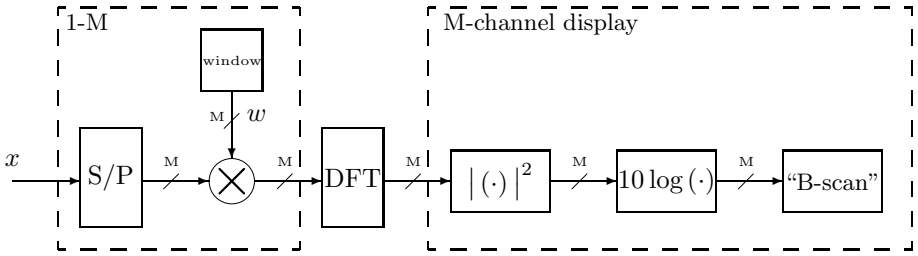
Spectral processors and displays

Niten Olofsson

October 24, 2013

Overview

A conceptual description of the different steps, different processor elements for the processor graph, and implicitly also what classes to create follows.



The following steps are performed: blocking up the time series/serial-to-parallell (S/P), windowing each block (multiplication by $\{w_k\}_{k=0}^{M-1}$), computing the discrete fourier transform of the windowed block (DFT), taking the modulus square ($|(\cdot)|^2$ or energy) and finally express the result in decibels ($10\log_{10}(\cdot)$ or dB).

1-M

This module buffers M elements, then shifts N_{overlap} for every step and outputs M elements in M channels, i.e., 1 element per channel. Prior to being output the elements are multiplied by a window.

S/P

The buffer will have an option of overwriting old elements in a cyclic, cyclic buffering. The default behaviour is to shift the content of the buffer N_{overlap} elements.

The cyclic buffering is useful for the case when we are only interested in the magnitude of the power spectrum. The shift introduced by the cyclic buffering will only introduce a phase change in the DFT bins according to the basic theorems for Fourier series, and when taking the magnitude of a complex number, the phase information will be lost due to the complex number being multiplied by its conjugate.

Window

The content of the buffer will be multiplied by a window. Default is a rectangular, or no, window.

DFT

The module calculates an M-point DFT and outputs the result in M parallell channels. The FFT algorithm is used.

M-channel display

The display module takes M channels as input, and display the modulus of each channel in parallell. On can also sum over a couple of time steps, if one wants to remove the effects of transient and mainly wath for long-term behaviour of the data. Since I do not know the measurement needs well enough I leave this as an option and you ephys guys can use it if you want to.

$\left|\,(\cdot)\,\right|^2$ or energy

Calculates the "energy" of its input. Can sum over a chosen number of indices, N_{energy}, and possibly average, therefore introducing a delay of N_{energy}. In the spectrogram, $N_{\text{energy}} = 1$.

$10\log_{10}(\cdot)$ or dB

Computes the dB value from intensity input. Optional is to multiply by 20 instead, if amplitude and not intensity data is given.

"B-scan"

The "B-scan" can either insert new data on to the right and scroll older data to the left (default), or it can use the waterfall display mode where the new data enters on top and is vertically scrolled down in the display. In radar and sonar processing, this is almost a B-scan (there are some details which different vendors and armed forces cannot agree upon, therefore the usage of almost so not to start a flamewar over an insignificant choice of words.).

Interpolation

In a first attempt, after consulting a developer of Sonic AWE, I suggest that a nearest-neighbour approach is used when the number of points/pixels of a line in the display is not an integer number of M ("FFT-bins"). This will be the fastest, and since minimum delay is essential in the final version, interpolation is one of the steps that might need some significant number crunching. If nearest neighbour suffices, we should stick to that.

Why an M-channel display and not a spectrogram-display?

There are other alternatives than just displaying spectral data. For instance, if one has many channels and just want to verify that they are working, one can calculate the energy for each

channel, and then look at energys in parallell, if one has some saturated and/or defunct channels, the will immediately stand out, whereas 32 LFPs can be very difficult to monitor simultaneously.

Some issues not handled yet

I have not delved too much into the matter of DFT-size independant intensity calculations, should these calculations be in the energy module or in the DFT module? In this first iteration, that will not be dealt with, but as soon as something is up and running, those issues will be dealt with. The size and complexity of the spectrogram is not greater than one can just rewrite the whole thing after changing the architecture if need be, my experience is that the second and third design and coding go much much faster than the first attempt.

Classes

In a first iteration, I suggest that the three main modules are implemented independantly, part from the energy class which for testing purposes of the "B-scan" display could be useful as a processor of its own; it is very easy to display M channels in the M-channel display and in advance determine what the energy should be and how the display should look like, in contrast to spectral data. On the other hand, one can use SciPy or matlab spectrograms to compare with.

M-channel stream

I have not delved into the details of open-ephys regarding splitter and joiners, will do that now, by implementing the S/P module.