Subject: problems with FFT cross spectra and other floating point operations

Posted by steve on Thu, 17 Dec 1992 03:09:46 GMT

View Forum Message <> Reply to Message

I am doing some image processing and I find that I get results which differ drastically from those in Matlab. I am doing Fourier cross correlation using phase-only filtering of video images. IDL gives floating underflow.

```
==> qf=af*conj(bf)/(abs(af)*abs(bf))
% Program caused arithmetic error: Floating underflow
% Program caused arithmetic error: Floating illegal operand
% Program caused arithmetic error: Zero / Zero
```

where af and bf are the fourier transforms of the two images.
I have C code that works, and Matlab code that works, but I get this error in IDL.

I also noticed that when I subtract the DC from an image, and then do an FFT, the element af(0,0) is not always zero. In fact, as the image gets bigger, I get values further and further from zero. For a 256 by 265 image, the DC spike in the Fourier domain is so high that I can't see anything else in shade_surf,abs(af) or tvscl,abs(af).

I wrote this little script which shows the accumulation of excess DC:

```
 ------------------------------------------------------------ --------------
; when I set N=91 or less, I get zero as expected.
; when I set N=92 or more, I get a non-zero DC value in fft.

N = 100;  size of square array
q=findgen(N)#findgen(N); create some arbitrary square array
;Subtract the DC component
 w=float(q)-norm(q,/one)/float((size(q))(1))/float((size(q))( 2))
wf = fft(w,1);      Fourier transform of w
print,wf(0,0);      The DC component should be zero now.
;I dont know why it is not zero.
 ------------------------------------------------------------ -------------
```

Perhaps this is due to a bug in IMSL/IDL?, and perhaps other versions (e.g. Wave or RSI) don't suffer from this bug???

Could it have something to do with precision of numerical representation, and if so, how do I represent complex double in IDL or Wave?

As a side note, I observe the following problems with simple numerical computations. Have I overlooked something, or are these bugs?

```
==> print,double(10)^(-double(323))
  9.8813129e-324
;this is fine
```

```
==> print,double(10)^(-double(324))
      0.0000000
; no underflow warning is given

==>  print,double(10)^(-double(99999999999999999999999999999999) )
      10.000000
; no warning, and a strange answer results (should be zero)

; note that on the big side, things seem to be working ok:
==> print,double(10)^(double(308))
  1.0000000e+308
==> print,double(10)^(double(309))
           INF

; also when working with complex numbers, there appears to be a serious
; limitation, in the sense that the biggest real and imaginary parts are
; of single precision, yet complex double is the format of choice for
; most numerical work (e.g. Matlab)
==> print,complex(double(10)^(double(38)))
(  1.00000e+38,      0.00000)
==> print,complex(double(10)^(double(39)))
(          INF,      0.00000)
==> print,complex(double(10)^(-double(46)))
(      0.00000,      0.00000)
```

Could this be the reason that IDL answers don't aggree with Matlab, C, or Fortran answers?

Any help would be appreciated.