Subject: Re: Correction: 2D FFt
Posted by peter on Tue, 22 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Walid Atia (atia@wam.umd.edu) wrote:
: Hi,

: I just realized that FFT(data,-1) takes the FFT of an array of up to 7
: dimensions. What confused me was that when I plotted, say, the FFT of a
: gaussian, I got values only near the sides. However, the FFT of a
: gaussian is a gaussian, so I thought that the FFT routine must not be
: working correctly for my array. Just in case anyone out there is
: interested, the problem lies in the way IDL stores the FFT data. The
: edges are taken as 0 frequency, rather than the more intuitively obvious
: center to be zero. A simple shift in the FFT'd image solves the
: problem, and yields the expected results. The code which illustrates
: this is:

<snip>

: Does anyone know of a more direct (and elegant) way of performing this
: transform? And why does IDL store the 2D transform this way--doesn't
: the usual 2D transform treat the center of the array as zero frequency,
: so as to get a symmetrical function given a symmetrical image?

The customary way to perform FFTs always places DC in element 0, not in
the middle. IDL follows this convention. You'll confuse more than a
few people if you adopt any other rule, so please don't!

Peter

Subject: Re: Correction: 2D FFt
Posted by thompson on Tue, 22 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Walid Atia <atia@wam.umd.edu> writes:

> ... And why does IDL store the 2D transform this way--doesn't
> the usual 2D transform treat the center of the array as zero frequency,
> so as to get a symmetrical function given a symmetrical image?

Actually, every Fortran FFT I've ever worked with stores the data in this way.
It makes a certain amount of sense, in that the power at zero frequency is at
(0,0), just like the 1D power at zero frequency is at (0). But it does take a
bit of getting used to.

A lot of times, one is only interested in the power, and only needs the lower left quadrant of the FFT, i.e. F(0:NX,0:NY), where NX and NY are the Nyquist frequencies in the two dimensions.

Note that the IDL DIST() function provides the radius vector in frequency space, for forming filters that are symmetric, i.e. depend only on the spatial frequency.

Bill Thompson

Subject: Re: Correction: 2D FFt
Posted by agraps on Tue, 22 Oct 1996 07:00:00 GMT

View Forum Message <> Reply to Message

Walid Atia <atia@wam.umd.edu> writes:

> I just realized that FFT(data,-1) takes the FFT of an array of up to 7
> dimensions. What confused me was that when I plotted, say, the FFT of a
> gaussian, I got values only near the sides. However, the FFT of a
> gaussian is a gaussian, so I thought that the FFT routine must not be
> working correctly for my array. Just in case anyone out there is
> interested, the problem lies in the way IDL stores the FFT data. The
> edges are taken as 0 frequency, rather than the more intuitively obvious
> center to be zero.

To be fair, I don't think that IDL is the only software package that performs the ordering that way. I believe that many others order the output into quadrants as well. One should always check the output of the FFT first and then reorder it (or the filter) if necessary before applying any kind of filter.

Well I learned this little fact too, about a year ago. I have some illustrations showing some high-pass filtering of solar images with the swapped gaussian filter I used to remove some noisy components of my images. (All done in IDL). The results were nice. See

http://quake.stanford.edu/~amara/movdev.html

(Warning: that page has about 500 kB of images)

Amara

--

```
 ************************************************************ *************
Amara Graps                     email: agraps@netcom.com
Computational Physics               vita:  finger agraps@best.com
Multiplex Answers                 URL:   http://www.amara.com/
```

*** *************

Subject: Re: Correction: 2D FFt
Posted by Graeme K Harkness on Wed, 23 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Walid Atia wrote:

```
>
>  Hi,
>
>  I just realized that FFT(data,-1) takes the FFT of an array of up to 7
>  dimensions.  What confused me was that when I plotted, say, the FFT of a
>  gaussian, I got values only near the sides.  However, the FFT of a
>  gaussian is a gaussian, so I thought that the FFT routine must not be
>  working correctly for my array.  Just in case anyone out there is
>  interested, the problem lies in the way IDL stores the FFT data.  The
>  edges are taken as 0 frequency, rather than the more intuitively obvious
>  center to be zero.  A simple shift in the FFT'd image solves the
>  problem, and yields the expected results.  The code which illustrates
>  this is:
>
>  x=shift(dist(201),100,100) 'create an array of r-values
>  z=exp(-(x/10)^2) 'gaussian with spot size (1/e) of 10.
>  shade_surf,z 'plot the gaussian
>  Fz=FFT(z,-1) 'take the FFT of z--note that Fz is now of complex type
>  shade_surf,abs(Fz) 'this will give the wrong picture!
>  Fzfix=shift(Fz,100,100) 'correct the transform so that the center of the
>                     image is at zero frequency
>  shade_surf,abs(Fzfix) 'the FFT of a gaussian is a gaussian.
>  IFz=FFT(Fz,1) 'take the inverse fft.
>  shade_surf,IFz 'gives the original data, as expected.
>
>  Does anyone know of a more direct (and elegant) way of performing this
>  transform?  And why does IDL store the 2D transform this way--doesn't
>  the usual 2D transform treat the center of the array as zero frequency,
>  so as to get a symmetrical function given a symmetrical image?
>
>  Hope someone besides myself found this useful!
>
>  Walid
```

Walid,

I'm pretty sure that this re-arrangement of the frequency space is
related
to the methods used to do Fast Fourier Transforms in general (but it's
been

a long time since I studied this stuff :-)

I have a couple of routines (FFT1D and FFT2D) which take the Fourier
transforms and return you the re-ordered data (with zero frequency at
the
centre). They do use the standard IDL routines and then re-order
afterwards
so they aren't the most efficient things ever but they work very well.
(I suppose if you wanted very efficient code you wouldn't be using the
FFTs
in IDL anyway since I'd bet you could call an external function in
FORTRAN
(or something) to do it much faster!)

If you'd like a copy of these, then drop me an e-mail and I'll forward
them
on to you. If more people would like them I can post them if you like.

Cheers,

Graeme

....

| Graeme K Harkness |
| Department of Physics & Applied Physics |
| University of Strathclyde |
| GLASGOW G4 0NG UK |
| Tel: +44-141-552 4400 x3354 |
| Fax: +44-141-552 2891 |
| graeme@phys.strath.ac.uk |

Subject: Re: Correction: 2D FFt
Posted by steinhh on Tue, 29 Oct 1996 08:00:00 GMT

View Forum Message <> Reply to Message

In article <326DCEDB.41C6@phys.strath.ac.uk>, Graeme K Harkness
<graeme@phys.strath.ac.uk> writes:
|> Walid,
|>
|> I'm pretty sure that this re-arrangement of the frequency space is
|> related
|> to the methods used to do Fast Fourier Transforms in general (but it's
|> been
|> a long time since I studied this stuff :-)
|>

Well, sort of, but it also makes sense to have e.g., the zero frequency
at (0,0) as others have pointed out..

|> I have a couple of routines (FFT1D and FFT2D) which take the Fourier
|> transforms and return you the re-ordered data (with zero frequency at
|> the
|> centre). They do use the standard IDL routines and then re-order
|> afterwards
|> so they aren't the most efficient things ever but they work very well.
|> (I suppose if you wanted very efficient code you wouldn't be using the
|> FFTs
|> in IDL anyway since I'd bet you could call an external function in
|> FORTRAN
|> (or something) to do it much faster!)
|>

I wouldn't recommend spending time on trying to beat IDL's array operations, especially stuff like the FFT functions! They're quite well optimized.

Once when I had to compute a lot of auto-correlation functions, I tried to use Numerical Recipies to beat IDL, taking advantage of the fact that my data points were real, not complex, etc.., and I was ending up with real data points as well.

Even using every trick in the book, I ended up not saving more than about 5% of the execution time. The effort would have been a waste of time if it hadn't been for the fact that I had to also apply a filter in frequency space, which could be done a lot more efficient when they were done in the program containing the ffts.

Stein Vidar