Subject: Re: FFT accuracy
Posted by ali on Mon, 20 Apr 1992 17:24:39 GMT
View Forum Message <> Reply to Message

In article <9204171736.AA03199@dip.eecs.umich.edu>, pan@ZIP.EECS.UMICH.EDU writes...
> I had two examples that show the problem of numerical accuracy in
> using FFT of either IDL or PVWAVES. Is the FFT function
> provided by either IDL or PVWAVES is suitable for accuracy-demanding
> calculation? Should I go look for a double precision FFT?
> Any comment is appreciated? -tinsu pan
>

For the examples shown, IDL computes the FFTs to the best accuracy attainable with single precision floating point. The differences between IDL and WAVE are simply due to differences in the floating point implementation on the IBM RS-6000 and the DECStation.

The examples would more properly be written:

```
  x = findgen(256,256)
  y = float(fft(fft(x,-1),1))
  err = abs(x-y)
  print, 'Maximum relative error: ', max(err,i) / x(i)
Maximum relative error:   1.83841e-07
  print, 'Total relative error: ', total(err)/total(x)
Total relative error:   4.21436e-08
```

For 512 x 512 the errors are:

```
Maximum relative error:   1.34636e-06
Total relative error:   8.42841e-08
```

These examples, run on a SPARC show the worst case error to be well within the expected IEEE single precision significance of between 6 to 9 decimal digits.

A double-precision FFT would, of course, provide better accuracy, but is not provided because there is no double precision complex data type.

There was, however, a much worse error in the FFT that occured with arrays whose dimensions were larger prime numbers. This error has been fixed in IDL and the revised code will appear in IDL Version 2.3, which will be released shortly.

Here is an excerpt from the release notes of IDL Version 2.3:

```
 The fast Fourier (FFT) function produced unnacceptable
 errors when working on arrays with dimensions that had large
 PRIME factors.  There was no problem with dimensions that
 are a power of two, three, etc., or whose largest prime factor
 is less than those discussed below .

 For example, a forward/inverse transform
 on an array with 1181 elements produced unrecognizable
 results.  The problem diminished with smaller prime factors.
 The problem has been fixed, at the expense of longer
 execution time.  The inaccuracies became apparent when
 the largest prime factor of a dimension was over
 approximately 227.

 Arrays with dimensions whose largest prime factor is less than
 approximately 100 presented no problem.  Execution time
 remains unchanged for this type of arrays.

 Examples: Array with a dimension of:
  260 (2x2x5x13)  no problem
  261 (3x3x29)  no problem
  262 (2x131)  moderate problem
  263 (1x263)  problem
  264 (2x2x2x3x11) no problem
  265 (5x53)  no problem
```

- Research Systems, Inc.

Subject: Re: FFT accuracy
Posted by thompson on Mon, 20 Apr 1992 20:34:00 GMT
View Forum Message <> Reply to Message

In article <1992Apr20.172439.11546@colorado.edu>, ali@anchor.cs.colorado.edu (Ali Bahrami) writes...

> ...A double-precision FFT would, of course, provide better accuracy, but
> is not provided because there is no double precision complex data
> type.

Well, that raises a good question. Why isn't there a double precision complex data type? I've always wondered why that is.

Bill Thompson

Subject: Re: FFT accuracy
Posted by dale on Thu, 23 Apr 1992 13:59:27 GMT
View Forum Message <> Reply to Message

In article <1992Apr20.172439.11546@colorado.edu>, ali@anchor.cs.colorado.edu (Ali Bahrami) writes...

> ...A double-precision FFT would, of course, provide better accuracy, but
> is not provided because there is no double precision complex data
> type.

Since IMSL/IDL has just been released a few of us have been looking at this discussion with interest. Finally Mike Pulverenti came up with the following observations that give a workaround and hope for the future:

IMSL/IDL is developing a scheme to compute double precision
complex FFT's of complex arrays internally, but return
the results in the current complex data type of IMSL/IDL.
If the IMSL/IDL routine FFTCOMP is called with a complex
array, and the DOUBLE keyword is present and nonzero, then
IMSL/IDL will internally promote your data to double complex,
compute the FFT in double precision, and then demote the results
back to single precision complex for return to the user.

If you can't wait for the next release of IMSL/IDL, you
can alternatively compute a double precision complex FFT of
a real array by computing a double precision real FFT, then permute
the result to mimic the results of a complex FFT.
This will, in effect, produce a double precision complex FFT
of real data. The example below demonstrates the pattern of
the permutation needed.

```
==> x = random(5)                    ; Define an array of random numbers.
==> pm, fftcomp(x, /double)            ; Compute the double precision
      3.5419804                      ; real FFT.
     0.17009673
     0.16646779
   -0.045797205
     0.11475440
==> pm, fftcomp(x, /complex)           ; Note that every element of the
(      3.54198,      0.00000)         ; complex FFT appears in the
(     0.170097,     0.166468)          ; double precision FFT.
(   -0.0457972,     0.114754)
(   -0.0457972,    -0.114754)
(     0.170097,    -0.166468)
```

Using a double precision FFT code should certainly help if you
require more accuracy, but the examples given in article
<9204171736.AA03199@dip.eecs.umich.edu>, pan@ZIP.EECS.UMICH.EDU
do not imply poor accuracy of the FFT's.

One method to determine the accuracy of the FFT's is by first examining
the 'best case' scenario of computing the DFT by means of a simple
matrix-vector operation involving an NxN orthogonal matrix F,
where F(k,j) is defined to be exp((2*(PI)*k*j)i/N). (The 'i' in the
last expression denotes the imaginary part of a complex number.)
Computing the DFT by this method is known to be stable numerical process.
In order to determine the numerical stability of the FFT algorithm
you are using, you can compare it's results with the original DFT.

Dale

Have a good day!

Dale L. Neaderhouser dale@imsl3.imsl.com FAX: 713-242-9799
Senior Software Engineer uunet!imsl!dale IMSL: 713-242-6776
Post Sales Technical Support: 800-324-4675 Sales: 800-222-4675