
Subject: New and better!! (Was: Re: Timing results - matrix multiply vs. indexing)

Posted by [steinhh](#) on Fri, 01 Jul 1994 10:48:34 GMT

[View Forum Message](#) <> [Reply to Message](#)

I'm not sure if this actually gets out to the rest of the world, as I did suggest this method during the general discussion two weeks ago, so could somebody outside, say, Scandinavia, send me a mail to confirm that they are seeing this? Bill, do you read me?

Anyway, the method I suggested was to use reform and then rebin with the /sample keyword to replicate the data along the new dimension.

For the original problem:

```
> small_array = reform(small_array,1,8)
> newarray = rebin(cos(small_array),300,8,/sample) * sin(large_array)
```

Or, in the test program supplied from David Landers, it would read:

```
> arr2 = rebin(reform(arr1,1,Size2,/overwrite),Size1,Size2,/sample)
> arr1 = reform(arr1,Size2,/overwrite)
```

I simply inserted the above two lines in the loop where the previous lindgen timing was done, and, surprise! On a DECstation 5000/240 it comes out as: (The BYTE(i)/INT(i)/LONG(i)/FLOAT(i)/DOUBLE(i) lines now refer to the REBIN(REFORM(..)) method, also marked with a *)

Converting Array(135) to Array(213,135) (107 reps)

BYTE(i)	0.7 *	
BYTE(*) =	1B # BYTE	18.3
BYTE(*) =	1.0 # BYTE	17.6
LONG =	1B # BYTE	4.1
FLOAT =	1.0 # BYTE	2.7
INT(i)	0.7 *	
INT(*) =	1 # INT	19.8
INT(*) =	1.0 # INT	18.7
LONG =	1 # INT	4.3
FLOAT =	1.0 # INT	2.6
LONG(i)	0.7 *	
LONG(*) =	1L # LONG	19.8
LONG(*) =	1.0 # LONG	25.5
LONG =	1L # LONG	4.2
FLOAT =	1.0 # LONG	2.6
FLOAT(i)	0.7 *	
FLOAT(*) =	1.0 # FLOAT	18.6
FLOAT(*) =	1.0 # FLOAT	23.6
FLOAT =	1.0 # FLOAT	2.5
FLOAT =	1.0 # FLOAT	2.5

```

DOUBLE(i)          0.9 *
DOUBLE(*) =      1.0D # DOUBLE  20.4
DOUBLE(*) =      1.0 # DOUBLE  19.9
  DOUBLE =      1.0D # DOUBLE  4.1
  DOUBLE =      1.0 # DOUBLE  3.9
% REBIN: Complex expression not allowed in this context: ARR1.
% Execution halted at TEST_ARR </mn/leda/u1/steinhh/IDL/UTIL/idltiming.pro(
35)> (REBIN).
%   Called from $MAIN$ .

```

In other words, it's from 3.5 to 4.3 times faster than using replicate/#.
 The drawback, of course, is that REBIN doesn't take complex values.
 Circumnavigation by doing:

```

arr1 = reform(arr1,1,Size2)
arr2 = complex(rebin(float(arr1),Size1,Size2,/sample),$
  rebin(imaginary(arr1),Size1,Size2,/sample))
arr1 = reform(arr1,Size2,/overwrite)

```

which gives:

```

COMPLEX(i)          10.5
COMPLEX(*) = (1.0, 0.0) # COMPLEX  20.0
COMPLEX(*) =      1.0 # COMPLEX  19.9
  COMPLEX = (1.0, 0.0) # COMPLEX  4.4
  COMPLEX =      1.0 # COMPLEX  4.4

```

Mind you, on a DEC Alpha it looks like:

Converting Array(135) to Array(213,135) (107 reps)

```

BYTE(i)          0.3 *
BYTE(*) =      1B # BYTE  4.0
BYTE(*) =      1.0 # BYTE  4.1
  LONG =      1B # BYTE  0.7
  FLOAT =      1.0 # BYTE  0.5
INT(i)          0.3 *
INT(*) =      1 # INT  4.4
INT(*) =      1.0 # INT  4.4
  LONG =      1 # INT  0.8
  FLOAT =      1.0 # INT  0.5
LONG(i)          0.3 *
LONG(*) =      1L # LONG  3.7
LONG(*) =      1.0 # LONG  4.3
  LONG =      1L # LONG  0.6
  FLOAT =      1.0 # LONG  0.4
FLOAT(i)          0.1 *
FLOAT(*) =      1.0 # FLOAT  3.9

```

```

FLOAT(*) =      1.0 # FLOAT   4.2
  FLOAT =      1.0 # FLOAT   0.4
  FLOAT =      1.0 # FLOAT   0.4
DOUBLE(i)              0.4 *
DOUBLE(*) =      1.0D # DOUBLE 4.0
DOUBLE(*) =      1.0 # DOUBLE 4.1
  DOUBLE =      1.0D # DOUBLE 0.6
  DOUBLE =      1.0 # DOUBLE 0.4
COMPLEX(i)              0.6 *With fix
COMPLEX(*) = (1.0, 0.0) # COMPLEX 3.6
COMPLEX(*) =      1.0 # COMPLEX 4.5
  COMPLEX = (1.0, 0.0) # COMPLEX 0.6
  COMPLEX =      1.0 # COMPLEX 0.6

```

It's a pity that REBIN doesn't take complex values, at least combined with the /sample keyword. Something for future versions, RSI? Anyhow, isn't there a way to fool IDL into looking (temporarily) at a COMPLEX as if it were a DOUBLE? This would make the REBIN(REFORM(..)) method better on all platforms/cases.

This method should also work on 2D -> 3D data, whereas the matrix multiplication method only works for 1D -> 2D data. Unless a general fix for rebin(/sample) comes from IDL, the indexed method is the only way for strings, however...

Stein Vidar Haugan
