Subject: Re: Swap even/odd elements in array
Posted by JD Smith on Tue, 28 Jun 2005 19:00:49 GMT
View Forum Message <> Reply to Message

On Wed, 22 Jun 2005 10:52:04 +0200, Xavier Llobet wrote:

> In article <1119428174.330105.170110@o13g2000cwo.googlegroups.com>,
> photosalex@freenetname.co.uk wrote:
>
>> Hi All,
>> sorry if the question is trivial:
>> how could I swap even and odd elements of a 1-D 16-bit INT array
>> without using loops? That is, if the source array is
>>
>> [a0,a1,a2,a3,..]
>>
>> I want it to be
>>
>> [a1,a0,a3,a2,..]
>>
>> Thanks!
>
> n = (size(a))(1)
> ind=2*indgen(n)
> b=reform(transpose(reform([a(ind+1),a(ind)],n/2,2)),n)

A similar approach using the new stride indexing operator would be:

b=reform(transpose([[a[1:*:2]],[a[0:*:2]]]),n_elements(a))

The problem with all of these dimensional-juggling approaches is that they fail for vectors of odd length (or, more generally, require the length be a strict multiple of p, where p is the transposition length).

Here's a method which doesn't have this requirement:

```
ind=lindgen(n_elements(a))/2*2 & ind[0:*:2]+=1
b=a[ind]
```

It can be generalized for any p like this:

```
ind=lindgen(n_elements(a))/p*p
for i=0,p-2 do ind[i:*:p]+=p-1-i
b=a[ind]
```

It does require the vector to have at least p-1 elements.

JD