The Fractional Fourier Transform

Section F.5 in

B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, 1996.

F.5. Fractional Fourier transform

The FFT algorithm is extremely effective in calculating the matrix × vector product (F.1-3), and its inverse, when all input values are given and all output values are desired. Direct matrix × vector multiplication is normally much slower, but it does allow cost savings in some frequently occurring situations, as when:

(1) long sections of the input vector contain only zeros;

For example, soliton-type solutions to nonlinear wave equations may be nonzero (to machine precision) over only small sections of a long space domain.

(2) only some of the elements of the output vector are needed.

A large DFT matrix can provide a very accurate trapezoidal-rule approximation of the continuous Fourier integral $f(x) = (1/2\pi)\int_{-\infty}^{\infty} \hat{f}(\omega)e^{i\omega x}d\omega$ (or its inverse). Equation (F.1-3) imposes the same uniform discretization levels for $f(x)$ and $\hat{f}(\omega)$. These two functions can have very different characters. For example, we may want to evaluate $f(x)$ at a dense set of points over a small interval in some case where $\hat{f}(\omega)$ is best represented by a sparse set of values over a wide frequency range.

The fractional Fourier transform (FRFT) combines the best of both worlds. To calculate how any m (equi-spaced, e.g., contiguous) entries of the output vector depend on m (also equi-spaced) elements of the input vector, the cost becomes approximately four times that of a size-m FFT (independently of N, the size of the full DFT matrix).

The key observation (Bluestein 1970) is that any sum of the form

$$G_{n,\alpha}(k) = \sum_{j=0}^{m-1} x_j e^{-2\pi i(k+n)j\alpha}, \quad k = 0, 1, \ldots, m-1, \qquad \text{(F.5-1)}$$

n, α arbitrary constants,

can be reformulated as a (nonperiodic) convolution of size m. As noted in Section F.4, it can then be evaluated efficiently by a periodic convolution of twice the size (or larger, should that be faster or easier). Equation (F.5-1) amounts to a major generalization of (F.1-1) – the latter is recovered as the special case of $n = 0$ and $\alpha = 1/m$.

Noting that $2(k+n)j = j^2 + (k+n)^2 - (k+n-j)^2$, we obtain

$$G_{n,\alpha}(k) = e^{-i\pi(k+n)^2\alpha} \sum_{j=0}^{m-1} (x_j e^{-\pi i j^2\alpha})(e^{\pi i(k+n-j)^2\alpha}).$$

This sum can be written as $\sum_{j=0}^{m-1} y_j z_{k-j}$; i.e., it takes the form of a convolution. If several x vectors are to be used, all exponentials and the transform of the z vector should be precomputed.

Applications of the FRFT extend well beyond the two examples listed at the beginning of this section. Bailey and Swarztrauber (1991, 1994) point out several more, including:

(3) computing DFTs of sequences with arbitrary length N (e.g. a prime number, or not a product of small factors – the FFT routines in Section F.1 required N to be a power of 2);
(4) numerical computation of Laplace transforms (by selecting α to be complex);
(5) analyzing sequences with non-integer periodic components (noting that α need not be a rational number) – for example, an analysis of seasonal variations from daily measurements (with a year : day ratio of 365.2422...); and
(6) detecting lines in images, and detecting signals with linearly drifting frequencies.

References:

D. H. Bailey and P. N. Schwarztrauber (1991), *The fractional Fourier transform and applications*, SIAM Rev 33: 389-404.

D. H. Bailey and P. N. Schwarztrauber (1994), *A fast method for the numerical evaluation of continuous Fourier and Laplace transforms*, SIAM J. Sci. Comput. 15: 1105-1110.

L. I. Bluestein (1970), *A linear filtering approach to the computation of the discrete Fourier transform*, IEEE Trans. Audio Electroacoust. 18: 451-455.