Subject: Re: Inverse transforming a product of FFTs
Posted by K. Bowman on Thu, 01 Jul 2004 20:44:12 GMT
View Forum Message <> Reply to Message

In article <52822646.0407010822.1dc890e@posting.google.com>,
 olde_english33@hotmail.com (Eric) wrote:

> I have created two arrays of different time series. One is uniformly
> random, call it rantime, and the other one is based off of a recorded
> set of data values, call it rectime. Consider the following code,
> which I have implemented.
>
> specrantime=fft(rantime)
> specrectime=fft(rectime)
> prod=specrantime*specrectime
> yt=fft(prod, 1)
>
> Now, according to the help files, when you take the forward transform,
> the sum is multiplied by 1/N, which in my case is 31. My question is
> does taking the back transform of a product of two FFTs, both with 31
> variables, lose a factor of 31 when taking the inverse transform?
> That is, after I have computed the inverse transform, do I still need
> to multiply by a factor of 31 to get the right data out?

The inverse FFT of the product of the FFTs of f1 and f2 is equal to the
convolution of f1 and f2.

Try an example and see:

```
;Compute product of FFT's
n = 16
x = 2.0*COS(2.0*!PI*FINDGEN(n)/n)
z = FFT(FFT(x)*FFT(x), /INVERSE)
PRINT, x
PRINT, FLOAT(z)

;Compute convolution
conv = FLTARR(n)
dx = 1.0/n
FOR i = 0, n-1 DO conv[i] = TOTAL(x*SHIFT(x,i)*dx)
PRINT, conv
```

```
IDL> @fft_prod
    2.00000     1.84776     1.41421    0.765367 -8.74228e-08
-0.765367    -1.41421
   -1.84776    -2.00000    -1.84776    -1.41421   -0.765366
2.38498e-08    0.765367
```

```
   1.41421     1.84776

   2.00000     1.84776     1.41421     0.765366 -3.34942e-07
-0.765367    -1.41421
  -1.84776    -2.00000    -1.84776    -1.41421    -0.765366
3.34942e-07   0.765367
   1.41421     1.84776

   2.00000     1.84776     1.41421     0.765367 2.98023e-08
-0.765367    -1.41421
  -1.84776    -2.00000    -1.84776    -1.41421    -0.765367
7.45058e-09   0.765367
   1.41421     1.84776
```

Since 31 is a prime, the FFT turns into a DFT, and you may be better off computing the convolution directly.

Ken Bowman