Software De-dispersion for RTL SDR Pulsar Data

Introduction

Data files produced by RTL SDR dongles can be folded directly for pulsar detection using software such as *rapulsar.exe*. Using simple I/Q vector averaging software, the data can be down-sampled by factors of more than 100 prior to folding and/or period search processing to speed up useful data extraction. Ideally, wide band RF data should be de-dispersed to optimise later search and folding processing. De-dispersion is normally carried out by time adjusting data sampled from RF filter banks before combination. This note describes how data already digitised from the RTL SDR can be spectrum analysed or filtered using the FFT algorithm. Two methods are discussed, one summing power with some down-sampling; the second, a 'coherent' method that de-disperses the *rtlsdr.exe* .bin data file and outputs a .bin-compatible file. Both accurately de-disperses the data offering an improved folded data SNR.

Pulsar Dispersion Modelling

The broadband pulsar noise radiation interacts with free electrons in the interstellar medium, causing lower frequencies to be delayed compared to the higher frequency components.

De-dispersion processing compensates for the delays and is carried out using time shifts of filtered sub-bands followed by recombining to increase the observable pulse power. A basic Gaussian pulse shape approximation shows that the power gain with de-dispersion is minimal for pulsars with band delays smaller than half the pulsar pulse width but is much more important for detectability with larger RF bandwidths. The following table based on simple 4/8-element models illustrates this point.

Dispersion/ Pulse Width	Increase in Observed Width	Power Gain dB
0.25	-	0.05
0.5	1.6	0.22
0.75	1.12	0.51
1	1.24	0.88
2	2.0	2.95
4	2.7	5.75

The modelling also shows that de-dispersion sharpens the centre of the pulse and the reduction of the breadth of the base is more evident than the reduction of the half-height width. Pulsar B0329 at ~400MHz having a dispersion, about equal to the pulse width over an RF bandwidth of 2MHz is a good example where de-dispersion can be expected to give up to a 25% improvement of SNR.

De-dispersion Processing with an FFT - 'Total Power'

The process considered, accepts RTL SDR .bin files and performs a series of FFTs, then re-organises the data in frequency and delay time, before band-combining and outputting the dispersed data file, suitable for period-folding and SNR integration. The process is depicted in Figure 1.

A four-point FFT is shown for clarity although the number of points N determines the data down-sampling ratio.

The delay blocks are clocked at N times the input data clock period t_f and the delays are given in numbers of FFT blocks by, $nT_{DM}/N/N/t_f$, where $n = 0$ to N-1 representing the FFT frequency bins, and T_{DM} is the dispersion measure time calculated for the

whole received RF bandwidth. For example, for a 2MHz data bandwidth, T_{DM} in μs, the real time delays for a four-channel RF filter bank are, 0, $T_{DM}/4$, $T_{DM}/2$ and $3T_{DM}/4$, and for a 4-point FFT, the delays in terms of FFT blocks (2μs) are , 0, $T_{DM}/4/2$, $T_{DM}/2/2$ and $3T_{DM}/4/2$.
Continuing this 2MHz data example, for N-point FFTs, the bin delays in FFT blocks, becomes, $2nT_{DM}/N/N$, which, obviously imposes a limit on the FFT size, given the RF dispersion time. In fact, above a certain size, increasing FFT length does not significantly improve the pulse compression but may impact on computation time.

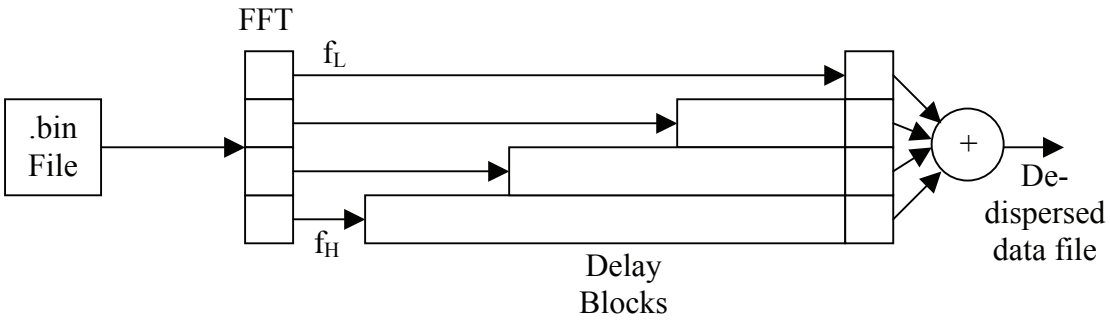

Figure 1 De-dispersion Principle

Figure 1 demonstrates how the higher frequency components can be delayed to match the low frequency component arrival times and so sum powers to optimise the received SNR. In practice, the software implementation uses rolling arrays to accomplish the same effect as described in the following section.

De-dispersion Software Process - total power
The software process comprises 3 sections:

1. Initialization
i. Calculate pulsar pulse dispersion time from the RF and pulsar Dispersion Measure,
ii. Calculate FFT frequency bin-required delays in terms of FFT data blocks (= N x input data clock).
iii. Place values in the *delay array.*

2. Data-in and calculations
i. Read N I/Q data pairs.
ii. Perform N-point FFT.
iii. Re-order FFT data to compensate for the RTL zero-IF aliasing.
iv. Repeat sequence 2.

3. Data de-dispersion and output
i. Place FFT bin amplitude data in a 4096xN array addressed offset by the each required bin delays as calculated in the *delay array*.
ii. Once the array is filled, sum time-aligned FFT data and place in the output file.
iii. Continue to fill a second identical array and on roll-over, refill the first array from the start.
iv. Once the second array is filled, sum time-aligned FFT data and place floating point data in the output text file.
v. Repeat step iii to end of file.

Results - total power de-dispersion

Figures 3 and 4 show the results of software FFT de-dispersion of 100MB blocks from the start and end of the Pulsar B0329 data file, *dump2000a.bin*, recorded by M Klaassen using an RTL2832U SDR receiver tuned to 419MHz behind the Dwingeloo 25m radio telescope on 14/01/2016. The file contained 400M samples and the figure labels define the data used. Both figures show the results of data folding using 1001 bins to show fine detail.

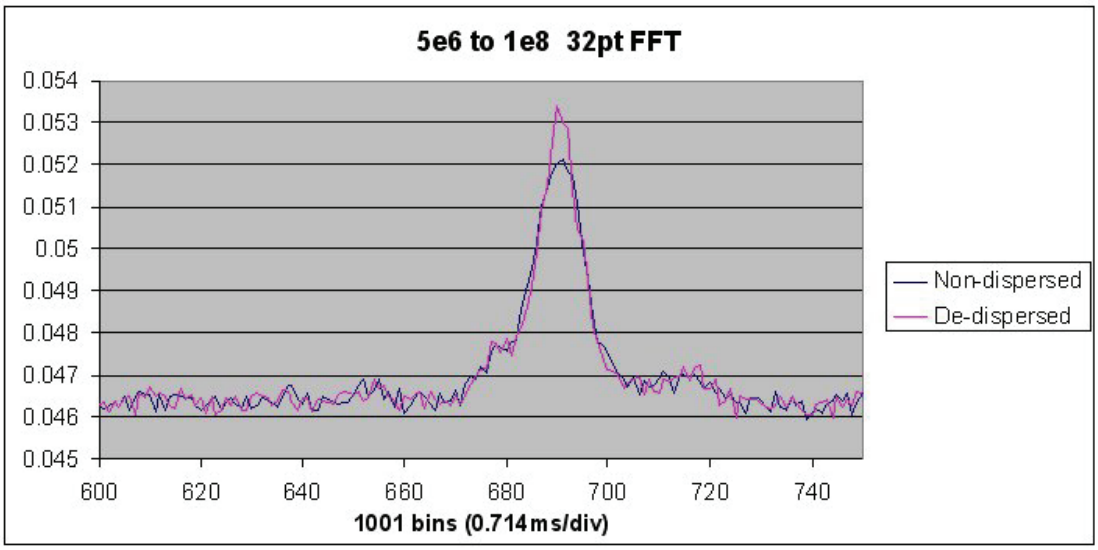

Figure 2 De-dispersion based on a 32-point FFT

SNR improvement results in the four sections of the raw data file reviewed, varied from 0.22 to 1.0dB, which is in line with the model predictions. B0329 scintillation and jitter was evident in both amplitude and time-of-arrival.

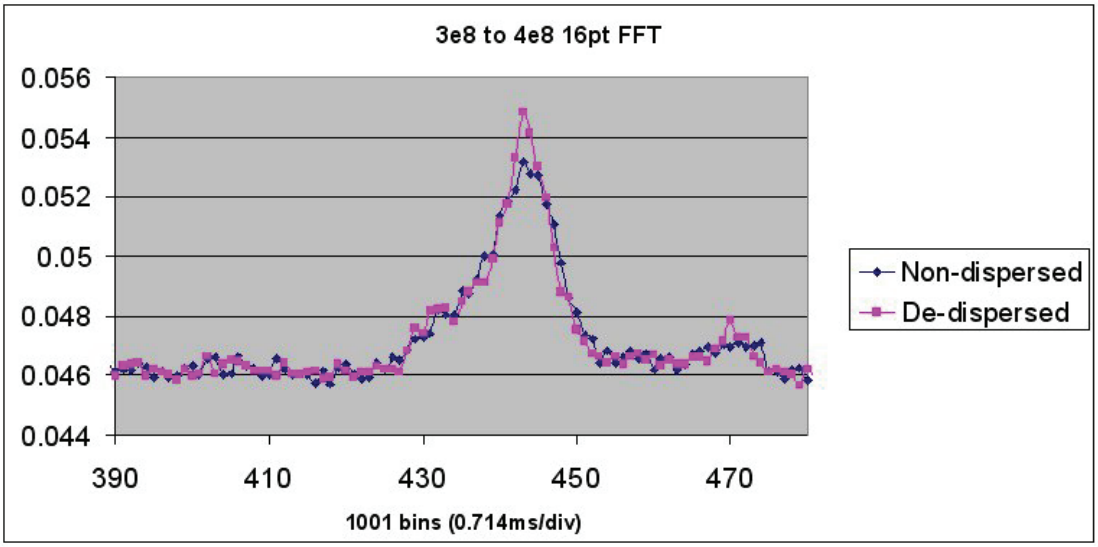

Figure 3 De-dispersion based on a 16-point FFT

De-dispersion Software Process - 'coherent'
The software process again comprises 3 sections and outputs a binary file in the *rtlsdr.exe* format, slightly smaller in size due to the incomplete delayed records being rejected. The file is time-synchronised with the non-dispersed input file for easy comparison.

1. Initialization
i. Calculate pulsar pulse dispersion time from the RF and pulsar Dispersion Measure,
ii. Calculate FFT frequency bin-required delays in terms of FFT data blocks (= *N* x input data clock).
iii. Place values in the *delay array.*

2. Data-in and calculations
i. Read *N* I/Q data pairs.
ii. Perform *N*-point FFT.
iii. Re-order FFT data to compensate for the RTL zero-IF aliasing.
iv. Repeat sequence 2.

3. Data de-dispersion and output
i. Place FFT bin I and Q data in a 4096x2*N* array addressed offset by the each required bin delays as calculated in the *delay array*.
ii. Once the array is filled, access time-aligned FFT data, re-alias spectrum data, take inverse FFT and place as unsigned characters in the output .bin file.
iii. Continue to fill a second identical array and on roll-over, refill the first array from the start.
iv. Repeat steps ii -> iii to end of file.

Results - power v coherent de-dispersion
Figure 4 compares the performance of both total power and coherent methods of de-dispersion on the central 200MB part of the Pulsar B0329 data file, *dump2000a.bin*. As before, the original file data is shown in the blue trace. Total power (*de-dispers2* software) is shown in magenta and overlaid in yellow is the coherent de-dispersion result (*de-dispers2Co* software). The only obvious difference is a slight DC offset, probably due to floating-point FFT rounding when converted to integer/binary.

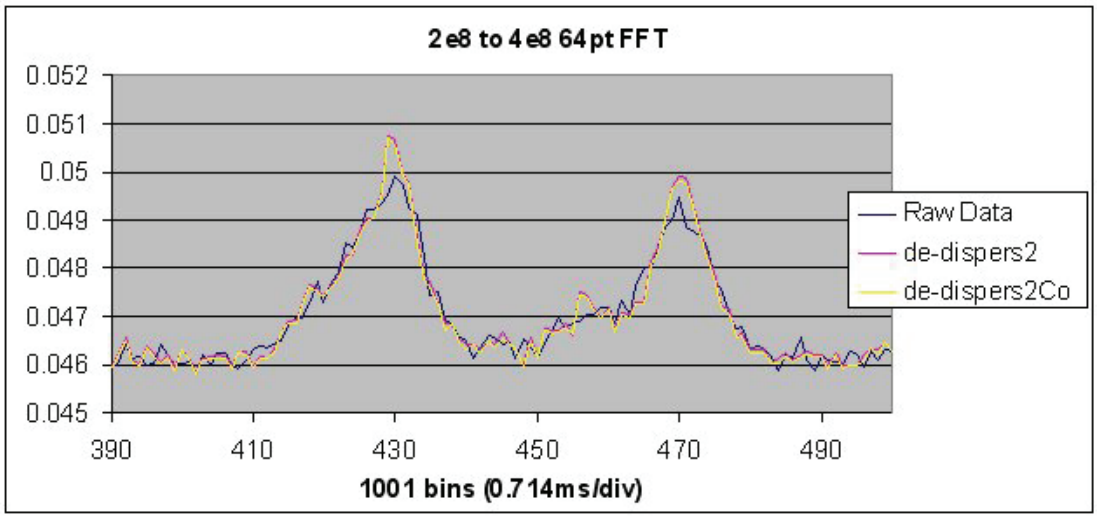

Figure 4 'Power' v 'Coherent De-dispersion Comparison

The presence of two pulses in this record appears to be a real pulse position jump in the pulsar data. It is interesting to note that both are similarly compressed in the de-dispersion process.

Discussion

Even for B0329, where the dispersion/half-height pulse width ratio is about unity a sharpening of the pulse is evident coupled with a small noticeable SNR enhancement consistently, in separate sections of the raw data file. As well as SNR improvement, de-dispersion offers a better indication of pulse shape and variations over time.
Early experiments on 1420MHz Vela data were inconclusive although significant improvement can be expected at lower RF frequencies where the much larger DM can cause dispersion equivalent to multiple pulse widths.
Since both de-dispersion methods appear to produce similar results after synchronous integration, the chosen software is a personal choice. The total power version requires a modified *rapulsar.exe* tool to cope with text file output from d*e-dispers2* but does offer some data compression dependent upon the FFT length chosen. The 'coherent' version may be preferred as it outputs an accurately de-dispersed input-compatible file (from *rtlsdr.exe*) useful for driving other custom software directly, such as *rapulsar2.exe.*

Conclusions

This exercise has shown that post-processing of raw pulsar data to de-disperse the pulsar signals can give a useful enhancement of SNR and improve observability and detectability.

Software Tools - (http://www.y1pwe.co.uk/RAProgs/de-disperscv2.zip)

De-dispersion Tool

Format: *de-dispers2* <infile> <outfile> <clock rate (MHz) <fft points> <RF centre(MHz)> <dispersion measure(DM)>

Accepts *rtlsdr* .bin files, takes *N*-point FFT, de-aliases data, de-disperses frequency components in time, sums aligned FFT bin powers, ready for folding by *rapulsar2_avg2* using output down-sampled clock rate. Outputs a text file of dispersion corrected power data at the initial clock rate, reduced by a factor equal to the number of FFT points. Negative DM used for odd sideband in superhet receivers.

Synchronous Folding Tool

Format: *rapulsar2_avg2* <infile> <outfile> <clock rate (MHz)> <No. of output data points> < pulsar period(ms)>

Accepts *de-dispers2* .txt files, applies averaging and folding algorithm in blocks. Outputs text file of folded average. Also accepts *pdetect2* output .txt files.

'Coherent' De-dispersion Tool

Format: *de-dispers2Co* <infile> <outfile> <clock rate (MHz) <fft points> <RF centre(MHz)> <dispersion measure(DM)>

Accepts *rtlsdr* .bin files, takes *N*-point FFT, de-aliases frequency data, de-disperses frequency components in time, re-aliases data, takes inverse FFT, outputs binary data in rtlsdr.exe format ready for folding by *rapulsar2.* Outputs a .bin file of dispersion corrected sampled data at the initial clock rate, slightly smaller in size due to incomplete delayed sets, but still in time synchronism with the input file. Negative DM used for odd sideband in superhet receivers. DM=0, no de-dispersion.

Binary File Trimming Tool

Format: *filetrim* <infile> <outfile> <Start Bytes> <End Bytes>

Accepts .bin files, cuts command number 'Start Bytes' from front and command number 'End Bytes' from beginning and end of the input file respectively and outputs a binary file of *rtlsdr* .bin-compatible data.