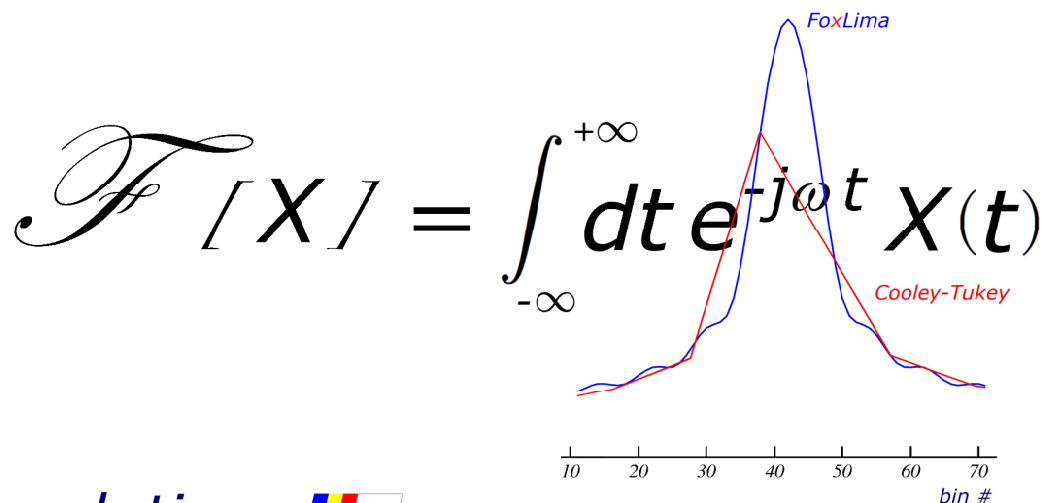


FoxLima

non-abelian FFT



NXV4 solutions 

fftnxv4@gmail.com

Performance

- ▶ $71 \log(N)$ ns / sample Cooley-Tukey<double>
- ▶ $0.083 N^{0.79}$ ns / sample FoxLima<double>

on: proc: x86_64 Intel Xeon E312xx Sandy Bridge @ 2.6 GHz
 cache: 32k, 32k, 4096k
 op-sys: Linux: 2.6.32-504.16.2.el6.x86_64
 gcc: 4.4.7 20120313 (Red Hat 4.4.7-11)

Header file: #include "fxl.hh"

Libraries: libfxl4.a (static) linux-2.6.32-504.12.2.el6.x86_64
libfxl4.so (dynamic) gcc-4.4.7 20120313

Class ... fxl<T>std. T = double, long double
cpx<double>, cpx<long double>
extra T = mtx<T>

Member variables (public):

- ▶ ddc / ddr = T pointers to Np long arrays of "imag" / "real" part
(for $mtx<T> = A + iB$, A,B self-adjoint)
- ▶ fzc / fzr = T pointers to Np long arrays of "imag" / "real" part
(for $mtx<T> = A + iB$, A,B self-adjoint) after
the FoxLima filter
- ▶ N = int # of Fourier bins = Np
- ▶ p = int oversampling ratio
- ▶ rms = double signal rms

Constructor (public)

```
fxl F( $\hat{T}^*$ ,  $\hat{T}^*$ ,  $\hat{N}$ ,  $\hat{q}$ ,  $\hat{str}$ , ...)
|      |      |      |      |      |
|      |      |      |      |      |_ arg's for the command (see below)
|      |      |      |      |      |_ command
|      |      |      |      |      |_  $2^q$  FFT bins - allows oversampling,  $p = 2^q / N$ , recommended  $q$ :
|      |      |      |      |      |_  $q > 3 + \log(N) / \log(2)$ 
|      |      |      |      |      |_ # i/p samples
|      |      |      |      |      |_ pointer "real"-part (or vec of matrices, symmetrical part)
|      |      |      |      |      |_ pointer "imag"-part (or vec of matrices, anti-symmetrical part)
```

command = "XX", or "XX raw" – with XX = F0, F1, F2, F4, F8, F16, GX, FC, FX, FE, FW
F0 = Cooley-Tukey, -6dB/oct tails, $p = 2^q / N$
F1 = FoxLima-F1, -6dB/oct tails, $p = 2^q / N$
F2 = FoxLima-F2, -12dB/oct tails, $p = 2^q / N$, similar to Welch apodisation,
but w/ exact cancellation of $1 / (f-f_0)$ tails
F4 = FoxLima-F4, -24dB/oct tails, $p = 2^q / N$, suis-generis Welch-4
window, but w/ exact cancellation of
 $1 / (f-f_0)^k$ tails (for $k = 1, 2, 3$)
F8 = FoxLima-F8, -48dB/oct tails, $p = 2^q / N$, suis-generis Welch-8 window
F16 = FoxLima-F16, -96dB/oct tail, $p = 2^q / N$, suis-generis Welch-16
GX = Gaussian(sigma), $p = 2^q / N$, gaussian apodisation
FC = coherent chirp, $p = 2^q$ with no windowing, σ size
FX = coherent chirp, $p = 2^q$ with Gaussian(σ) window
FE = coherent chirp, $p = 2^q$ with back-exp(τ) window
FW = coherent chirp, $p = 2^q$ with Welch(σ) window

For the following: λ_{-3} p = # bins at which smooth function falls -3 dB
 λ_{-20} p = # bins at which smooth function falls -20 dB
 condition: $1.472 < \lambda_{-20} / \lambda_{-3} < 4.684$
 $2f_0 \Delta t = f_0 / \text{FSR} = \text{chirp coherence-frequency}$
 $\sigma_{\text{chirp}} = \text{chirp coherence-length \# bins}$

case	F0, F1, F2, F4, F8, F16 raw	arg = void
	F0, F1, F2, F4, F8, F16 smooth	arg = $\lambda_{-3}, \lambda_{-20}$
	GX raw	arg = σ
	FC raw	arg = $2f_0 \Delta t, \sigma_{\text{chirp}}$
	FX raw	arg = $2f_0 \Delta t, \sigma_{\text{chirp}}$
	FE raw	arg = $2f_0 \Delta t, \tau_{\text{chirp}}$
	FW raw	arg = $2f_0 \Delta t, \sigma_{\text{chirp}}$
	GX smooth	arg = $\sigma, \lambda_{-3}, \lambda_{-20}$
	FC smooth	arg = $2f_0 \Delta t, \sigma_{\text{chirp}}, \lambda_{-3}, \lambda_{-20}$
	FX smooth	arg = $2f_0 \Delta t, \sigma_{\text{chirp}}, \lambda_{-3}, \lambda_{-20}$
	FE smooth	arg = $2f_0 \Delta t, \tau_{\text{chirp}}, \lambda_{-3}, \lambda_{-20}$
	FW smooth	arg = $2f_0 \Delta t, \sigma_{\text{chirp}}, \lambda_{-3}, \lambda_{-20}$

Parameters

- ▶ $f_n = (n - Np/2) / Np / T$ frequency of bin n
- ▶ $A_n = \sqrt{(\text{ddr}_n^2 + \text{ddc}_n^2)} / Np$ amplitude of bin n (Cooley-Tukey)
- $\sqrt{(\text{fzr}_n^2 + \text{fzc}_n^2)} / Np$ - - " - - (FoxLima)

Resolution

- ▶ $df / f = 0.7 / \text{int}(f T)$ frequency resolution
- ▶ $dA / A = \text{from } i/p$ amplitude resolution

Description

The **f_{xl} class** is a suite of D-FFT methods.

COHERENT SIGNALS

F0 – is the classic Cooley-Tukey algorithm. It receives N time bins and outputs 2^q bins, where $p = 2^q / N$ is the (frequency)-oversampling ratio (recommended around x8). The **oversampling** solves the imprecision of Cooley-Tukey's algorithm which for certain frequencies (periodically in the spectrum) has large errors, missing their peaks in:

- frequency: $\delta f < 1 / Np\Delta t$,
- amplitude: $\delta A / A < 2p / \pi$,
- phase: $\delta \phi < \pi / 2p$.

It is evident that for $p = 1$ the errors can be large.

The Cooley-Tukey algorithm has -6 dB/oct fall-off spectrum leakage tails. For certain peaks, with $f = n / T$, this is not visible as the sampling occurs exactly at the position of the zero's. For all else it is visible with various degrees. Metrologically, the favourable peaks do not represent the real performance of the algorithm.

- F1** – is a filter to the Cooley-Tukey algorithm. It solves the problem of the Gibbs satellite-lobes, but has still the spectrum leakage tails, also -6 dB/oct fall-off. Its advantage is speed, requiring little CPU time.
- F2** – is the best trade-off Cooley-Tukey filter, between thin peak width and Gibbs lobes' suppression. It has a -12 dB/oct fall-off of spectrum leakage tails. Suppression is exact with respect to digitisation – in order $k = 1$ of the $1 / (f-f_0)^k$ tails. It is loosely equivalent to Welch apodisation, however performed in frequency space rather than in temporal space.
- F4** – has an advanced rejection of Gibbs lobes and -24 dB/oct fall-off of spectrum leakage tails. Suppression is exact with respect to digitisation – in orders $k = 1, 2, 3$ of the $1 / (f-f_0)^k$ tails. It is loosely equivalent to Welch-4 apodisation, however performed in frequency space rather than in temporal space.
- F8** – has an advanced rejection of Gibbs lobes and -48 dB/oct fall-off of spectrum leakage tails. It is a Welch-8 apodisation, performed in temporal space. It is susceptible to spectrum distortions due to inexact cancellations of digitisation terms when the signal has large phase noise, or it represents a short temporal sequence.
- F16** – has an advanced rejection of Gibbs lobes and -96 dB/oct fall-off of spectrum leakage tails. It is a Welch-16 apodisation, performed in temporal space. It is susceptible to spectrum distortions due to inexact cancellations of digitisation terms when the signal has large phase noise, or it represents a short temporal sequence.
- GX** – has an advanced rejection of Gibbs lobes and performant fall-off of spectrum leakage tails. It is a gaussian apodisation, performed in temporal space, of user given σ .

NOISE SIGNALS

The signal below seems to have a well defined FFT peak in the spectrum, as being a single frequency.

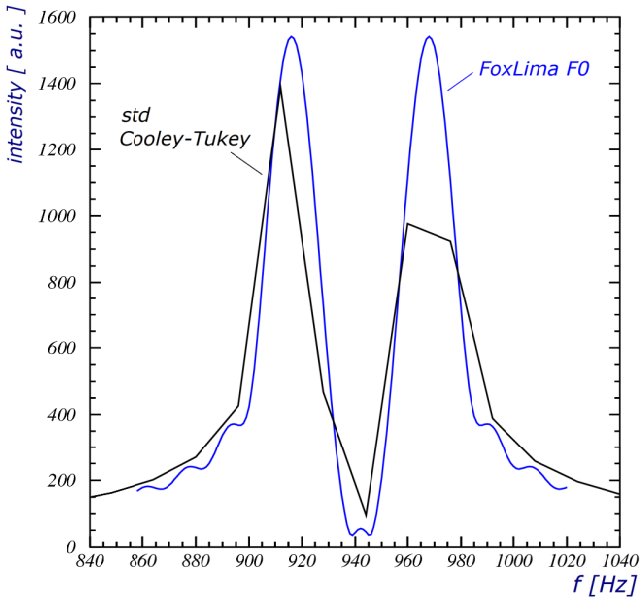


The problem however is that its second part is phased 180° behind the first. As such, when they are added, the FFT signal will be zero.

This may be mathematically correct, however engineering-wise unsatisfactory: there is a definite frequency in the spectrum, which needs to be flagged accordingly.

Similarly, certain sources emit “chirps”, of valid FFT-spectrum, however (usually) at random time intervals (same $\Delta\phi/\omega$ for all frequencies), and possibly also at various phase lags among frequencies.

For a given frequency the input sample can be divided into smaller samples and for each the FFT be computed. The results can be then summed as absolute values. For the frequency asked this will give the maximum amplitude attainable if all samples are coherent.



The problem is that this procedure has poor resolution. For any FFT the thinness of the peaks is given by the sum $\sum_{k=1, N-1} \exp(2\pi i(\omega - \omega_0)kT)$ which gives an “Airy”-function of frequency resolution $\delta f = 1 / NT$.

To overcome this, a phase-jump estimate at f_0 is performed, then this is applied to all other frequencies. This should be adjusted for frequency drift, however for those around f_0 it is sufficient (and other distant frequencies would have entailed too much noise from the extrapolation anyway).

This preserves the main mechanism of FFT computation, the summation of the above mentioned sum, and hence the resolution of signals – in spite of the small sample partition (and poor afferent σ_f apodisation).

FC – is without windowing of the smaller samples. User given σ and f_0 .

FX – is a gaussian apodisation of the smaller samples. User given σ and f_0 .

FE – is a back-exponential apodisation of the smaller samples. User given τ and f_0 .

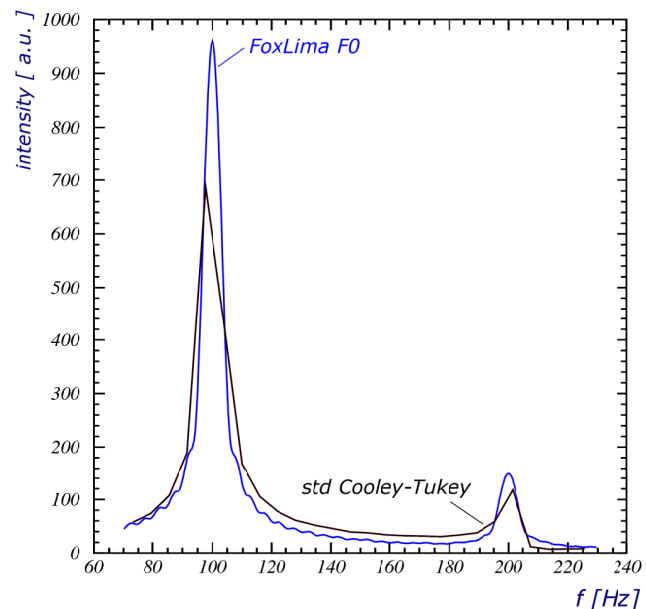
FW – is a Welch apodisation of the smaller samples. User given σ and f_0 .

Performance

The main advantages of the FoxLima suite are:

1. *exact signal metrics* – due to the possibility to oversample in frequency-space, FoxLima algorithms provide exact peak resolution over Cooley-Tukey, that can register errors of up to:
 - $\Delta f = 1 / 2N\Delta t$ in frequency
 - $\Delta A = 36\% A$ in amplitude
 - $\Delta \phi = 90^\circ$ in phase

These aspects can be seen in the figure above, and the figure to the right, where the sub-sampling problem of the Cooley-Tukey algorithm are evidenced as peak truncation, or central-frequency imprecision.

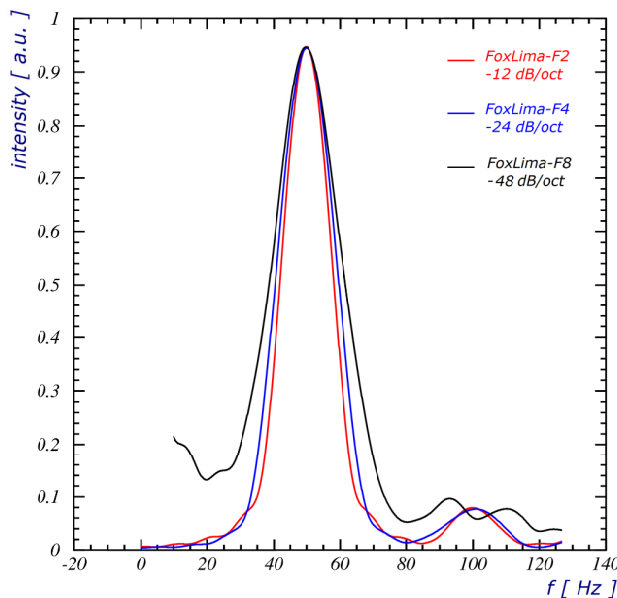
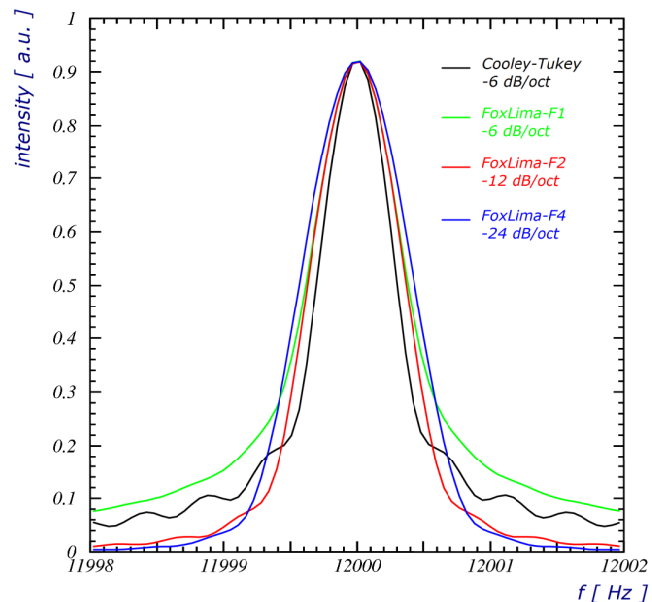


2. *exact leakage elimination* – due to its Fourier-space apodisation, algorithms F1, F2 and F4 eliminate exactly the $1 / (f-f_0)^k$ spectrum-leakage tails for:
- $k = 0$ (F1 algorithm)
 - $k = 1$ (F2 algorithm) and
 - $k = 1,2,3$ (F4 algorithm).

The figure to the right shows smoothed versions of the algorithms for width and tail fall-off.

The 3 Fourier-space apodisation algorithms have fall-offs of:

- -6 dB/oct (F1 algorithm)
- -12 dB/oct (F2 algorithm)
- -24 dB/oct (F4 algorithm).



peak, whereas F2 and F4 are both very precise.

It is important to note, that time-space apodisation does not take into account discretisation components exactly as does Fourier-space apodisation.

Therefore signals with phase-noise, or a short time data-acquisition window are not described as exact by time-space apodisation algorithms.

This is shown in the figure here to the left – on a short time-window (of 6 x 50 Hz periods).

The F8 algorithm has significant deviations beyond the central-

3. *Weyl-Wigner FFT* – as mentioned above, random-chirp signals may deceptively add destructively showing little (or no signal at all) at certain frequencies.

The Weyl-Wigner algorithms add coherently such signals around a (user given) coherence frequency (f_0), of sampling window (the approximate size of the chirp, σ).

A comparison with the F2 algorithm (on a noise signal) is given in the figure here below.

A small presence in F2, with a large one in WW denotes significant cancellation (or phase noise).

The apodisation over the σ window is:

- **FC** – plain Cooley-Tukey
- **FX** – gaussian
- **FW** – Welch
- **FE** – back-exponential

The apodisation used in the figure here is gaussian.

