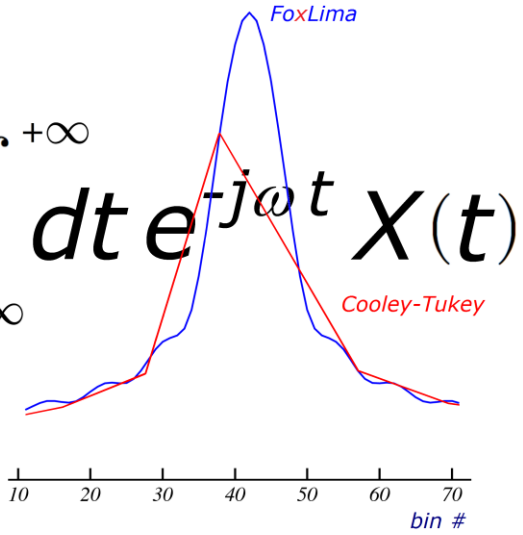
FoxLima

non-abelian FFT

$$\mathcal{F}[X] = \int_{-\infty}^{+\infty} dt\, e^{-j\omega t} X(t)$$



NXV4 solutions

Purpose

- Fourier transform (Cooley-Tukey), radix 2^p:
  - interval: smaller, equal, or larger than 2^p
  - super-sampling: for metrological quality (A, f, φ)
  - apodisation methods:
    - in Fourier-space: `F2, F4`
    - in real-space: `F8, F16, GX`

- Weyl-Wigner Fourier transform for noise-signals:
  - window interval coherentisation of random chirps around given frequency

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> ..... T = `double, long double, cpx<double>, cpx<long double>`

Member **variables** (public):

- `ddr / ddc` = `T` .... pointers to *Np* long arrays of "real" / "imag" part of spectrum raw [0 ... +F -F ... 0]
- `fzr / fzc` = `T` .... pointers to *Np* long arrays of "real" / "imag" part of spectrum after FoxLima filter [-F ... 0 ... +F]

after the FoxLima filter

- `N` = `int` # of Fourier bins = *Np*
- `p` = `int` oversampling ratio
- `rms` = `double` signal rms

Constructor (public)

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

command = `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: $1.472 < \lambda_{-20} / \lambda_{-3} < 4.684$
$2f_0 \Delta t = f_0 / FSR$ = chirp coherence-frequency
σ_{chirp} = chirp coherence-length # bins

case		
	F0, F1, F2, F4, F8, F16 raw	arg = void
	F0, F1, F2, F4, F8, F16 smooth	arg = λ_{-3}, λ_{-20}
	GX raw	arg = σ
	FC raw	arg = $2f_0\Delta t$, σ_{chirp}
	FX raw	arg = $2f_0\Delta t$, σ_{chirp}
	FE raw	arg = $2f_0\Delta t$, τ_{chirp}
	FW raw	arg = $2f_0\Delta t$, σ_{chirp}
	GX smooth	arg = σ, λ_{-3}, λ_{-20}
	FC smooth	arg = $2f_0\Delta t$, σ_{chirp}, λ_{-3}, λ_{-20}
	FX smooth	arg = $2f_0\Delta t$, σ_{chirp}, λ_{-3}, λ_{-20}
	FE smooth	arg = $2f_0\Delta t$, τ_{chirp}, λ_{-3}, λ_{-20}
	FW smooth	arg = $2f_0\Delta t$, σ_{chirp}, λ_{-3}, λ_{-20}

Parameters

- $f_n = (n-Np/2) / Np / T$ ..... frequency of bin n
- $A_n = \sqrt{(ddr_n^2+ddc_n^2)} / Np$ ..... amplitude of bin n (Cooley-Tukey)
  $\sqrt{(fzr_n^2 + fzc_n^2)} / Np$ ..... - - ‘’ - - (FoxLima)

Resolution

- $df / f = 0.7 / int(f T)$ ..... frequency resolution
- dA / A = from i/p ..... amplitude resolution

Description

The `fxl 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, with a frequency oversampling ratio of $p = 2^q / N$ (recommended ×8 - ×16). The **oversampling** solves the imprecision of the Cooley-Tukey algorithm, which for frequencies (periodic in the spectrum) has large errors, missing their peaks in:

- frequency, by: $\delta f < 1/pN\Delta t$,
- amplitude, by: $\delta A < 2pA/\pi$,
- phase, by: $\delta\varphi < \pi/2p$.

The native Cooley-Tukey algorithm has p = 1, hence the errors can be large.

Also, since the Cooley-Tukey algorithm has -6 dB/oct fall-off spectrum leakage tails, for certain peaks (with f = n / T), the Gibbs side-lobes are not directly visible as the sampling occurs exactly at the position of their zero's (for all other frequencies they are visible in various degrees). Metrologically, the favourable case when the Gibbs lobes are not visible is not a true representation of algorithm's performance, as the spectrum leakage is still present.

F1 – is a smoothing-filter to the Cooley-Tukey algorithm. It solves the problem of the Gibbs satellite-lobes, but has still spectrum leakage tails, to the effect of -6 dB/oct fall-offs. Its advantage is speed, requiring little CPU time with respect to other filters.

F2 – is a Cooley-Tukey apodisation method (with or without smoothing-filter) with the best trade-off between peak width and suppression of the Gibbs side-lobes. It has -12 dB/oct spectrum leakage fall-off. Being performed in Fourier-space, its spectrum-leakage 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.

F4 – is an advanced version of F2, with -24 dB/oct spectrum leakage fall-off and wider peaks. Spectrum-leakage 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 (like F2) performed in Fourier-space rather than in temporal space, hence exact.

F8 – is an higher Gibbs lobe rejection version of F4, with -48 dB/oct spectrum leakage fall-off. It is a Welch-8 apodisation, performed in real-space. It is susceptible to spectrum distorsions due to inexact cancellations of digitisation terms when the signal has large phase noise, or it represents a short temporal sequence.

F16 – it is an advanced version of F8, with -96 dB/oct spectrum leakage fall-offs. It is a Welch-16 apodisation, performed in real-space. It is susceptible to spectrum distorsions due to inexact cancellations of digitisation terms when the signal has large phase noise, or it represents a short temporal sequence.

GX – is a user customised version of F8 and F16 with gaussian apodisation (of user given σ) in real-space.

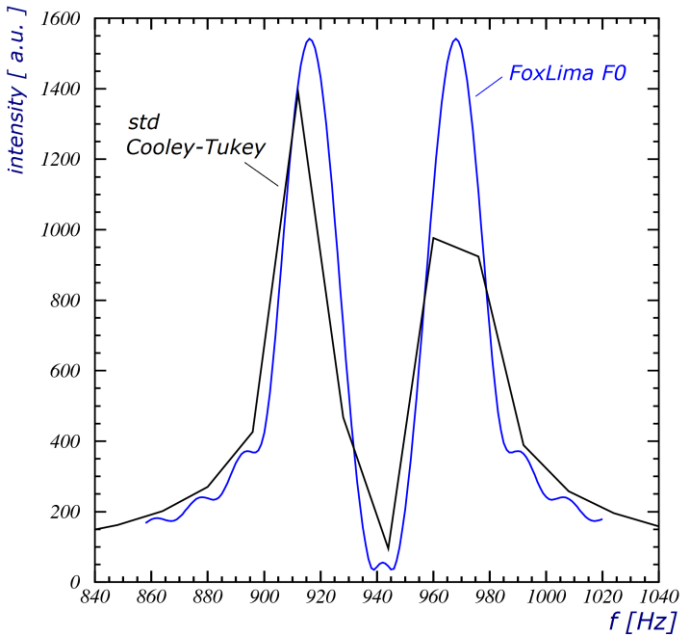
NOISE SIGNALS

The signal below is composed of two (coherent) chirps, at a certain delay one after the other. They have the same frequency and it would seem they should also have a very well defined FFT-peak in the spectrum.

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, a wide set of noise-sources emit "chirps" (each with a valid coherent FFT-spectrum), however at random time intervals (thus same $\Delta\varphi/\omega$ for all frequencies).

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



The problem with this procedure is the poor resolution. For any FFT the thinness of the peaks is given by the sum:

$$\Sigma_{k=1,N-1} \exp(2\pi i(\omega-\omega_0)kT)$$

which gives an "Airy"-function of frequency resolution $\delta f = 1 / NT$.

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

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

The apodisation methods available are:

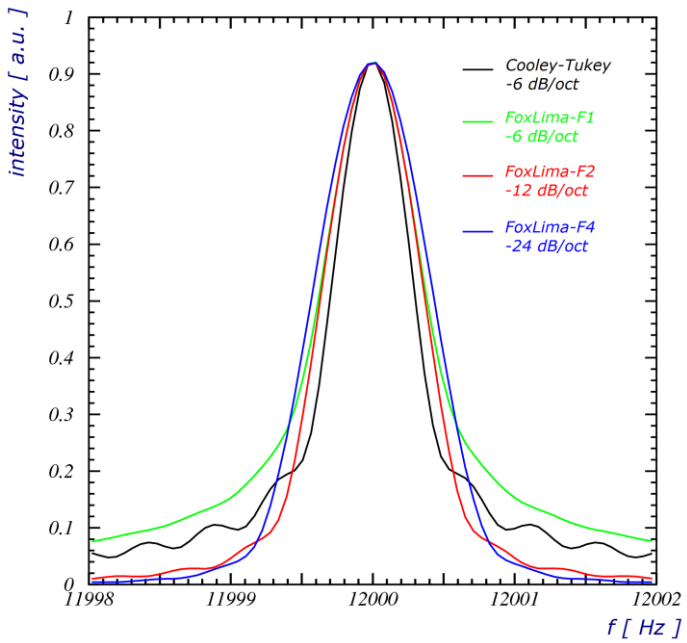
- `FC` – without windowing of the smaller samples. User given σ and f_0.
- `FX` – gaussian apodisation of the smaller samples. User given σ and f_0.
- `FE` – back-exponential apodisation of the smaller samples. User given τ and f_0.
- `FW` – Welch apodisation of the smaller samples. User given σ and 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
   - $\Delta\varphi = 90^o$ in phase

   These aspects can be seen in the figure to the right and the figure below, where the sub-sampling problems of the Cooley-Tukey algorithm are evidentiated as peak truncation, central-frequency and phase imprecision.



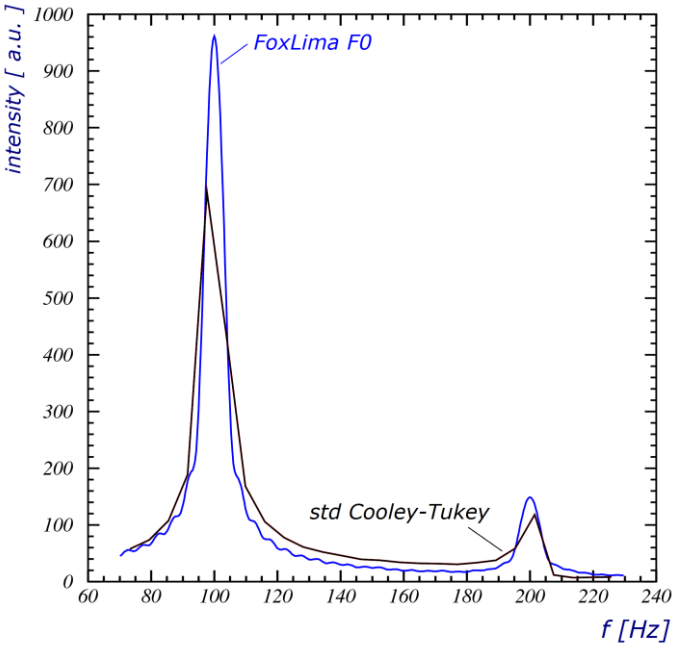
2. *exact leakage elimination* – due to its Fourier-space apodisation, algorithms F1, F2 and F4 eliminate exactly the $1 / (f-f_0)^k$ spectrum-leakage tails for:
   - k = 0 (F1 algorithm)
   - k = 1 (F2 algorithm) and
   - k = 1,2,3 (F4 algorithm).

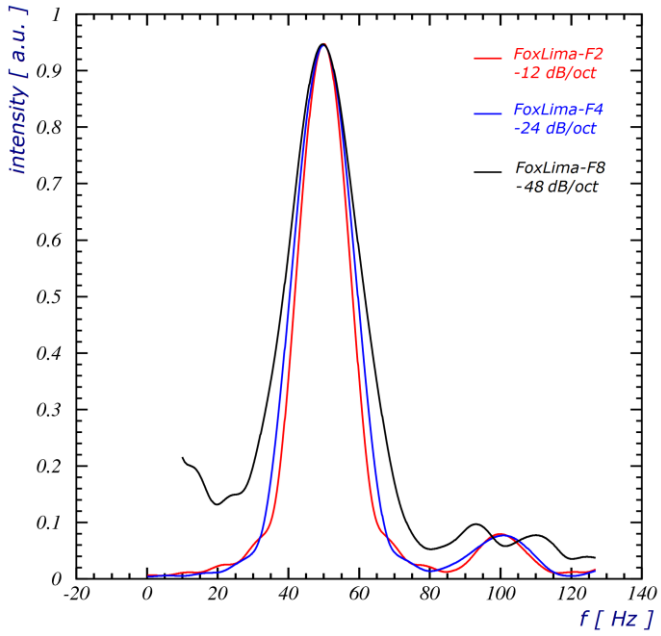
   The figure above shows smoothed versions of the algorithms for width and tail fall-off.

   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).

   It is important to note, that real-space apodisation does not take into account discretisation components exactly, as does the 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- peak, whereas F2 and F4 are both very precise.

3. *Weyl-Wigner FFT* – as mentioned above, random-chirp signals may deceivingly 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 can be:
   - **FC** – plain Cooley-Tukey
   - **FX** – gaussian
   - **FW** – Welch
   - **FE** – back-exponential

   The apodisation used in the figure here is gaussian.