fill a 2-d square
: matrix. The values above the diagonal are given by one
: formula while those values below the diagonal are given
: by another. this is how i currently fill the matrix:

```
:           for i=0,n-1 do begin
:               Matrix(i:*,i-1) = x(i-1)/x(k:*)
:               Matrix(i-1,i:*) = y(k:*)/y(k-1)
:           endfor
:           diag=findgen(n)
:           Matrix(diag,diag)=z(diag)
```

: Does anyone know of a way to speed this up? Maybe a where to find the
: matrix elements above the diagonal and one for below?
: Any help is greatly appreciated,
: Thanks,
: Andy
: nicholas.uap.nrl.navy.mil
:

This should be no problem provied you have enough memory for several
arrays of size n. Here is the way I'd do it:

```
l = lindgen(n,n)
lc = l mod n
lr = l / n

upper = lc gt lr
lower = lc lt lr

Matrix = fltarr(n,n)
matrix(upper) = ...
matrix(lower) = ...
```

If you need to include the diagonal then just replace gt or lt with ge or le

--
James Tappin, School of Physics & Space Research
University of Birmingham
sjt@xun8.sr.bham.ac.uk

"If all else fails--read the instructions!"

O__
-- \/`

Subject: Re: Matrix filling methods?
Posted by steinhh on Fri, 19 Aug 1994 14:26:41 GMT
View Forum Message <> Reply to Message

In article <3321q2$p23@sun4.bham.ac.uk>, sjt@xun8.sr.bham.ac.uk (James Tappin) writes:
|> Andy Nicholas (nicholas@dsuap1) wrote:
[..snip..]
|> : for i=0,n-1 do begin
|> : Matrix(i:*,i-1) = x(i-1)/x(k:*)
|> : Matrix(i-1,i:*) = y(k:*)/y(k-1)
|> : endfor
|> : diag=findgen(n)
|> : Matrix(diag,diag)=z(diag)
|>
|> : Does anyone know of a way to speed this up? Maybe a where to find the
|> : matrix elements above the diagonal and one for below?
|> : Any help is greatly appreciated,
|> : Thanks,
|> : Andy
|> : nicholas.uap.nrl.navy.mil
|> :
|>
|> This should be no problem provied you have enough memory for several
|> arrays of size n. Here is the way I'd do it:
|>
|> l = lindgen(n,n)
|> lc = l mod n
|> lr = l / n
|>
|> upper = lc gt lr
|> lower = lc lt lr
|>
|> Matrix = fltarr(n,n)
|> matrix(upper) = ...
|> matrix(lower) = ...
|>
|> If you need to include the diagonal then just replace gt or lt with ge or le
[..snip..]

Ahm, wouldn't the correct use of upper and lower be:

matrix = <upper-expression>*upper + <lower-expression>*lower

The lc gt lr instruction yields a matrix with zeros and ones, so
I wouldn't use it as index without also using where() around it..

Anyway, I think the multiplication method is faster -- IDL is not
very bright when it comes to optimizing memory shuffling when you
use array indexes as in matrix(where(upper)) = ...
This is performed by a relatively complex loop inside IDL, calculating
the destination address for each element, instead of swoshing the
whole thing at once.

Stein Vidar
