Subject: Re: FFT and ROTATE
Posted by Kenneth P. Bowman on Thu, 04 Sep 2008 06:11:22 GMT
View Forum Message <> Reply to Message

In article
<0473bd21-6928-45b9-9e59-56a8aa5ab9df@r15g2000prh.googlegroups.com>,
 wheinz@gmail.com wrote:

> Hello,
>
> I have been wrestling with the FFT and ROTATE functions recently. One
> of the properties of the Fourier transform is that the transform of a
> rotated object is equal to the rotation of the transform of the
> unrotated object. To test this in IDL, I took the FFT of an nxn array
> (called image) and the FFT of that array rotated 90 degrees, image90 =
> ROTATE(image,1). Then, I sorted the real and imaginary parts of the
> coefficients of the results of the FFTs and compared the sorted
> values. I expected that the sorted list of real parts from the FFT of
> the original and rotated arrays would be identical, and that the same
> would be true for the imaginary parts. This is not the case. The sets
> of the magnitudes of the coefficients are equal, as expected.
>
> I understand that edge effects can play a role, but when I rotate a
> square array by 90 degrees, I expect that the sets of real and
> imaginary values that define the coefficients to be equal.
>
> Is this a consequence of how the FFT is calculated -- rows first then
> columns or vice versa? Or is there something else going on? Any help
> or suggestions will be greatly appreciated.

Are the differences of the order of the machine precision?

Ken Bowman

Subject: Re: FFT and ROTATE
Posted by Wox on Thu, 04 Sep 2008 11:55:27 GMT
View Forum Message <> Reply to Message

On Wed, 3 Sep 2008 15:20:19 -0700 (PDT), wheinz@gmail.com wrote:

> Hello,
>
> I have been wrestling with the FFT and ROTATE functions recently. One
> of the properties of the Fourier transform is that the transform of a
> rotated object is equal to the rotation of the transform of the
> unrotated object. To test this in IDL, I took the FFT of an nxn array
> (called image) and the FFT of that array rotated 90 degrees, image90 =

> ROTATE(image,1). Then, I sorted the real and imaginary parts of the
> coefficients of the results of the FFTs and compared the sorted
> values. I expected that the sorted list of real parts from the FFT of
> the original and rotated arrays would be identical, and that the same
> would be true for the imaginary parts. This is not the case. The sets
> of the magnitudes of the coefficients are equal, as expected.

First of all, care must be taken when rotating the fourier transform. Check IDL help on this: you have to shift with half the image size in both directions. I also noticed that it only works with uneven image sizes (i.e. rotation around the center pixel). Anyway, check the code below:

```
pro rotFFTtest
  ;;load an image
  fn = filepath('md1107g8a.jpg',SUBDIRECTORY='examples/data')
  image=bytarr(251,251)
  image[0,0]= read_image(fn)
  image90 = rotate(image,1)

  ;display the images
  window,0
  tvscl,image,0
  tvscl,image90,1

  n = size(image,/dim)
  nfreq=n/2+1 ; # positive freq in each dim
  nfreq_m=nfreq-1-(~(n mod 2)) ; # negative fequencies in each dim

  ;;take fft of image, then get the real and imaginary parts
  f = fft(image)
  f = shift(f,-nfreq[0],-nfreq[1])
  f90_1 = rotate(f,1)
  f90_1 = shift(f90_1,nfreq[0],nfreq[1])

  ;;take the fft of image90 then get the real and imaginary parts.
  f90_2 = fft(image90)

  tvscl,fft(f90_1,/inverse),2
  tvscl,fft(f90_2,/inverse),3
end
```

Subject: Re: FFT and ROTATE

On Sep 3, 5:20 pm, whe...@gmail.com wrote:
> Hello,
>
> I have been wrestling with the FFT and ROTATE functions recently. One
> of the properties of the Fourier transform is that the transform of a
> rotated object is equal to the rotation of the transform of the
> unrotated object. To test this in IDL, I took the FFT of an nxn array
> (called image) and the FFT of that array rotated 90 degrees, image90 =
> ROTATE(image,1). Then, I sorted the real and imaginary parts of the
> coefficients of the results of the FFTs and compared the sorted
> values. I expected that the sorted list of real parts from the FFT of
> the original and rotated arrays would be identical, and that the same
> would be true for the imaginary parts. This is not the case. The sets
> of the magnitudes of the coefficients are equal, as expected.
>
> I understand that edge effects can play a role, but when I rotate a
> square array by 90 degrees, I expect that the sets of real and
> imaginary values that define the coefficients to be equal.
>
> Is this a consequence of how the FFT is calculated -- rows first then
> columns or vice versa? Or is there something else going on? Any help
> or suggestions will be greatly appreciated.
>
> Here is some code I am using to try to wrap my head around this.
>
> Thanks,
> Will
>
> pro rotFFTtest
> ;;load an image
> fn = filepath('md1107g8a.jpg',SUBDIRECTORY='examples/data')
> image= read_image(fn)
> image90 = rotate(image,1)
>
> ;display the images
> tvscl,image,0
> tvscl,image90,1
>
> n = n_elements(image)
>
> ;;take fft of image, then get the real and imaginary parts
> f = fft(image)
> fr = real_part(f)
> fi = imaginary(f)
>
> ;;take the fft of image90 then get the real and imaginary parts.

```
>    f90 = fft(image90)
>    fr90 = real_part(f90)
>    fi90 = imaginary(f90)
>
>    ;;sort and print the real and imaginary parts
>    frs = sort(fr)
>    fr90s = sort(fr90)
>    fis = sort(fi)
>    fi90s = sort(fi90)
>
>    ;;print some of the sorted coefficients
>    print,'fr[frs[[1:4]]',fr[frs[1:4]]
>    print,'fr90[fr90s[[1:4]]',fr90[fr90s[1:4]]
>
>  ;print all of the sorted coefficients
>  ;print,'Real','Real rotated','Imag.','Imag. rotated',FORMAT='(4A17)'
>  ;for i = 0L,n-1L do
>   print,fr[frs[i]],fr90[fr90s[i]],fi[fis[i]],fi90[fi90s[i]],FO RMAT='(4G17.13)'
>
>  end
```

This might help, too: http://cimss.ssec.wisc.edu/~paulv/fft/fft_comparison.html

---

Subject: Re: FFT and ROTATE
Posted by wheinz on Fri, 05 Sep 2008 21:04:11 GMT
View Forum Message <> Reply to Message

Thanks for the suggestions. I am still confused by a few things, but with Wox's code I think I can clarify my question. The problem is that although Wox's code shows that the FFT of the rotated image and the rotated FFT of the original image give you back images that look like the original when inverse -transfored, the values of the coefficients in those two FFTs are not the same. I added some print statements to Wox's code to show what is confusing me.

```
pro rotFFTtest3
  ;;load an image
  fn = filepath('md1107g8a.jpg',SUBDIRECTORY='examples/data')
  image=bytarr(251,251)
  image90=bytarr(251,251)
  image[0,0]= read_image(fn)
  image90[0,0] = rotate(read_image(fn),1)

  ;display the images
  window,0
  tvscl,image,0
  tvscl,image90,1
```

```
  n = size(image,/dim)
  nfreq=n/2+1 ; # positive freq in each dim
  nfreq_m=nfreq-1-(~(n mod 2)) ; # negative fequencies in each dim

  ;;take fft of image
  f = fft(image)

  ;;shift it
  f = shift(f,-nfreq[0],-nfreq[1])

  ;;rotate the fft 90 degrees
  f90_1 = rotate(f,1)

  ;;shift it back
  f90_1 = shift(f90_1,nfreq[0],nfreq[1])
  ;;take the fft of image90 -- the rotated image
  f90_2 = fft(image90)

;;**********new code to print sorted values of coefficients**********
  ;;get the real parts of the ffts
  fr =real_part(f)
  f90_1r = real_part(f90_1)
  f90_2r = real_part(f90_2)

  ;;now we look at the set of real coefficiencts of each fft
  sf = fr[sort(fr)]
  s1 = f90_1r[sort(f90_1r)]
  s2 = f90_2r[sort(f90_2r)]

  print,'The sorted list of real coefficients.'
  print,'  fft(image):',sf[0:4]
  print,' rotated fft:',s1[0:4]
  print,'fft(image90):',s2[0:4]

  ;;**********end new code to print sorted values of
coefficients**********


  tvscl,fft(f90_1,/inverse),2
  tvscl,fft(f90_2,/inverse),5

end
```

The program prints the following:

```
IDL> rotFFTtest2
The sorted list of real coefficients.
  fft(image):     -36.3499     -36.3499     -26.5806     -26.5806
```

-5.83106
 rotated fft: -36.3499   -36.3499   -26.5806   -26.5806
-5.83106
fft(image90):   -36.3499   -36.3499   -26.6964   -26.6964
-5.65933

So, you can see that the values differ, and (to answer ken's question) the differences in the values are greater than the machine's precision. Here I only printed the real parts, but the same thing happens with the imaginary parts.

The fact that the inverse transformations return the original image suggests that there is a phase shift introduced somewhere in the transform. I can understand that if the array has an even number of elements in x or y, then a rotation forces a translation of the pixels. But an array with an odd number of elements in x and y should not, especially if we are only looking at 90 degree rotations.

Why aren't the sorted lists of the real parts of the coefficients from the rotated fft and the fft of the rotated image equal?

The article Vince linked to states that different implementations of the FFT use different indexing and normalization approaches. Is it possible that these differences could come from whatever scheme IDL uses for its FFT?

Thanks,
Will

Subject: Re: FFT and ROTATE
Posted by Wox on Sat, 06 Sep 2008 11:47:20 GMT
View Forum Message <> Reply to Message

On Fri, 5 Sep 2008 14:04:11 -0700 (PDT), wheinz@gmail.com wrote:

> The fact that the inverse transformations return the original image
> suggests that there is a phase shift introduced somewhere in the
> transform. I can understand that if the array has an even number of
> elements in x or y, then a rotation forces a translation of the
> pixels. But an array with an odd number of elements in x and y should
> not, especially if we are only looking at 90 degree rotations.
>
> Why aren't the sorted lists of the real parts of the coefficients from
> the rotated fft and the fft of the rotated image equal?

Maybe it has something to do with this (see rotation and edge effects):

http://www.cs.unm.edu/~brayer/vision/fourier.html
There you see that the modulus is also different when you compare mod(fft(rot(image))) with mod(rot(fft(image))). Apparently the fact that the moduli were the same in your case, was because you rotated 90 degrees. See below for rotating 45 degrees. Someone with a better knowledge of fourier transformation can probably explain better. So I don't think it's a rounding problem, but a problem that comes with images having a finite size.

```
pro rotFFTtest3
  ;;make image
   image=rebin(bytscl(cos(!pi/18.*findgen(355))),355,355,/sampl e)
  image90 = rot(image,-45)
  i0=52
  i1=i0+250
  image=image[i0:i1,i0:i1]
  image90=image90[i0:i1,i0:i1]

  ;display the images
  window,0
  tvscl,image,0
  tvscl,image90,1
  n = size(image,/dim)
  nfreq=n/2+1 ; # positive freq in each dim
  nfreq_m=nfreq-1-(~(n mod 2)) ; # negative fequencies in each dim

  ;;take fft of image
  f = fft(image)

  ;;shift it
  f = shift(f,-nfreq[0],-nfreq[1])

  ;;rotate the fft 90 degrees
  f90_1 = rot(f,45)

  ;;shift it back
  f90_1 = shift(f90_1,nfreq[0],nfreq[1])

  ;;take the fft of image90 -- the rotated image
  f90_2 = fft(image90)

  tvscl,fft(f90_1,/inverse),2; fft(rot(fft(image)))
  tvscl,fft(f90_2,/inverse),4; fft(fft(rot(image)))

  ;;phase shift between fft(rot(image)) and rot(fft(image))
  f90_1phase=atan(f90_1,/phase)
  f90_2phase=atan(f90_2,/phase)
```

```
  phshift=abs(f90_1phase-f90_2phase)*180/!pi
  f90_1mod=abs(f90_1)
  f90_2mod=abs(f90_2)
  moddiff=abs(f90_1mod-f90_2mod)

  window,1
  tvscl,shift(moddiff,-nfreq[0],-nfreq[1])<0.1,1
  tvscl,shift(phshift,-nfreq[0],-nfreq[1]),0
  print,'fft(rot(image)) vs. rot(fft(image))'
  print,'Phase shift range (degrees): ',min(phshift),max(phshift)
  print,'Modulus diff range: ',min(moddiff),max(moddiff)
end
```

Subject: Re: FFT and ROTATE
Posted by Kenneth P. Bowman on Sun, 07 Sep 2008 12:59:58 GMT
View Forum Message <> Reply to Message

In article
<0fc5b913-ae5d-4a1c-afb3-ba1184f174ce@c58g2000hsc.googlegroups.com>,
 wheinz@gmail.com wrote:

> Thanks for the suggestions. I am still confused by a few things, but
> with Wox's code I think I can clarify my question. The problem is that
> although Wox's code shows that the FFT of the rotated image and the
> rotated FFT of the original image give you back images that look like
> the original when inverse -transfored, the values of the coefficients
> in those two FFTs are not the same. I added some print statements to
> Wox's code to show what is confusing me.

I find that the only way to really understand FFTs is to work with a
example where I know the answer exactly. I suggest that you create
a highly simplified image by combining a few harmonic components
in the x- and y-directions, then look at the FFTs with and
without rotation.

For example

```
   f(x,y) = sin(2 * !pi * x)  #  sin(4 * !pi * y)
```

where x = y = FINDGEN(16)/16.0

You can use try different (distinct) sine and cosine components
to investigate how they translate into real and imaginary
parts.

Also, you can look at dimensions sizes that are not powers of 2.