Base-2 Fast Fourier Transform in ksTools

Sergey Kasandrov, 2009

http://sergworks.wordpress.com/kstools

Summary: the FFT and FFT-related procedures implemented in ksMath unit.

1. Introduction: Discrete Fourier Transform.

A fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT). The [forward] DFT of a discretely sampled complex data f_k , *k = 0..N-1* is usually defined by

$$F_n = \sum_{k=0}^{N-1} f_k e^{-2\pi ikn/N} \qquad (1)$$

The original data f_k can be exactly recovered from F_n by the inverse DFT:

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} F_k e^{2\pi ikn/N} \qquad (2)$$

The above definitions are mainly used in digital signal processing (DSP) and other application areas. The mathematicians prefer the symmetric definitions: the forward DFT is

$$F_n = \sum_{k=0}^{N-1} f_k e^{2\pi ikn/N} \qquad (3)$$

and the inverse DFT is

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} F_k e^{-2\pi ikn/N} \qquad (4)$$

To avoid confusion and to simplify the computational procedures we omit the normalization factor (1/N) and consider the symmetric "plus"

$$F_n = \sum_{k=0}^{N-1} f_k e^{2\pi ikn/N} \qquad (5)$$

and "minus"

$$F_n = \sum_{k=0}^{N-1} f_k e^{-2\pi ikn/N} \qquad (6)$$

transforms instead of "forward' and "inverse". We can interpret "plus" and "minus" transforms as we like - plus as forward and minus as inverse or vice versa.

2. Fast Fourier Transform.

The simplest and most widely used FFT algorithms are base-2 algorithms. The base-2 FFT algorithms require the N in the above formulae be a power of 2. There are two kinds of base-2 FFT algorithms - with time decimation and with frequency decimation. From the practical point of view there is little or no difference between them.

The basic FFT procedure in ksMath unit

procedure ComplexFFT(Data: Pointer; DataSize: Integer; Sign: Integer = 0);

implements both “plus” and “minus” base-2 complex FFT with time decimation. No normalization (division by N) for inverse transform is performed. The transformation is done "in place" and the resulting array is DFT of the original data.
The Data argument is a pointer to a packed array of complex values of the type

```
TksComplex = packed record
  Re, Im: Extended;
end;
```

The DataSize is a number of values in the complex array, must be a power of 2.
The Sign argument defines a kind of the transform: Sign >= 0 means “plus” transform while Sign < 0 means “minus” transform.

The ComplexFFT procedure can be used to transform the real functions by representing them as complex values with zero imaginary part, but this can be done more effectively by the RealFFT procedure:

procedure RealFFT(Data: Pointer; DataSize: Integer; Sign: Integer = 0);

The Data argument is a pointer to a packed array of real values of the type Extended.
The DataSize is a number of values in the array, must be a power of 2.
The Sign argument defines a kind of the transform.

Though the RealFFT arguments are the same as ComplexFFT arguments, their meaning is different. First of all, the DFT of a real function f_k is a symmetric complex function F_n satisfying

$$F_{N-n}=\left(F_n\right)^* \tag{7}$$

where the asterisk denotes complex conjugation. Because of (7) the resulting 2N real values constituting N complex values F_n can be packed into the N real values. The N real values R_n returned by RealFFT with Sign >= 0 are actually represents N complex values G_n . Since the “minus” transform of a real function is a complex conjugate of the “plus” transform, the output of the RealFFT procedure with Sign >= 0 can be interpreted both as a result of the “plus” or “minus” transform. For the “plus” transform the relation is

$$\begin{aligned} G_0 &= R_0 \quad [\Im(G_0)=0] \\ G_{N/2} &= R_1 \quad [\Im(G_{N/2})=0] \\ G_n &= R_{2n}+iR_{2n+1} \ , \quad 0<n<N/2 \\ G_{N-n} &= R_{2n}-iR_{2n+1} \ , \quad 0<n<N/2 \end{aligned} \tag{8}$$

If we interpret the output of the RealFFT procedure with Sign >= 0 as "plus" transform, the output of the RealFFT with Sign<0 is "minus" transform of a complex function G_n satisfying

$$G_{N-n}=(G_n)^*$$

represented as an array of N real values R_n according to (8).

As with ComplexFFT, the normalization is not performed, the transformation is done in-place and the DataSize argument must be a power of 2.

3. Correlation and autocorrelation.

The circular correlation of the two discretely sampled functions f_k and g_k is defined by

$$Corr(f,g)_n=\sum_{k=0}^{N-1} f_{k+n} g_k \tag{9}$$

The circular correlation can be effectively computed using the fast Fourier transform. The corresponding procedure for real functions is

procedure RealCorr(Data1, Data2: Pointer; Corr: Pointer; DataSize: Integer);

Data1 and Data2 arguments are pointers to packed real (type Extended) arrays containing functions values;
Corr is a pointer to the packed real (type Extended) array that receives the computed correlation.
DataSize is a number of values in the arrays, must be a power of 2.

The circular auto-correlation of a real function can be computed by

procedure RealAutoCorr(Data: Pointer; DataSize: Integer; Spectrum: Boolean = False);

the computation is done in-place and the DataSize argument must be a power of 2.
Spectrum argument defines the type of data pointed by Data argument. If Spectrum = False, the Data points to the original function data, Spectrum = True means the output of RealFFT procedure with Sign >= 0.

4. Helper functions.

The output of RealFFT procedure with Sign >= 0 can be interpreted as "plus" or "minus" transform. In the application area (DSP) the "minus" interpretation is preferable. To simplify the "minus" interpretation of the array returned by RealFFT procedure the ksMath unit introduces 3 helper functions:

function RFTSpectrum(Data: Pointer; DataSize, Index: Integer): TksComplex;

function RFTAmplitude(Data: Pointer; DataSize, Index: Integer): Extended;
function RFTPhase(Data: Pointer; DataSize, Index: Integer): Extended;

The RFT prefix means “Real Fourier Transform”.
Data is a pointer to the real array returned by RealFFT procedure,
DataSize is the number of values in the array,
Index is a position of an element of the array.

5. Final remarks.

The ksTools distribution includes FFTTests project implementing the unit tests for FFT and FFT-related procedures from ksMath unit. The tests contained in TestFFT.pas unit are also good usage example for all procedures mentioned above.