scift user manual

N.F. Aguirre

February 4, 2015

Contents

1 Introduction

The continuous Fourier transform (CFT) of a function $f(t)$ and its inverse will be defined here as

$$\begin{aligned} F(p) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x)\ e^{-ipx}\, dx \\ f(x) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(p)\ e^{ipx}\, dp \end{aligned} \tag{1}$$

where i is the imaginary unit. The discrete Fourier Transform (DFT) and the inverse DFT of a secuence $\mathbf{z} = \{z_1, z_2, \ldots, z_n\}$ (which may have real or complex values) will be defined as

$$\begin{aligned} D_k(z) &= \sum_{j=1}^{n} z_j e^{-2\pi i(j-1)(k-1)/n} \qquad \therefore \qquad k = 1, 2, \ldots, n \\ D_k^{-1}(z) &= \frac{1}{n} \sum_{j=1}^{n} z_j e^{2\pi i(j-1)(k-1)/n} \qquad \therefore \qquad k = 1, 2, \ldots, n \end{aligned} \tag{2}$$

An Fast Fourier Transform (FFT) computes the DFT and produces exactly the same result as evaluating the DFT definition directly; the most important difference is that an FFT is much faster.

Let us assume that $f(x)$ is a zero outside the interval $(-L/2, L/2)$. Let $\Delta x = L/n$ be the interval in x for the n input values of $f(x)$.

Further, Fourier transform software routines require a fixed discrete grid spacing. Then the sampling interval on grid spacing is $\Delta x = a/n$, and $\Delta p = 2\pi/(n\Delta x)$ on momentum spacing.

$$\begin{aligned} x_j &= j\Delta x, \qquad f_j = f(x_j) \qquad \therefore \qquad j = -\frac{n}{2}, \ldots, \frac{n}{2} \\ p_k &= k\Delta p, \qquad F_k = F(p_k) \qquad \therefore \qquad k = -\frac{n}{2}, \ldots, \frac{n}{2} \end{aligned} \tag{3}$$

Then, we can write CFT as follows

$$F_k = \frac{\Delta x}{\sqrt{2\pi}} \sum_{j=-n/2}^{n/2} f_j\ e^{-2\pi ikj/n} \qquad \therefore \qquad k = -\frac{n}{2}, \ldots, \frac{n}{2} \tag{4}$$

However most of implementations use positive indices, and therefore the spatial samples inside the sum must be re-arranged. According with the FFTW library we use the standard order (*s-order*) vector. This means that the positive values are stored in the first half of the vector and the negative frequencies are stored in backwards order in the second half of the vector, with the zero-value component in the first element. We will represent this format as a *tilde*, for example: $\mathbf{z}$ represent the array in *s-order* and $\tilde{\mathbf{z}}$ in *n-order*.

$$s\text{-order:} \qquad \boldsymbol{z} = \Big(z\Big)_{J=1}^{n} \qquad\qquad n\text{-order:} \qquad \tilde{\boldsymbol{z}} = \Big(z\Big)_{j=-n/2}^{n/2} \tag{5}$$

where both sequences are related throught the following functions,

$$\tilde{\boldsymbol{z}} = \text{fftshift}(\boldsymbol{z}) \qquad\qquad \boldsymbol{z} = \text{ifftshift}(\tilde{\boldsymbol{z}}) \tag{6}$$

The next table represents the structure corresponds in the case where n is an even number

n	$n/2-1$				$n/2+1$			
j	$-\frac{n}{2}+1$	$-\frac{n}{2}+2$	$\cdots$	-1	0	$\cdots$	$\frac{n}{2}-1$	$\frac{n}{2}$
J	$\frac{n}{2}+2$	$\frac{n}{2}+3$	$\ldots$	n	1	$\ldots$	$\frac{n}{2}$	$\frac{n}{2}+1$

$n=8$	3				5			
j	-3	-2	-1	0	1	2	3	4
J	6	7	8	1	2	3	4	5

If n is odd then general structure of the table above still applies, but $n/2+1$ does not appear.

n	$\lfloor \frac{n}{2} \rfloor$				$\lfloor \frac{n}{2} \rfloor + 1$			
j	$-\lfloor \frac{n}{2} \rfloor$	$-\lfloor \frac{n}{2} \rfloor + 1$	$\cdots$	-1	0	$\cdots$	$\lfloor \frac{n}{2} \rfloor - 1$	$\lfloor \frac{n}{2} \rfloor$
J	$\lfloor \frac{n}{2} \rfloor + 1$	$\lfloor \frac{n}{2} \rfloor + 2$	$\ldots$	n	1	$\ldots$	$\lfloor \frac{n}{2} \rfloor - 1$	$\lfloor \frac{n}{2} \rfloor$

$n=7$	3			4			
j	-3	-2	-1	0	1	2	3
J	5	6	7	1	2	3	4

After this transformation Equation (4) can be reexpressed as follow

$$F_K = \frac{\Delta x}{\sqrt{2\pi}} \sum_{J=1}^{n} f_J \; e^{-2\pi i (K-1)(J-1)/n} \qquad \therefore \qquad K = 1, 2, \ldots, n \tag{7}$$

The reordering of spatial samples means that the frequency samples should be re-arranged as well. Now it is clear that we can use the DFT definition to express the FT (Denote K the new index in the frequency domain, which leads to the following expression:)

$$F_K = \frac{\Delta x}{\sqrt{2\pi}} D_K(\boldsymbol{f}) \qquad \therefore \qquad K = 1, 2, \ldots, n \tag{8}$$

To be specific, the abscissas for the input data are,

$$x_j = \left(j - 1 - \frac{n}{2}\right)\Delta x \qquad \therefore \qquad j = 1, 2, \ldots, n \tag{9}$$

The abscissas for the output data set will be defined as,

$$p_k = \left(k - 1 - \frac{n}{2}\right)\Delta p \qquad \therefore \qquad k = 1, 2, \ldots, n \tag{10}$$

where $\Delta p = 2\pi/a = 2\pi/n\Delta x$ which conduce to the incertidumbre principle $\Delta x \Delta p = 2\pi/n$ and the important known as "angular Nyquist frequency" $\Omega = \pi/\Delta x$, which corresponds to the maximum and the negative mimimum value of frequency for the series.

2 Convention

To facilitate our discusion, we first define two general functions $f(x)$ and $F(p)$, which are Fourier transforms of each other,

$$\begin{aligned} \psi(x) &= \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} \psi(p)\ e^{ipx/\hbar}\, dp \\ \psi(p) &= \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} \psi(x)\ e^{-ipx/\hbar}\, dx \end{aligned} \tag{11}$$

In general we are going to use atomic units ($\hbar = 1$).

$\texttt{FFT}(\mathbf{f},\ \tilde{\mathbf{F}}^{\pm} \to \mathbf{f},\ \text{sign} = +1)$	$\tilde{F}^{\pm}_{\alpha} = \dfrac{1}{\sqrt{2\pi}} \displaystyle\sum_{\beta=1}^{n} f_\beta\ \exp\left\{\pm 2\pi i\ \frac{(\alpha-1)(\beta-1)}{n}\right\}, \qquad \alpha = 1, 2, \ldots, n$
$\texttt{iFFT}(\tilde{\mathbf{F}},\ \mathbf{f}^{\pm} \to \tilde{\mathbf{F}},\ \text{sign} = -1)$	$f_\beta = \dfrac{1}{n\sqrt{2\pi}} \displaystyle\sum_{\alpha=1}^{n} \tilde{F}^{\pm}_{\alpha}\ \exp\left\{\mp 2\pi i\ \frac{(\beta-1)(\alpha-1)}{n}\right\}, \qquad \beta = 1, 2, \ldots, n$

Table 1: asdasd

3 Fast Fourier transform (FFT)

Let $\mathbf{f} = \{f_1, f_2, \ldots, f_n\}$ and $\mathbf{F} = \{F_1, F_2, \ldots, F_n\}$ two arrays of complex numbers. The direct Discrete Fourier Transform (dDFT) of the array $\mathbf{f}$ is defined by the formula

$$F^{\pm}_{\alpha} = \sum_{\beta=1}^{n} f_\beta\ \exp\left\{\pm 2\pi i\ \frac{(\alpha-1)(\beta-1)}{n}\right\}, \qquad \alpha = 1, 2, \ldots, n \tag{12}$$

The inverse DFT (iDFT) for the array $\mathbf{F}$ is defined as

$$f_\beta = \frac{1}{n} \sum_{\alpha=1}^{n} F^{\pm}_{\alpha}\ \exp\left\{\mp 2\pi i\ \frac{(\beta-1)(\alpha-1)}{n}\right\}, \qquad \beta = 1, 2, \ldots, n \tag{13}$$

It differs from the direct transform by the normalization constant $1/n$.

The forward DFT (fDFT) corresponds to a sign of $+1$ in the exponent and backward DFT (bDFT) to a sign -1. Traditionally the dDFT is associated with the sign $+$ and a the iDFT with the sign -1, in such case dDFT is equivalent to fDFT and iDFT is equivalent to bDFT. However, here we will keep this difference.

According with the FFTW library we use the standard order (*s-order*) output. This means that the positive frequencies are stored in the first half of the output and the negative frequencies are stored in backwards order in the second half of the output, with the zero-frequency component in the first element. We will represent this format as a *tilde*, for example: $\mathbf{x}$ represent the array in *n-order* and $\tilde{\mathbf{x}}$ in *s-order*.

There are three different ways to use the available methods available in scift to calculate the FFT.

$$[\mathbf{x},\mathbf{f}] \underset{\text{iFFT}(\mp)}{\overset{\text{dFFT}(\pm)}{\rightleftarrows}} \left[\tilde{\boldsymbol{\omega}},\tilde{\mathbf{F}}^{\pm}\right] \underset{\text{ishift}}{\overset{\text{shift}}{\rightleftarrows}} [\boldsymbol{\omega},\mathbf{F}^{\pm}] \underset{\text{iFFT}(\mp)}{\overset{\text{dFFT}(\pm)}{\rightleftarrows}} \left[\tilde{\boldsymbol{x}},\tilde{\mathbf{f}}^{\pm}\right] \underset{\text{ishift}}{\overset{\text{shift}}{\rightleftarrows}} [\boldsymbol{x},\mathbf{f}^{\pm}] \tag{14}$$

$$[\mathbf{x},\mathbf{f}] \longrightarrow \left[\tilde{\boldsymbol{\omega}},\tilde{\mathbf{F}}\right] \longrightarrow [\boldsymbol{\omega},\mathbf{F}] \tag{15}$$

When n is even the location $n/2$ contains the most positive and negative frequencies $\pm\frac{n/2}{nh}$ $\left(=\pm\frac{1}{2h}\right)$ which are equivalent. This frequency is called Nyquist frequency. This is the highest frequency component that should exist in the input series for the DFT to yield "uncorrupted" results. More specifically if there are no frequencies above Nyquist the original signal can be exactly reconstructed from the samples.

$n/2+1$					$n/2-1$			
$\tilde{\omega}_1$	$\tilde{\omega}_2$	$\tilde{\omega}_3$	$\dots$	$\tilde{\omega}_{n/2+1}$	$\tilde{\omega}_{n/2-1}$	$\dots$	$\tilde{\omega}_{n-1}$	$\tilde{\omega}_n$
0	$\frac{1}{nh}$	$\frac{2}{nh}$	$\dots$	$\frac{n/2}{nh}$	$-\frac{n/2-1}{nh}$	$\dots$	$-\frac{2}{nh}$	$-\frac{1}{nh}$

$n/2-1$				$n/2+1$				
ω_1	$\dots$	$\omega_{n/2}$	$\omega_{n/2+1}$	$\omega_{n/2}$	$\omega_{n/2+1}$	$\omega_{n/2+2}$	$\dots$	ω_n
$\tilde{\omega}_{n/2+1}$	$\dots$	$\tilde{\omega}_{n-1}$	$\tilde{\omega}_n$	$\tilde{\omega}_1$	$\tilde{\omega}_2$	$\tilde{\omega}_3$	$\dots$	$\tilde{\omega}_{n/2}$
$-\frac{n/2-1}{nh}$	$\dots$	$-\frac{2}{nh}$	$-\frac{1}{nh}$	0	$\frac{1}{nh}$	$\frac{2}{nh}$	$\dots$	$\frac{n/2}{nh}$

If n is odd then general structure of the table above still applies, but $n/2+1$ does not appear.

$n/2$					$n/2-1$			
$\tilde{\omega}_1$	$\tilde{\omega}_2$	$\tilde{\omega}_3$	$\dots$	$\tilde{\omega}_{n/2}$	$\tilde{\omega}_{n/2+1}$	$\dots$	$\tilde{\omega}_{n-1}$	$\tilde{\omega}_n$
0	$\frac{1}{nh}$	$\frac{2}{nh}$	$\dots$	$\frac{n/2-1}{nh}$	$-\frac{n/2-1}{nh}$	$\dots$	$-\frac{2}{nh}$	$-\frac{1}{nh}$

$n/2-1$				$n/2$				
ω_1	$\dots$	$\omega_{n/2}$	$\omega_{n/2+1}$	$\omega_{n/2}$	$\omega_{n/2+1}$	$\omega_{n/2+2}$	$\dots$	ω_n
$\tilde{\omega}_{n/2+1}$	$\dots$	$\tilde{\omega}_{n-1}$	$\tilde{\omega}_n$	$\tilde{\omega}_1$	$\tilde{\omega}_2$	$\tilde{\omega}_3$	$\dots$	$\tilde{\omega}_{n/2}$
$-\frac{n/2-1}{nh}$	$\dots$	$-\frac{2}{nh}$	$-\frac{1}{nh}$	0	$\frac{1}{nh}$	$\frac{2}{nh}$	$\dots$	$\frac{n/2-1}{nh}$

3.1 Directly (High level interface)

By using this option, scift internally make the shift and normalization for the x coordinate.

$$\left[x,f(x)\right] \xrightarrow{5} \left[\omega,\hat{f}(\omega)\right] \tag{16}$$

$$\left[\omega,\hat{f}(\omega)\right] \xrightarrow{8} \left[x,f(x)\right] \tag{17}$$

```
call xGrid.init( -3.0_8*Math_PI, 3.0_8*Math_PI, 1000 )
call funcA.init( xGrid, funcTestWithNoise )
call funcA.save("func")

funcA = FFT_fft( funcA )
call funcA.save("Ffunc")

funcA = FFT_ifft( funcA )
call funcA.save("iFFfunc")
```

This is the result

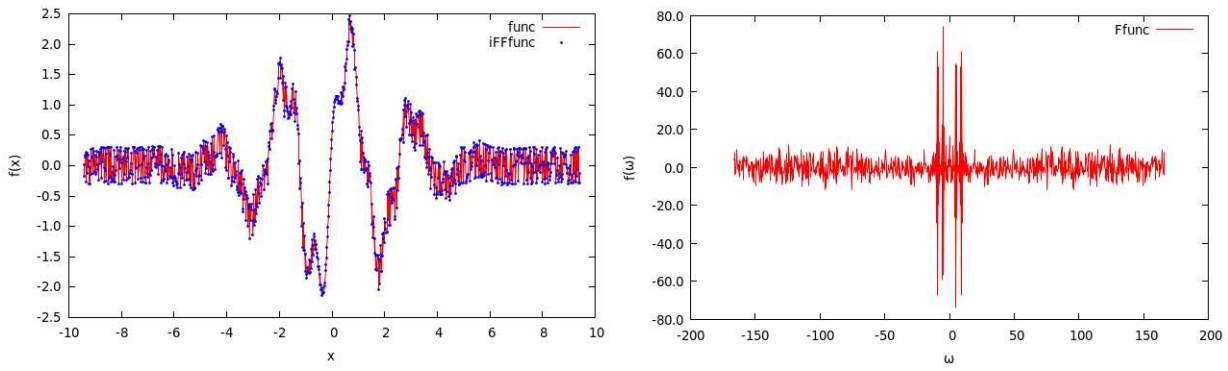

Figure 1: Geometria para s=3

3.2 By using plans (Low level interface)

In this case is neccesary to keep in mind the reorganization on the frequency axis and the normalization of the Fourier transform

$$\Big[x, f(x)\Big] \xrightarrow{8} \Big[x, \hat{f}(\tilde{\omega})\Big] \xrightarrow{9} \Big[\tilde{\omega}, \hat{f}(\tilde{\omega})\Big] \xrightarrow{10} \Big[\omega, \hat{f}(\omega)\Big] \tag{18}$$

$$\Big[\omega, \hat{f}(\omega)\Big] \xrightarrow{13} \Big[\tilde{\omega}, \hat{f}(\tilde{\omega})\Big] \xrightarrow{14} \Big[\tilde{\omega}, F(x)\Big] \xrightarrow{15} \Big[x, F(x)\Big] \xrightarrow{16} \Big[x, f(x)\Big] \tag{19}$$

```
call xGrid.init( -3.0_8*Math_PI, 3.0_8*Math_PI, 1000 )
call funcA.init( xGrid, funcTestWithNoise )
call funcA.save("func")

planF = FFT_plan( funcA, FFT_FORWARD )
planB = FFT_plan( funcA, FFT_BACKWARD )

call FFT_execute( planF )
funcA.xGrid = FFT_omegaGrid( funcA.xGrid )
call FFT_shift( funcA )
call funcA.save("Ffunc")

call FFT_ishift( funcA )
call FFT_execute( planB )
funcA.xGrid = FFT_xGrid( funcA.xGrid )
funcA = funcA/real( funcA.nPoints(), 8 )
call funcA.save("iFFfunc")

call FFT_destroyPlan( planF )
```

3.3 By using the oriented object interface

The function

$$\Big[x, f(x)\Big] \begin{array}{ll} \xleftrightarrow{\text{sync=F}} & \Big[x, \hat{f}(\tilde{\omega})\Big] \\ \xleftrightarrow{\text{sync=T}} & \Big[\tilde{\omega}, \hat{f}(\tilde{\omega})\Big] \\ \xleftrightarrow{\text{sync=T,shift=T}} & \Big[\omega, \hat{f}(\omega)\Big] \end{array} \tag{20}$$

```
call xGrid.init( -3.0_8*Math_PI, 3.0_8*Math_PI, 1000 )
call funcA.init( xGrid, funcTestWithNoise )
call funcA.save("func")
```

```
4
5  call fft.init( funcA, FFT_SPATIAL_DOMAIN )
6
7  call fft.execute( FFT_FORWARD, sync=.true., shift=.true. )
8  call funcA.save("Ffunc")
9
10 call fft.execute( FFT_BACKWARD, sync=.true., shift=.true. )
11 call funcA.save("iFFfunc")
```

3.4 Calculating the derivative of second order of a signal

The function

$$\frac{d^n}{dx^n} f(x) = (i\omega)^n \hat{f}(\omega) \tag{21}$$

$$\Big[x, f(x)\Big] \xrightarrow{7} \Big[x, \hat{f}(\tilde{\omega})\Big] \xrightarrow{9} \Big[x, (i\tilde{\omega})^2 \hat{f}(\tilde{\omega})\Big] \xrightarrow{11} \Big[x, \frac{d^2}{dx^2} f(x)\Big] \tag{22}$$

```
1  call xGrid.init( -3.0_8*Math_PI, 3.0_8*Math_PI, 1000 )
2  call funcA.init( xGrid, funcTest )
3  call funcA.save("func")
4
5  call fft.init( funcA, FFT_SPATIAL_DOMAIN )
6
7  call fft.execute( FFT_FORWARD )
8
9  funcA.yArray = ( Math_I*fft.omega )**2*funcA.yArray
10
11 call fft.execute( FFT_BACKWARD )
12
13 call funcA.save("dfunc")
14
15 call funcB.init( xGrid, d2funcTest )
16 call funcB.save("exact")
```

This is the result

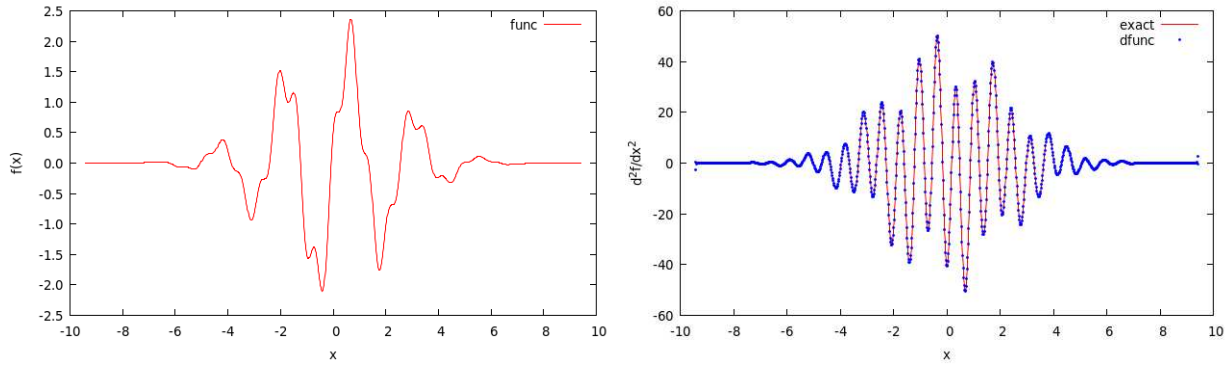

Figure 2: Left panel: Original function. Right panel: Derivative of second order of the signal calculated by using the FFT interface, the exact solution is also included.