Notes on Performance/Accuracy/Testing

<u>Performance</u>

P2DFFT has been used in several different projects and has undergone significant changes since the initial releases. These releases have focused on adding functions to the code package (like generating reverse FFT images and creating test spirals), usability, and algorithmic improvements. One of the most important areas has been to improve the performance of the code.

Initial single threaded versions of 2DFFT used a serial string of separate instances to generate the results. With the availability of multi threaded systems and the modest memory needs of each process, making the analysis multi-threaded added a significant performance improvement in the code (P2DFFT 3.x and later).

One of the performance challenges was the reliance on the Numerical Recipes fourn.c file to do the FFT transforms. This was responsible for 95%+ of the processing time, and has some limitations on the distribution of the file (even though it is available on the Internet).

As an alternative, the Harvard LIBFFTW3 (Fastest Fourier Transform in the West, version 3) was used starting P2DFFT 4.0+. This library both simplifies the code (because it manages multiple threads instead of relying on external logic) and uses code which is dynamically optimized for the execution environment. The FFTW3 package improved performance considerably and since it is widely available in most Linux distributions, it removed any licensing issues.

P2DFFT5 made further optimizations. First, it removed the need for a FORTRAN 77 package and eliminated the use of temporary I/O files. Calling multiple processes and using files for inter-process communication incurred a substantial performance penalty. The other major change in P2DFFT5 was in multi-threading. Testing showed that the automatic multithreading support built into FFTW3 did not scale well in machines with >4 cores. The solution was to still use the FFTW3 transform libraries, but to use the thread management structure from P2DFFT3.x.

Some sample FITS images were used with different versions of the program to measure the performance. The images are identical and since the FFT computational effort scales with image size, four different size images were used. The results are in Figure 1.

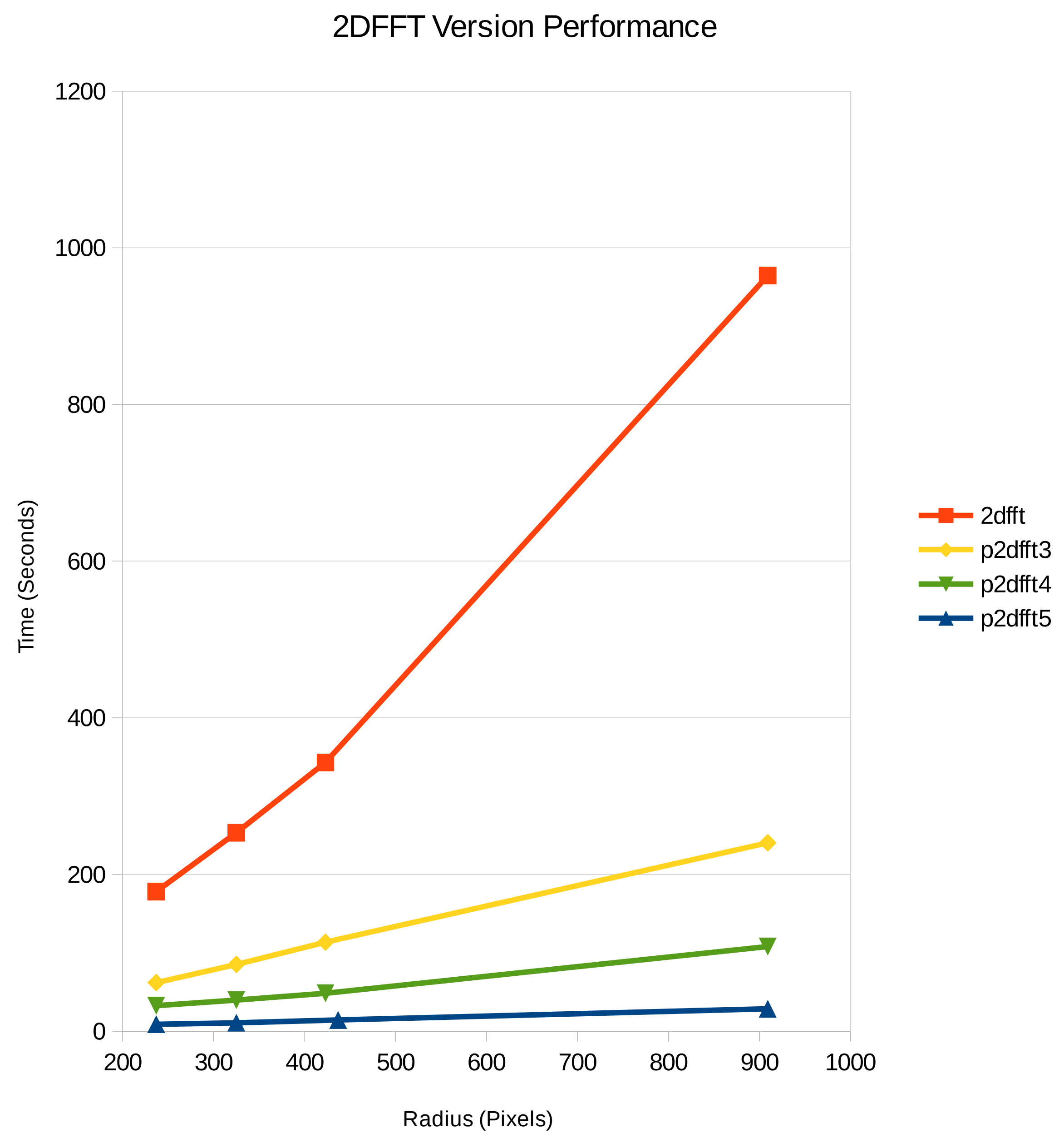

Figure 1. Performance of 2dfft/p2dfft program by number of annuli

The most significant performance improvement is gained for larger radii by using multiple threads. The change to the FFTW library also significantly improved performance (see notes on accuracy below). The detailed performance data is shown in

Table 1. Please note that the times may vary on different machines and the relative difference for any given machine is the indicator of performance.

Program	Size (Pixels)	Time (Secs)
2DFFT	237	178.17
2DFFT	325	253.29
2DFFT	423	343.06
2DFFT	909	964.48
P2DFFT3	237	62.1
P2DFFT3	325	85.36
P2DFFT3	423	113.69
P2DFFT3	909	240.51
P2DFFT4	237	32.68
P2DFFT4	325	39.72
P2DFFT4	423	48.41
P2DFFT4	909	108.17
P2DFFT5	237	8.77
P2DFFT5	325	10.62
P2DFFT5	437	14.37
P2DFFT5	909	28.47

Table 1. Run times for 2dfft versions by number of annuli

Accuracy

Each significant update of p2dfft is tested against results from the original 2DFFT program using the p2dfft-test package. This compares the resulting output from the original 2DFFT for several standard input images with the results from the same images using the latest version. This insures that no major inaccuracies are introduced and previous data is consistent with newer runs. The results show generally good agreement with the following caveats:

1. Using P2DFFT 2.x+ on different architectures will return close agreement but per radius values may be different by:
   - Frequency ± 0.001 Hz
   - Amplitude ± 0.01
   - Pitch Angle ± 0.01 Degrees

    - Phase Angle ± 0.1 Degrees
    - Rip File Values ± 0.001%
    - Dat File Values ± 0.001%
2. For P2DFFT 4.x+ (libfftw), the last 1.5% of radii will return significantly different results from the original 2dfft, but this is considered acceptable since Davis et. al. 2012 documents that the last 10% of the annuli can return results which are less reliable. Other results will differ by more those in case #1 due to some differences in the algorithm implementation and the precision. The FFTW3 results were compared to the original 2DFFT results using the following tolerances:
    - Frequency ± 0.25 Hz
    - Amplitude ± 0.01
    - Pitch Angle ± 0.25 Degrees
    - Phase Angle ± 0.25 Degrees (But some results will fall outside this range due to changes in frequency/amplitude/pitch values)
    - Rip File Values ± 0.1%
    - Dat File Values ± 0.1%

Testing Future Versions

The *p2dfft-test* package (available separately) can be used to test new versions of the package. It is suggested that if you are testing FFTW versions against the Numerical recipes version, it is best to use to the looser values in #2. Please note that a few values will still fall outside of these ranges, but will not be significant. If you are testing like versions (e.g. two FFTW3 versions), it is recommended to use the tighter values in #1. In this case, there should be no discrepancies between versions outside of the limits. However, please also note that the numbers will never be identical between different distributions and architectures, which is the reason for using tolerances in the test package.

References

Davis, B.L., Berrier, J.C., Shields, D.W., Kennefick, J., Kennefick, D., Seigar, M.S., Lacy, C.H.S., & Puerari, I., 2012, ApJS, 199, 33
Frigo, M. & Johnson, S.G., 1997, MIT-LCS-TR-728, http://www.fftw.org/fftw-paper.pdf