EXPLORING THE DYNAMIC RADIO SKY WITH MANY-CORE HIGH-PERFORMANCE COMPUTING

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2018

By
Mateusz Malenta
School of Physics and Astronomy

Contents

Declaration **9**

Copyright **10**

Publications **11**

Acknowledgements **12**

1 Introduction **15**

1.1 Pulsars 15

1.1.1 Key Discoveries 16

1.1.2 Pulsar Model 17

1.2 Rotating Radio Transients 20

1.3 Fast Radio Bursts 21

1.3.1 Current Theories 24

1.4 Propagation Effects 28

1.4.1 Dispersion 28

1.4.2 Scattering 31

1.5 Thesis Outline 33

2 Signal Processing **34**

2.1 Radio Telescope Basics 34

2.2 Candidate Detection 36

2.2.1 Dedispersion 36

2.2.2 Periodic Signals Detection 38

2.2.3 Single-pulse Detection 41

2.3 Phased Array Feeds 42

2.3.1 Theoretical Description of Beamforming 45

2.3.2 Practical Beamforming Procedure 47
2.4 GPU and Parallel Computing 48
2.4.1 Compute Unified Device Architecture - CUDA 51

3 PAFINDER - Phased Array Feed FRB Finder 54
3.1 PAF at Effelsberg 54
3.1.1 Processing Facilities 55
3.2 PAFINDER Pipeline Design and Implementation 56
3.2.1 Receiving the Data 59
3.2.2 Unpacking 63
3.2.3 Filterbank 63
3.2.4 Scaling 65
3.2.5 Detection 69
3.3 Deploying on the Cluster 70
3.4 Tests and Benchmarks 75
3.4.1 GPU Kernels Optimisation 75
3.4.2 Data Quality Tests 85

4 PAFINDER Observations 91
4.1 Metadata 91
4.2 System Temperature Measurements 92
4.3 RRAT J1819 – 1458 Observations 96
4.3.1 Single-pulse Detections 97
4.3.2 Timing and Profile Evolution 98
4.4 Fast Radio Bursts Observations 106
4.4.1 FRB121102 108
4.4.2 FRB110523 and FRB130729 113

5 Single Pulse Simulations 114
5.1 Motivation for Simulating FRBs 114
5.1.1 Simulation Framework 115
5.2 Simulation Overview 117
5.2.1 Voltage Generation 117
5.2.2 Dispersion 120
5.2.3 Burst Injection 127

6 GHRSS survey **128**
6.1 Survey Description 128
6.1.1 Survey Limits 130
6.2 Processing Pipeline 134
6.2.1 Preprocessing and Distribution on the Cluster 134
6.3 Bifrost Code Overview 137
6.3.1 Dedispersion 138
6.3.2 Pulsar Search 141
6.3.3 Single-pulse Search 144
6.4 Results 145
6.4.1 Known Sources Detections 146
6.4.2 Simulation Results 147
6.4.3 FRB Search Results 152

7 Conclusions **155**
7.1 New GPU Processing Pipeline for Effelsberg PAF 155
7.2 Fast Radio Bursts Simulations 156
7.3 GHRSS Survey 157
7.4 Future Work 158

Word Count: 47414

List of Tables

3.1 Specification of PAF processing nodes 56

4.1 Mean and median system temperatures for beams used during observations 95
4.2 Sources observed during PAF January 2018 session 107
4.3 Number of single-pulse candidates during each FRB121102 pointing . 109

6.1 GHRSS survey parameters 129
6.2 Specification of GHRSS processing nodes 134

List of Figures

1.1 A simple pulsar model 18
1.2 The distribution of Fast Radio Bursts on the sky 22
1.3 Dispersion comparison 30
1.4 Scattering mechanism 31

2.1 Simple heterodyne receiver system 35
2.2 Stages in the periodic signal detection 39
2.3 PAF signal path 45
2.4 Application of the beamforming procedure 49
2.5 CPU and GPU systems 50
2.6 Concurrent execution of CPU and GPU code 52
2.7 Concurrent execution of GPU kernels 53

3.1 The overview of PAF frontend and backend 55
3.2 Basic components of the PAFINDER pipeline 57
3.3 Buffers used for different stages of the processing pipeline 58
3.4 Header of the CODIF data frame 59
3.5 Packing of the data into the CODIF frame 60
3.6 Data representation after the unpacking 63
3.7 The effect of scaling on recorded signal 66
3.8 Example single-pulse detections plot 71
3.9 Additional PAFINDER candidate plots 72
3.10 Pipeline monitoring software 74
3.11 Unpack kernel benchmarking 78
3.12 Power kernel benchmarking 80
3.13 Scaling factors kernel benchmarking 82
3.14 B0355+54 data as recorded using the real-time pipeline 83
3.15 Real-time processing time comparison 84

3.16 Network performance tests results 86
3.17 Tone sweep test results 87
3.18 Comparison of the data scaled using different scaling techniques . . . 89
3.19 Comparison of single-pulse detections 90

4.1 Central beam on-source and off-source powers 93
4.2 System temperature for the inner and outer ring of beams 94
4.3 Single-pulse detections towards J1819 – 1458 97
4.4 Example of two confirmed single pulse detections 99
4.5 Basic derived properties of J1819 – 1458 100
4.6 Timing residuals of J1819 – 1458 101
4.7 Selection of J1819 – 1458 pulses 102
4.8 A group of multi-compoment detections 103
4.9 Single-pulse groups separation 105
4.10 FRB sources altitude 106
4.11 High-S/N FRB121102 candidate 108
4.12 Single-pulse detections towards the position of FRB121102 111

5.1 Simulated FRB voltage 119
5.2 Example of the semi-coherent dispersion 124
5.3 Semi-coherent dispersion burst comparison 125
5.4 Dispersion methods workflow 126

6.1 GHRSS pulsar sensitivity 131
6.2 GHRSS FRB sensitivity 132
6.3 GHRSS pipeline workflow 135
6.4 Dedispersed time series statistics 140
6.5 Comparison of properties of dedispersed time series using different DM values 142
6.6 Scaled dedispersed time series values distribution 143
6.7 Comparison of folded profiles for two pulsar candidates 145
6.8 Bifrost pulsar redetections 148
6.9 Single pulse detection outputs for known pulsars 149
6.10 Simulated FRBs detected by the pipeline 150
6.11 GHRSS pointings 151
6.12 Modelled FRB burst rate 153
6.13 Predicted spectral index values 154

The University of Manchester

ABSTRACT OF THESIS submitted by Mateusz Malenta
for the Degree of Doctor of Philosophy and entitled
"Exploring the dynamic radio sky with many-core high-performance computing"
June 2018.

As new radio telescopes and processing facilities are being built, the amount of data that has to be processed is growing continuously. This poses significant challenges, especially if the real-time processing is required, which is important for surveys looking for poorly understood objects, such as Fast Radio Bursts, where quick detection and localisation can enable rapid follow-up observations at different frequencies. With the data rates increasing all the time, new processing techniques using the newest hardware, such as GPUs, have to be developed.

A new pipeline, called PAFINDER, has been developed to process data taken with a phased array feed, which can generate up to 36 beams on the sky, with data rates of 25 GBps per beam. With the majority of work done on GPUs, the pipeline reaches real-time performance when generating filterbank files used for offline processing. The full real-time processing, including single-pulse searches has also been implemented and has been shown to perform well under favourable conditions. The pipeline was successfully used to record and process data containing observations of RRAT J1819 − 1458 and positions on the sky where 3 FRBs have been observed previously, including the repeating FRB121102. Detailed examination of J1819 − 1458 single-pulse detections revealed a complex emission environment with pulses coming from three different rotation phase bands and a number of multi-component emissions. No new FRBs and no repeated bursts from FRB121102 have been detected.

The GMRT High Resolution Southern Sky survey observes the sky at high galactic latitudes, searching for new pulsars and FRBs. 127 hours of data have been searched for the presence of any new bursts, with the help of new pipeline developed for this survey. No new FRBs have been found, which can be the result of bad RFI pollution, which was not fully removed despite new techniques being developed and combined with the existing solutions to mitigate these negative effects. Using the best estimates on the total amount of data that has been processed correctly, obtained using new single-pulse simulation software, no detections were found to be consistent with the expected rates for standard candle FRBs with a flat or positive spectrum.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Publications

B. Bhattacharyya, S. Cooper, **M. Malenta**, J. Roy, J. Chengalur, M. Keith, S. Kudale, M. McLaughlin, S. M. Ransom, P. S. Ray and B. W. Stappers, *The GMRT High Resolution Southern Sky Survey for Pulsars and Transients. I. Survey Description and Initial Discoveries*, ApJ, 2016, 817, 130.

X. Deng, A. P. Chippendale, G. Hobbs, S. Johnston, S. Dai, D. George, M. Kramer, R. Karuppusamy, **M. Malenta**, L. Spitler, T. Tzioumis, G. Wieching, *Observing Pulsars with a Phased Array Feed at the Parkes Telescope*, PASA, 2017, 34, E026

Acknowledgements

First of all I would like to thank my supervisors, Dr Michael Keith and Prof Ben Stappers for giving me opportunity to work on this project and for making it possible by providing continuous support, encouragement and ideas that made this project better.

The PAFINDER project would not have been possible without a group of people who helped me develop, fix and optimise this complicated piece of software. I would especially like to thank Prof Michael Kramer for a chance to work on this project over the years and the incredible amount of patience when major setbacks were met and deadlines were missed. A big thank you to Ewan Barr for his help and infinite knowledge of GPU computing and amazing optimisation techniques, Xinping Deng for his help with tests, identifying and solving problems with the pipeline and invaluable pulsar observations and Aaron Chippendale for helping me understand the inner workings of phased array feeds. I would also like to thank every single person who contributed to designing, building, testing and running the PAF and the pipeline, a whole army of people working at MPIfR, CSIRO and JBCA. Thank you, good luck and I am sorry to anyone who will have to maintain this pipeline in the future.

I would also like to thank Bhaswati Bhattacharyya and Jayanta Roy for giving me a change to work on the GHRSS project and their help and suggestions that let us improve the processing and a thank you to Prabu Thiagaraj for great advice and help on the nuances of the digital signal processing that made the development of the single-pulse simulation software much easier. A huge thanks to Anthony Holloway and Robert Dickson for solving the problems with our processing nodes, especially in the last few months of the project, when things were breaking on a daily basis. I would also like to thank all of the members of our pulsar group, especially my fellow PhD students I have shared the office with over the last 3.5 years for not kicking me out.

I would like to thank my parents for their encouragement and support over the years and Beth for helping me with my life and making the final months survivable, especially for playing the game of 'single, double, triple-component emission' at 2am.

I dedicate this to everyone who helped me make this project a reality.

Dude, suckin' at something is the first step towards being sorta good at something

Jake the Dog

C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off

Bjarne Stroustrup

Chapter 1

Introduction

1.1 Pulsars

When a star reaches the end of the main sequence stage, its ultimate fate is decided by its mass. The least massive stars, including our Sun are expected to go out in a relatively uneventful way, leaving a white dwarf behind. More massive stars however end their lives in a more spectacular fashion - a supernova. What is left behind can be a black hole, expected to form only when the most massive stars, with masses above $15-20\,M_{\odot}$ collapse. When the initial mass of the star lies somewhere in between the lightest and the most massive objects, usually between 4 to 10 solar masses, a neutron star can be formed. In such a scenario, the outer parts of the star are blown away during the explosion, with only the central core remaining. This core is supported against collapse by the neutron degeneracy pressure and is not massive enough ($1.4-3\,M_{\odot}$) to contract down further into a black hole. It is also much smaller than the star that produced it, and has the diameter of around 20 km (slightly less than the distance between the Jodrell Bank Centre for Astrophysics and the Jodrell Bank Observatory), which makes neutron stars one of the densest object in the known Universe. The exact masses and compositions of neutron stars are dependent on the equation of state, which is an active area of research at the moment. It is however believed that they are made out of predominantly neutrons at larger depths and a mixture of electrons, neutrons and possibly heavy nuclei, such as iron, close to and at the surface.

As the result of the conservation of angular momentum, neutron stars spin up rapidly while shrinking, and can reach rotation rates up to hundreds times per second by the time the collapse stops. Even though the main contributor, the conservation

of angular momentum on its own cannot explain the extent of different pulsar periods. If all the angular momentum was conserved and no energy-loss mechanisms were present, the core remaining after the supernova explosion would be expected to have rotational periods of the order of 1 ms, inconsistent with the observed spin periods of the 'ordinary' pulsars, which are found in the range of 10s – 1000s milliseconds. Part of the angular momentum can already be lost during the pre-supernova red supergiant expansion phase, due to emission of strong stellar wind (Cerdá-Durán & Elias-Rosa, 2018). During and right after the supernova explosion, considerable fraction of angular momentum is lost due to neutrino emission and energy radiated in the form of gravitational waves (Camelio et al., 2016) and the interaction between the neutron star and the supernova ejecta (Cerdá-Durán & Elias-Rosa, 2018).

1.1.1 Key Discoveries

Neutron stars remained a theoretical concept for more than three decades, since they were first proposed in 1934 (Baade & Zwicky, 1934). This changed in 1967 with the discovery of the first pulsar by Antony Hewish and Jocelyn Bell. A year later, in 1968, a radio source with a period of 33 milliseconds was found inside the Crab Nebula. This pulsar was also observed in the region close to the well-known supernova, SN 1054 (Reifenstein et al., 1969), confirming the association with the final stages of stellar evolution. A short observed period confirmed the prediction that pulsars were rotating neutron stars. Another model, considered at that time, involved white dwarfs as pulsar progenitors, but the short rotational period meant it was highly unlikely to be the case, as white dwarfs were expected to have periods of the order of minutes, hours or even days (Gold, 1968). The number of known pulsars is growing all the time and with more than 2500 objects discovered until now [1], the theory of rapidly rotating neutron stars still holds.

In 1974, the first pulsar in a binary system, B1913 + 16, was observed (Hulse & Taylor, 1975). At the time of the discovery, it was the second fastest spinning pulsar known, with a period of 59 milliseconds. The evidence for a companion object was present in the unexpectedly large changes in the pulsar's rotational period, which was reported to vary between 58.967 ms and 59.045 ms over the orbital period of 7.752 h (Hulse & Taylor, 1975). Due to the mass of the neutron star and its companion and their small separation, these objects were expected to emit gravitational radiation at a

[1] http://www.atnf.csiro.au/research/pulsar/psrcat

level allowing the measurement of the orbital shrinkage. The orbital period decrease was measured to be $(-2.30 \pm 0.22) \times 10^{-12}\,\mathrm{s\,s^{-1}}$ which was consistent with the prediction from Einstein's theory of general relativity at $(-2.403 \pm 0.005) \times 10^{-12}\,\mathrm{s\,s^{-1}}$ (Taylor & Weisberg, 1982). In 1993, Hulse and Taylor were awarded the Nobel Prize in Physics "for the discovery of a new type of pulsar, a discovery that has opened up new possibilities for the study of gravitation" (The Nobel Prize Committee, 1993).

Another breakthrough came in 1982, with the discovery of the first millisecond pulsar, B1937+21 (Backer et al., 1982). With a period of just 1.558 ms, it is still the third fastest spinning pulsar ever discovered (two faster millisecond pulsars: J1748−2446ad and J0952−0607 have the approximate periods of 1.39 ms (Hessels et al., 2006) and 1.41 ms (Bassa et al., 2017b) respectively). It became apparent that the original theory of the neutron star formation and evolution could not account for such a short period and small period derivative, with an upper limit of 10^{-15}, which was orders of magnitude smaller than for pulsars discovered until that point. Such small period derivative indicates a weak magnetic field strength (see Equation 1.2) associated with old pulsars, sometimes so old and with field strengths so low that they are expected not to radiate at all (they are below the so called 'death line'). Old pulsars are expected to have long periods, orders of magnitude longer than in the case of fastest millisecond pulsars, it was therefore proposed that the very fast rotation was a result of spinning up, due to the accretion of matter from a younger companion star and the transfer of angular momentum (Radhakrishnan & Srinivasan, 1982). Most of the currently known millisecond pulsars are members of binary systems, which supports this model of pulsar recycling.

1.1.2 Pulsar Model

The most basic model, which explains the observed properties of the pulsar radiation, includes a rapidly rotating, highly-magnetised neutron star, with the rotation axis inclined to the magnetic axis and therefore the radiation beams, as can be seen in Figure 1.1. The observed radiation is the result of the interaction of plasma with the strong magnetic field. The pulses usually have a low duty cycle (the ratio of the pulsar in the 'on' state to the whole period), in the range of a few %, which means that the radiation has to originate from a relatively small region. This small region, the polar cap, is defined by the position of the last open field line. In the region where the open field lines leave the surface of the neutron star, the ionised material is lifted from the surface. Electrons can then be accelerated to very high velocities, following the path of the curved magnetic field lines. This curvature and acceleration, resulting from the change

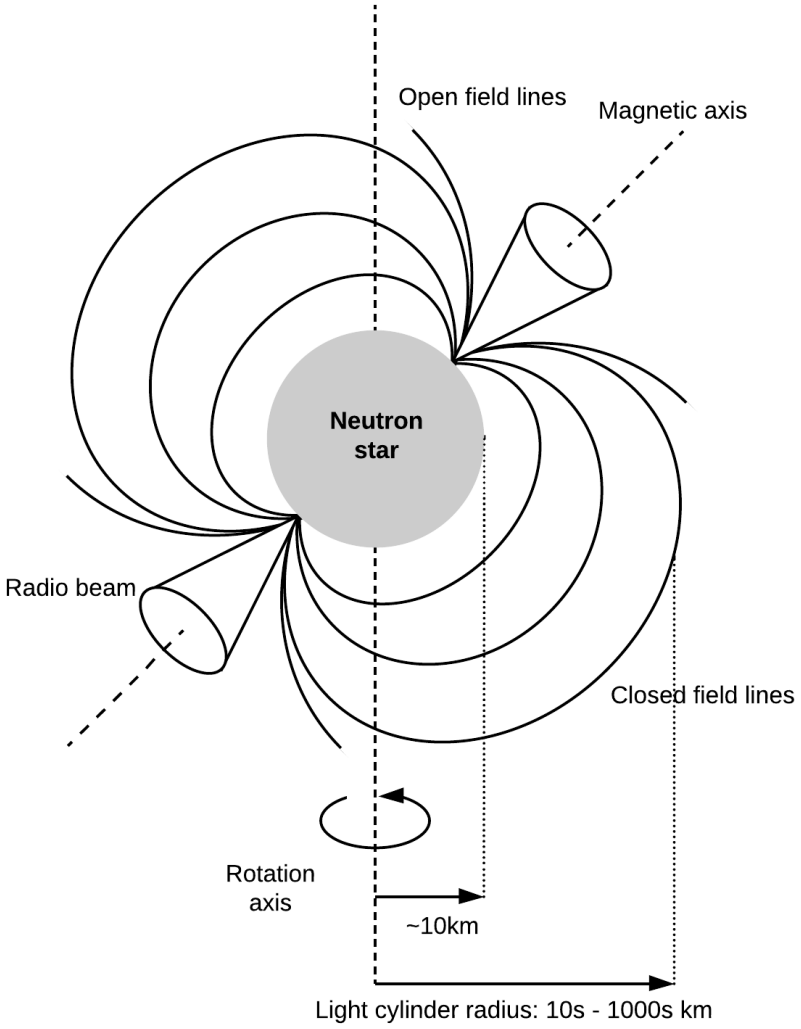


Figure 1.1: A simple pulsar model. A general case with magnetic field axis not aligned with rotation axis is shown. Different components are not drawn to scale.

in the direction in which the electrons move, allows them to emit curvature radiation tangential to the open field lines. It is the rapid rotation combined with the inclination that causes the observed 'lighthouse effect', as the signal from the pulsar can only be detected when the radiation beam crosses the observer's line of sight. Pulsars are usually described as extremely stable rotators, meaning the arrival times of the individual pulses can be accurately predicated on time scales spanning millions of rotations. That does not mean that they do not evolve and undergo changes. Some of these, such as

the spindown rate, can be measured regularly and predicted theoretically, while others, e.g. glitches, are more random in their nature.

The rotational periods of pulsars gradually increase with time and the amount by which the pulsars slow down can be measured. These are usually very small period changes, and to be accurately determined, precise timing has to be performed over the course of months and sometimes years. This slowing down is the result of the constant loss of the kinetic energy converted into the electromagnetic radiation in the form of magnetic dipole radiation. Assuming that pulsars behave like strong and massive dipoles in vacuum, the change in the frequency can be approximated as

$$\frac{df}{dt} = \dot{f} = -Kf^n, \tag{1.1}$$

where f is the rotational frequency, K is a constant and n is the braking index. The value of the braking index depends on the nature of the energy loss mechanism and $n = 3$ when pure magnetic dipole braking is assumed (Lorimer & Kramer, 2004). When the second derivative of frequency is available, it is possible to measure the true value of the braking index. Pulsars for which such measurements are possible show some deviation from $n = 3$, with values of braking index as low as 1.4, suggesting that pulsars can lose energy via some other mechanisms beyond the simple magnetic dipole radiation. As the rate of change of frequency depends on the magnetic field strength, it is possible to obtain an estimate of it from

$$B_S = 3.2 \times 10^{19} \sqrt{\left(\frac{P}{\mathrm{s}}\right)\left(\frac{\dot{P}}{\mathrm{s\,s^{-1}}}\right)}\,\mathrm{G}, \tag{1.2}$$

where P is pulsar period and $\dot{P}$ is the first period derivative. This equation holds for a 'model pulsar' with the pure magnetic dipole-induced energy loss mechanism and a neutron star with a radius of 10 km and the rotation axis at a right angle to the magnetic axis (Lorimer & Kramer, 2004). The magnetic field line strength has values anywhere between 10^{10} G for millisecond pulsars which have the shortest periods and the lowest period derivatives, up to more than 10^{15} G in the case of magnetars. Not all pulsars slow down in a consistent manner though. Some, including the Crab and Vela pulsars, exhibit a glitching behaviour. During these sporadic glitches, a sudden decrease in the rotational period is observed, which is then followed by a slow recovery to the initial state (Radhakrishnan & Manchester, 1969).

The geometry of the radiating region, the emission mechanisms and changes in the

environment of the neutron star can all be estimated from the regular observations of pulsars and measured changes in pulse profiles, polarisation and spin periods. Pulse profiles averaged over multiple rotations can be used to examine the long-period evolution and structure changes and can be used for precise estimates of the pulse arrival times. Individual pulses can be studied to help us gain insight into changes occurring on shorter time scales. Some pulsars show distinct emission modes, with profile components changing in amplitude and also originating at different rotational phases. Sometimes pulsations switch off entirely for a prolonged period of time. The Crab Pulsar also undergoes a totally different change, clearly visible during the examination of individual pulses. It can emit nanosecond-long, energetic bursts, which are few orders of magnitude greater in intensity than its regular pulses (Hankins et al., 2003). These so called Giant Pulses have so far been detected in only a handful of pulsars.

1.2 Rotating Radio Transients

Rotating Radio Transients (RRATs) were first found during the reprocessing of data recorded as part of the Parkes Multibeam Pulsar Survey (McLaughlin et al., 2006), when a single-pulse search was used instead of the usual periodicity search algorithms. A number of pulses were found, which did not appear in periodicity searches. Initially 17 such objects emitting only a single burst were discovered, but additional processing revealed that 6 of those were pulsars previously missed in the periodicity searches. The origin of 11 single bursts remained unknown. Subsequent observations resulted in multiple detections of all 11 sources, but even then, these could not be classified as ordinary pulsars as they could only be detected during the single-pulse searches and were not detectable in the standard periodicity searches. All of these objects had different burst rates and single pulses were detected once every few minutes to hours. The bursts lasted a few milliseconds, meaning that these RRATs had relatively low duty cycles, as compared to pulsars, and remained in their off state for the majority of time. Even though the regular periodicity searches cannot usually be used successfully to detect these objects, the rotation period of RRATs can be estimated by finding the greatest common denominator of the differences between the pulse arrival times. It is important to note that it is possible that the period derived in such a way may not be a true period, but instead an integer multiple of this value. With the increasing number of detections, the probability of such a situation decreases and the period can be established with a greater certainty.

The nature of these new objects was a mystery at the time of their detection and no definitive explanation has been provided over the last 15 years, although more than 100 RRATs currently known (Bingyi Cui, 2016). The measured periods are found in the range between hundreds of milliseconds (125 ms for J1554 − 5209 (Keane et al., 2010)) to a few seconds (7.707 s for J1652 − 4406 (Keane et al., 2011)). Combined with burst widths of the order of milliseconds, this implies compact rotating sources, most likely, rotating neutron stars, as is the case for pulsars. RRATs also tends to cluster within the Galactic plane, similar to the distribution of pulsars. RRATs can therefore be regular pulsars that undergo extreme nulling behaviour, with multiple pulses not detectable due to their low flux or the local environment of the RRAT and the geometric effects affecting the visibility of the radiation. Is is also possible that RRATs are distant and faint objects, and we are only capable of discovering occasional, more energetic outbursts. This theory is supported by the power law distribution of the peak flux density of the pulses similar to that of the giant pulses from the Crab Pulsar, but the magnetic field strengths at the light cylinder thousands of times weaker in terms of RRATs point towards a different emission mechanism (McLaughlin et al., 2006). However the burst widths for the Crab giant pulses are measured in nanoseconds, which is not the case for the millisecond-durations RRAT detections. Several theories currently attempt to explain the behaviour of RRATs, including the disruption of the magnetic fields caused by material falling onto the neutron star.

1.3 Fast Radio Bursts

At the time of writing just over 30 Fast Radio Bursts have been detected (Petroff et al., 2016)[2], with the vast majority discovered with the Parkes radio telescope, as shown in Figure 1.2. Naming follows the convention *FRByymmdd*, where year *yy*, month *mm* and day *dd* refer to the time when the observation containing the burst had been made, which can be more than a decade before the discovery. The first FRB, FRB010724, was detected by Lorimer in 2007 (Lorimer et al., 2007) after the analysis of archival data taken on 24$^{\text{th}}$ July 2001. It was a very bright burst, with an estimated peak flux of 30 ± 10 Jy, the third brightest FRB discovered to date, belonging to a group of only 5 FRBs that have observed peak fluxes greater than 10 Jy. What was most surprising about this event, was its lack of any periodicity or repetition - only a single burst was observed in the 90 h of data.

[2]http://www.frbcat.org

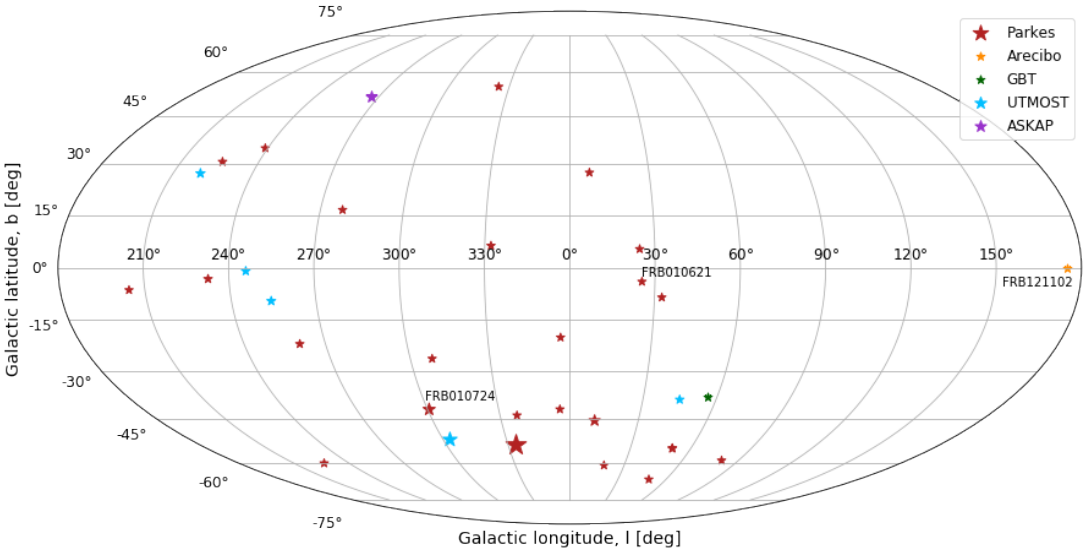


Figure 1.2: The distribution of Fast Radio Bursts on the sky. The centre of the Milky Way is in the middle of the plot at the point (0°, 0°). The size of each marker is proportional to the peak flux density of the burst. A selection effect is clearly visible, with most of the FRBs detected so far using radio telescopes in Southern Hemisphere: ASKAP (Macquart et al., 2010), Parkes (Keith et al., 2010; Keane et al., 2018) and UTMOST (Bailes et al., 2017), with the Northern Hemisphere detections coming from Arecibo (Cordes et al., 2006) and Green Bank Telescope (Ransom et al., 2009)

As can be seen in Figure 1.2, all except only a few FRBs have been discovered at high Galactic latitudes, away from the Galactic plane. At such latitudes, the contribution from our Galaxy to the total dispersion measure (see Section 1.4 for the detailed description of dispersion) is relatively small. The largest fractional contribution, as predicted by NE2001 (Cordes & Lazio, 2002), is for FRB010621, where the Milky Way adds $533\,\mathrm{pc\,cm^{-3}}$ towards the total DM of $746\,\mathrm{pc\,cm^{-3}}$. It has been argued (Bannister & Madsen, 2014), that when considering additional contributions from the diffuse gas, this Fast Radio Burst can be placed within the Milky Way with a probability of 90%, at a distance of 14 ± 6 kpc. So far, it is the only FRB that has a non-negligible probability of being placed inside the Milky Way. The remaining FRBs are most likely outside our own Galaxy, possibly at cosmological distances.

The existing electron density models are believed to be correct, therefore the excess in the measured DM values, which ranges from 10s to $1000\mathrm{spccm^{-3}}$, has to come from the intergalactic medium (IGM) and/or the host galaxy. Associating dispersion measure with interactions with the intergalactic medium places these objects at cosmological distances. Simple distance estimations (Lorimer et al., 2007; Thornton et al., 2013) assume a uniform IGM distribution and a ΛCDM Universe with all the baryons

in the IGM ionised, and follow the relationship

$$z \approx \frac{1}{1200} \frac{\mathrm{DM}}{\mathrm{pc\,cm^{-3}}} \tag{1.3}$$

where z is the redshift of the FRB. It was however noted that IGM can be concentrated within large scale structures such as galaxy clusters and, at very large distances, filaments (Dennison, 2014). The FRB host galaxy can also have a substantial contribution to the measured DM if the burst has been emitted close to the centre of the galaxy or a spiral galaxy is viewed at an inclination close to 90° (Thornton et al., 2013). Lorimer *et al.* estimated the probability that the FRB host galaxy contributed more than $100\,\mathrm{pc\,cm^{-3}}$ towards the DM was 25%. However, Thornton *et al.* showed the probability of viewing a spiral galaxy with $i > 87^\circ$ was around 5%, and therefore any large contribution from a host galaxy for all four FRBs reported in that study was extremely unlikely.

We still have to take the, sometimes large, uncertainties in the electron density distribution models into account. This is especially relevant for higher latitudes, where a smaller number of independent distance measurements, obtained for example from parallax measurements, exists, as compared to the Galactic plane. The lack of full understanding of the composition and processes underway in the Galactic halo also contribute to these uncertainties. Certain corrections and different models have therefore been proposed. The most widely used models, such as BM08 (Berkhuijsen, E. M. & Müller, P., 2008), GBC (Gómez et al., 2001), GMCM08 (Gaensler et al., 2008) and NE2001 (Cordes & Lazio, 2002) show similar results, different by only a factor of 1.5 – 2 for most lines of sight (Schnitzeler, 2012). It is still worth noting, that in some cases, models differ significantly from DMs estimated directly using independent methods, by up to a factor of 15, introduced by features such as the ionised hydrogen regions, which are not taken into account by earlier and less accurate models. These are however easy to identify and can be taken into account when analysing FRB observations. Even if the highest obtained deviation is taken into consideration, the vast majority of Fast Radio Bursts still reside outside of the Milky Way, by a comfortable margin, and can therefore be classified as objects originating at cosmological distances.

The inferred cosmological distances, combined with the apparent lack of repetition, were one of the first signs that a new class of objects might have been discovered. It is important to note that for some of the RRATs described above, only a single pulse has been found as well. The main feature that currently allows us to separate the FRB

population from RRATs is the fact that the modelled Galactic contribution to the DMs of FRBs is much smaller than what would be required for them to be located in our Milky Way. There is still a non-negligible, although a small probability that some non-repeating RRATs have been mislabelled and are indeed FRBs (Keane, 2016) and relying on the excess DM only underestimates the number of detected FRBs, especially if they are present in the Milky Way. Further observations are necessary to detect any possible repeated emission from non-repeating RRATs, that would allow us to fully separate then from FRBs.

Even though FRBs have initially been thought to burst once only, one of them, FRB121102, has been observed to repeat (Spitler et al., 2016), complicating the classification scheme even further. In total, more than several dozen bursts have been observed. The dispersion measures and the associated uncertainties suggest these events come from the same source. These bursts were not uniformly distributed in time and were usually observed in clusters, with multiple single-pulse detections during less than 10 minutes of observation (Spitler et al., 2016) and sometimes no candidates at all for as long as 5 hours during single pointing (Price et al., 2018b). The repeating nature of this FRB allowed for more accurate, sub-arcsecond localisation with targeted interferometric observations and coordinated efforts to observe it with multiple telescopes at different wavelengths (Law et al., 2017). FRB121102 has since been associated with a persistent radio source, as well as its optical counterpart and its host galaxy have been identified (Chatterjee et al., 2017; Marcote et al., 2017). It most likely resides in a star-forming region of its host irregular dwarf galaxy, which supports the theories of FRBs originating from young and energetic objects, such as strong magnetars (Tendulkar et al., 2017; Bassa et al., 2017a). The discovery of repeated bursts indicates that not all objects are destroyed during cataclysmic events, as the single-burst nature of most FRBs currently suggests. The inferred distance and a low burst flux, as compared to the rest of the FRB population, place the luminosity of this FRB few orders of magnitude below the luminosities of other events, showing that it may indeed belong to a different class of objects. It is still unknown whether other FRBs do not really repeat or that their apparent lack of repetition is a result of our inability to detect weaker bursts or not observing with a high enough cadence.

1.3.1 Current Theories

Giant pulses. Short duration and high flux density makes giant pulses a possible candidate for the explanation of Fast Radio Bursts. Weaker pulsations might pass

undetected due to their low signal-to-noise ratio and only the brightest pulses can be observed. The main argument against this theory applying uniformly to the whole FRB population is the lack of any repeated events in the follow-up observations for all bursts except FRB121102, as described above. For FRB010621, the probability of not seeing any other pulses was estimated to be very low if the emitting object has characteristics similar to those of the Crab Pulsar (Keane et al., 2012). The theory of extremely bright radiation from a rotating neutron star also fails to explain Lorimer Burst in a satisfactory manner, where the inferred distance to the source, combined with the flux of 30 Jy, would mean very high luminosity, unobserved anywhere else. It is possible that two classes of FRBs have already been observed, with different mechanisms producing repeating bursts and the ones that appear once only have their progenitors destroyed as a result.

Collapsing neutron stars. A rotating neutron star can have a mass above the critical mass for non-rotating models, as it is supported against the gravitational collapse by the centrifugal force. Spinning down will cause the centrifugal force to decrease until a critical moment, when it can no longer balance the gravitational force, causing the neutron star to collapse. This collapse can occur up to few million years after the birth of the neutron star, for the lowest-mass objects, so there may be no observable evidence left for the preceding supernova explosion (Falcke & Rezzolla, 2014), making any association with massive stellar progenitors difficult or even impossible. As the collapsing parts of the remnant move inside the event horizon on the scales of tens of milliseconds (Falcke & Rezzolla, 2014), the resulting Gamma Ray Burst (GRB) would be much shorter than other short GRBs and would therefore be characteristic for this process. Electromagnetic emission is expected to come from magnetic field lines reconnecting outside the horizon, as the magnetosphere is far enough from the neutron star, so that it is not consumed by the event horizon. An assumption that about 3% of all the neutron stars will collapse, can explain the observed FRB rates. Theoretical calculations (Ravi & Lasky, 2014; Zhang, 2014) showed that some stars can collapse much earlier, within seconds to hours after the neutron star formation. Predicted rates for this class of objects are however around 3 orders of magnitude lower than for FRBs and therefore they cannot account for all of them. Follow-up observations looking for radio bursts following the gamma-ray detections were proposed and will be necessary to help link bursts with this model (Falcke & Rezzolla, 2014).

Merging neutron stars. Neutron stars in binary systems are expected to have

their orbits shrink due to the gravitational interaction and spiral down towards their companions. As stars get closer to each other, their strong magnetic fields start to interact with each other. The main model (Hansen & Lyutikov, 2001) assumes a system with two components: a rapidly spinning, recycled pulsar, with a weaker magnetic field of the order of $10^9 - 10^{12}$ G and a slowly rotating one, with a much stronger magnetic field of $10^{12} - 10^{15}$ G. Interaction of the recycled star with the magnetic field of the 'normal' star produces surface charges, which in turn create a strong electric field. This field then accelerates charges in a radiation beam. When sufficient energies are reached, photons and electron-positron pairs are produced. This process extracts energy from the system, which can then be released in the form of coherent radio emission. The predicted flux density (Hansen & Lyutikov, 2001; Keane et al., 2012)

$$S \approx 1 \left(\frac{\varepsilon}{0.1}\right) \left(\frac{D}{100\,\mathrm{Mpc}}\right)^{-2} \left(\frac{B}{10^{15}\,\mathrm{G}}\right)^{\frac{2}{3}} \mathrm{mJy}, \tag{1.4}$$

where ε is the efficiency factor, D is the distance to the merging system and B is the magnetic field strength of the non-recycled neutron star, provides distance estimates for FRB010621 inconsistent with the dispersion measure obtained using the NE2001 electron density model (Keane et al., 2012). Modification to this model has been proposed, where the emission occurs after the final object formation (Pshirkov & Postnov, 2010). In this scenario the magnetic field strength is amplified as a result of a differential rotation. The modified model is argued to provide satisfactory explanation for Fast Radio Bursts emission, for magnetic field strengths amplified to 10^{13} G (Totani, 2013).

According to some models it is possible that the pair of merging neutron stars will produce a repeating FRB similar to FRB121102. The merger may leave a stable neutron star behind, which will sporadically cause bright bursts on the scale of milliseconds. This neutron star would have to be stable enough so it does not collapse into a black hole on time scales shorter than the time span over which the repeating FRB has been observed. Depending on the model, the emission may continue over the course of tens of years and become weaker with time as the remaining neutron star slows down (Yamasaki et al., 2017).

One of the important questions of the neutron star merger progenitor model is the rate at which such events occur. There are large uncertainties in the theoretical model predictions, but the recent discovery of the gravitational waves from the inspiral and

ultimate collision of two neutron stars, places these rates at $1500^{+3200}_{-1220}\,\mathrm{Gpc}^{-3}\mathrm{yr}^{-1}$ (Abbott et al., 2017). This is at least an order of magnitude lower than the rate required to observe all FRBs if they were expected to originate solely from the neutron star-neutron star mergers (Callister et al., 2016).

Non-destructive events. As all but one FRB have been observed only once, most theories tend to favour cataclysmic events, where the emitting object is destroyed, with the exception of for example giant pulses, as described above. The detection of the repeating FRB121102 and association with the host galaxy means it is necessary to take other scenarios into consideration, at least in the case of this object, if not the whole FRB family. In case of the FRB121102 it is possible that the spin down of the neutron star transfers power to a persistent radio source, most likely a Supernova Remnant (Marcote et al., 2017). Another possibility is that these bursts are caused by an Active Galactic Nucleus (AGN), or due to its interaction with a nearby neutron star. It is however difficult to reconcile this theory with the low mass of the host galaxy and the offset of the persistent radio source from the centre of the galaxy (Tendulkar et al., 2017).

RFI. In the years following the discovery of the first FRB, it was not possible to fully rule out terrestrial origins, especially the man-made Radio Frequency Interference (RFI). As most of the RFI signals are found at low-DM values, they are easy to filter out, but it is important to note that some of the terrestrial signals can undergo higher than usual dispersion. This was the case when a new class of events called perytons was reported in 2011 (Burke-Spolaor et al., 2011). They were found at DMs between 200 and $400\,\mathrm{pc\,cm}^{-3}$, a range which included the Lorimer burst with a DM of $375\,\mathrm{pc\,cm}^{-3}$, but unlike usual dispersed signals, they showed only an approximate f^{-2} dependence. These pulses were also observed in all of the 13 beams of the Parkes receiver. A truly astrophysical source should only be found in a small number of adjacent beams, which indicated a peryton source located in the vicinity of the telescope. These were later identified to be caused by microwave ovens located near the observing site (Petroff et al., 2015). As all of the FRBs found so far, have been observed in a single or only a couple of spatially coincident beams, this strongly indicates their extraterrestrial origin. The repeating nature of the FRB 121102 and the identification of its host galaxy also show that these signals have their origin in astrophysical and not terrestrial (man-made or natural) processes.

Other possibilities. The theories briefly outlined above are widely considered in

the available literature, but none currently fully explain the origin of Fast Radio Bursts. There are a number of other processes that are thought to explain, even partially, the nature of FRBs. These include, but are not limited to:

- evaporating black holes (Keane et al., 2012);
- superconducting cosmic strings (Yu et al., 2014);
- magnetar flares (Popov & Postnov, 2015);
- asteroids colliding with neutron stars (Geng & Huang, 2015);
- starquakes (Wang et al., 2018);
- transition from black to white holes (Barrau et al., 2014).

1.4 Propagation Effects

As the radiation emitted by a pulsar has to travel large distances before it is reaches the observer, it undergoes a lot of changes through the interaction with particles found in the interstellar medium (ISM). These interactions have to be taken into account and corrected if the correct shaped of the signal is to be recovered. If appropriate corrections are not applied the signal can deteriorate to the point that the signal becomes indistinguishable from the noise and only the brightest and closes pulsars can be observed. The two major effects for pulsar observations: dispersion and scattering, are discussed below.

1.4.1 Dispersion

Even though photons themselves do not have electric charge, they can interact with the ionised matter in the interstellar medium. As charged particles are accelerated in the electric field generated by moving photons, their movements induce a current inside the tenuous plasma. Electrons, protons and other ions all undergo acceleration, but only the current generated by electrons is important when interacting with electromagnetic waves. The velocity $\boldsymbol{v}$, which the particles are accelerated to, and therefore the generated current density, are inversely proportional to the mass of the particle

$$J = -Ne\boldsymbol{v} = \frac{iNe^2}{m\omega}\boldsymbol{E}, \tag{1.5}$$

where $\omega = 2\pi f$, N is the number of particles under consideration and $\boldsymbol{E}$ is the complex-valued electric field intensity, as described using Maxwell's equations. The current density produced by moving protons has magnitude almost 2000 times smaller than

the one introduced by electrons and can therefore be neglected as well as any currents coming from heavier ionised particles.

This interaction causes the plasma to become conductive and introduces the dependency of the group velocity, the velocity at which the electromagnetic waves travel through the interstellar medium, on the frequency of photons

$$v_g = c\sqrt{1 - \left(\frac{\omega_p}{\omega}\right)^2}, \tag{1.6}$$

where ω_p is the plasma frequency - the frequency below which the electromagnetic waves are not able to travel through the ionised plasma

$$\omega_p = 2\pi f_p = 2\sqrt{\frac{\pi n_e e^2}{m_e}}. \tag{1.7}$$

The velocity of photons is therefore no longer equal to c, but lower and also inversely proportional to the square of the frequency, which means that waves travelling with higher frequencies reach the observer first. The time difference, Δt between signals arriving across the band can be calculated using

$$\Delta t = 4.15 \times 10^6 \times \left(\frac{1}{f_b^2} - \frac{1}{f_t^2}\right)\left(\frac{\mathrm{DM}}{\mathrm{pc\,cm^{-3}}}\right)\mathrm{ms}, \tag{1.8}$$

where f_t is the frequency of the top channel and f_b the frequency of the bottom channel $f_t - B$, where B is the total system bandwidth, and both frequencies are expressed in MHz. The number at the front of the above equation is the approximation of the dispersion constant, $\mathcal{D}$ and is defined in CGS units as

$$\mathcal{D} = \frac{e^2}{2\pi m_e c} = \frac{f_p^2}{2cn_e} = 4.148808 \times 10^3\,\mathrm{MHz^2\,pc^{-1}\,cm^3\,s}. \tag{1.9}$$

The reciprocal of Equation 1.9 is often defined and used as the dispersion constant, which follows the original definition (Manchester & Taylor, 1972)

$$K = \frac{1}{\mathcal{D}} = 2.41 \times 10^{-4}\mathrm{MHz^{-2}\,pc\,cm^{-3}\,s^{-1}}, \tag{1.10}$$

The dispersion measure, DM measured in the units of $\mathrm{pc\,cm^{-3}}$ over the distance D

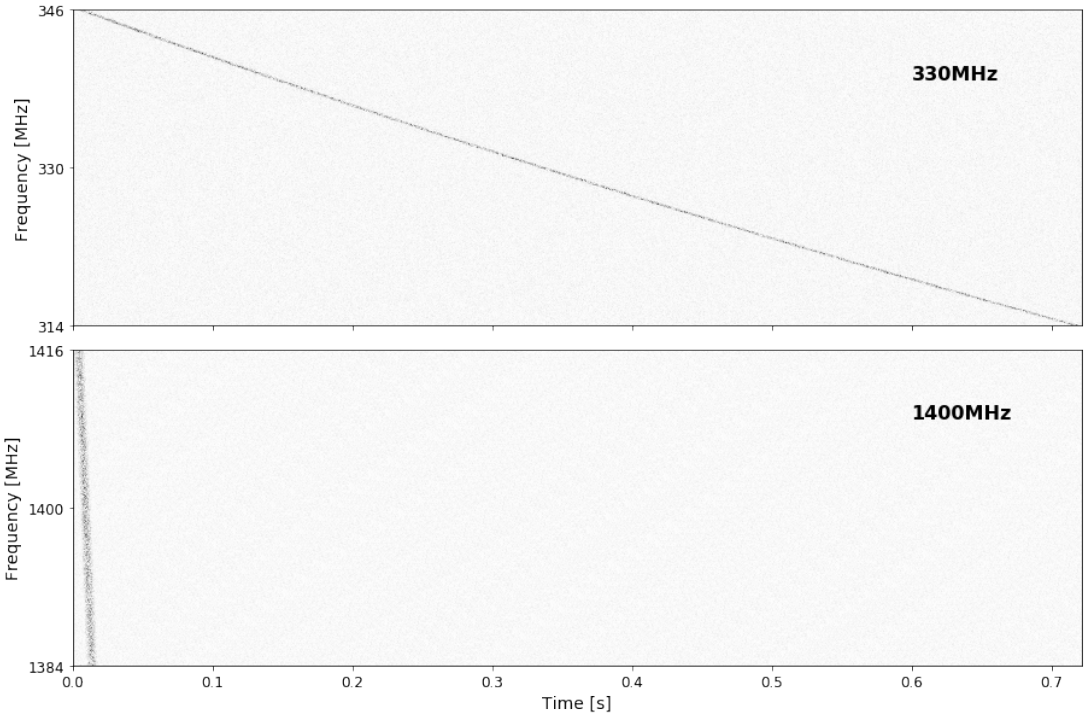


Figure 1.3: Dispersion comparison for simulated signals with the dispersion measure of $100\,\mathrm{pc\,cm^{-3}}$ with systems operating at centre frequencies of 330 and 1400 MHz. It can be seen that the dispersion delay becomes more pronounced at lower operational frequencies, which has implications for the amount of computational resources required to fully dedisperse the channelised data and the amount of smearing introduced in the incoherently dedispersed time series. The beginning of the horizontal time axis marks the start of the simulation.

between the pulsar and the observer, is defined as the integral

$$\mathrm{DM} = \int_0^D n_e(l)\mathrm{d}l. \tag{1.11}$$

It is the column density of electrons along the line of sight towards the pulsar. If the number density of the ISM is known, it can be used to obtain the distance to the pulsar, with the value of DM coming from the delay across the receiver bandwidth.

The amount of dispersion smearing per channel of width $\Delta f = \dfrac{B}{N_c}$, which increases the width of the pulse with the intrinsic width W, can be derived from Equation 1.8 and approximated by

$$\Delta W = 8.3 \times 10^6 \times \frac{\Delta f}{f^3}\mathrm{DM\,ms}. \tag{1.12}$$

This introduces pulse broadening, increasing the width of the dedispersed pulse beyond its intrinsic width and contributions from the sampling time.

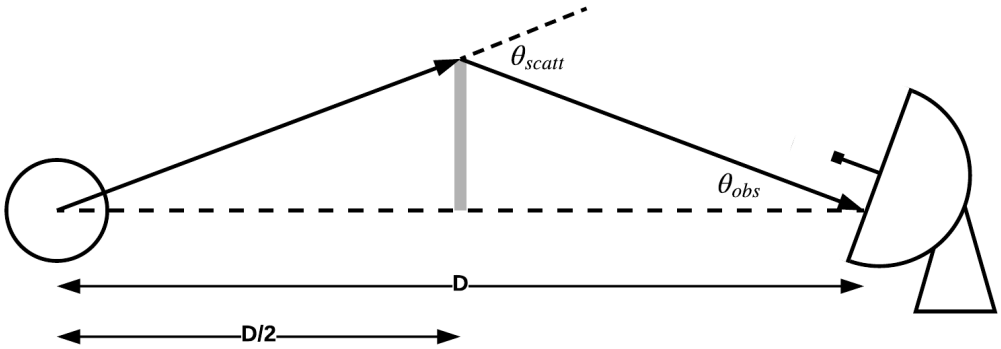


Figure 1.4: Thin screen approximation of scattering. The radiation from a pulsar at a distance D from the observer is scattered by a thin a screen containing irregularities in the electron density distribution with a characteristic scale a. The screen is place halfway between the pulsar on the observer in this approximation, but can be easily generalised.

The effect of dispersion on the signal can be clearly seen in Figure 1.3, which shows the comparison of time delays between bursts at a DM of 100 pc cm^{-3} for two systems: one with a top frequency of 330 MHz and the other at 1400 MHz. Both signals have been recorded over a bandwidth of 32 MHz. It is clearly visible that the dispersion delay is much more pronounced at lower frequencies. This has an important implication in the design and execution of pulsar and transient search systems, as the larger number of time samples has to be used to recover the original signals at low frequencies, increasing computational and memory requirements.

1.4.2 Scattering

As signals from a pulsar travel through the turbulent ISM, they encounter irregularities in the electron density distribution $\Delta n_{\rm e}$, which introduce phase difference to a previously coherent radiation (Scheuer, 1968). When covering the distance D to the observer and encountering irregularities with a characteristic length of a, the radiation, on average, has to travel through d/a irregularities. For the radiation with wavelength λ the root mean square phase variation can then be approximated by (Lorimer & Kramer, 2004)

$$\Delta\Phi = \sqrt{Da}\lambda\Delta n_{\rm e} r_e, \tag{1.13}$$

where r_e is the classical electron radius (expressed in the CGS units)

$$r_e = \frac{e^2}{m_e c^2}. \tag{1.14}$$

In the simplified scattering model, free electrons are assumed to form a single thin screen, which contains the electron density irregularities, placed halfway between the pulsar and the observer (Figure 1.4). Equation 1.13 holds for most cases in the above approximation and $\theta_{\rm scatt}$ can be calculated

$$\theta_{\rm scatt} = \frac{\Delta\Phi\lambda}{2\pi a} = \sqrt{\frac{D}{a}}\frac{\Delta n_{\rm e}\lambda^2 r_e}{2\pi}. \tag{1.15}$$

The difference in arrival times between rays travelling in a straight line and those scattered towards the observer is

$$\tau_s = \frac{D\theta_{\rm obs}^2}{2c} = \frac{D\theta_{\rm scatt}^2}{8c} = \frac{D^2}{a}\frac{(\Delta n_{\rm e})^2 r_{\rm e}^2\lambda^4}{32\pi^2 c} \text{ s}. \tag{1.16}$$

The most important dependency, from the observational point of view, is that on the frequency: $\tau_{\rm s} \propto f^{-4}$. This can be observed as a slow, exponential decay of the pulse.

Both dispersion and scattering influence the final form of the observed pulse. All the contributions to the final pulse width can be accurately approximated by uncorrelated random variables and can therefore be added in quadrature as

$$W = \sqrt{W_{\rm int}^2 + t_{\rm samp}^2 + \Delta W_{\rm ch}^2 + t_{\rm scatt}^2}, \tag{1.17}$$

where $W_{\rm int}$ is the intrinsic width of the pulse, $t_{\rm samp}$ is the data sampling interval, $\Delta W_{\rm ch}$ is the dispersion broadening per channel as described by Equation 1.12 and $t_{\rm scatt}$ is the broadening due to interstellar scattering.

1.5 Thesis Outline

This thesis is organised in the following manner:

- **Chapter 2** describes the techniques used the detect sources described in Chapter 1, including the overview of the radio astronomy techniques as well as different computational algorithms used to aid the search with the main focus on the GPU computing.

- **Chapter 3** contains a detailed description of the PAFINDER pipeline designed for the Effelsberg radio telescope Phased Array Feed used for real-time radio transient detections.

- **Chapter 4** discuses the results obtained during the commissioning of the Effelsberg PAF and the PAFINDER pipeline. The focus is on the ability of the pipeline to detect single pulses from known sources and then attempting to find new and redetect known FRBs.

- **Chapter 5** describes the single-pulse injection software used to simulate the data which can later be used to better understand to limits of current and future FRB surveys.

- **Chapter 6** presents the overview of the processing pipeline and FRB search results for the GMRT High Resolution Southern Sky survey.

- **Chapter 7** discuses the conclusions, the extent of work and results achieved and possible future developments for the work done so far.

Chapter 2

Signal Processing

2.1 Radio Telescope Basics

The simplified view of a single-dish radio telescope receiver system is shown in Figure 2.1. The incoming electromagnetic radiation induces surface currents in the surface of the reflector upon hitting the antenna. These currents become sources of radiation with the same frequency as the incoming signal. They can then be collected by the receiving system placed in the focus of the antenna. The signal arriving at the focus is usually very weak and therefore has to be amplified. It is important to introduce as little extra noise as possible in this stage so not to drown the signal even further. This can be achieved with the use of Low Noise Amplifiers (LNAs), which are often kept at low temperatures, typically around 20K. Even though components present in the system after the amplification stage have much larger noise contributions, they can be effectively neglected due to the large gain of the first LNA. The amplified signal is then passed through a bandpass filter, which narrows the range of frequencies that have to be processed in the following stages. This removes some unwanted signals, such as man-made RFI and prevents aliasing, before further processing. In the next step, the frequency of the incoming signal is converted to (usually) lower frequency with the help of mixers. Driven at a frequency f_{LO} from the local oscillator, they change the incoming frequency, f_{RF} by multiplying the two waveforms. It is practical to choose a lower output frequency as it is much easier to transmit, due to reduced losses, and amplify signals at lower frequencies. It is also more cost-effective to build receiver systems operating at lower frequencies. There are two possible solutions for a lower output frequency. When $f_{\mathrm{RF}} > f_{\mathrm{LO}}$ the upper sideband is obtained, and when $f_{\mathrm{RF}} < f_{\mathrm{LO}}$ the lower sideband is produced. The mixed signal is then passed through

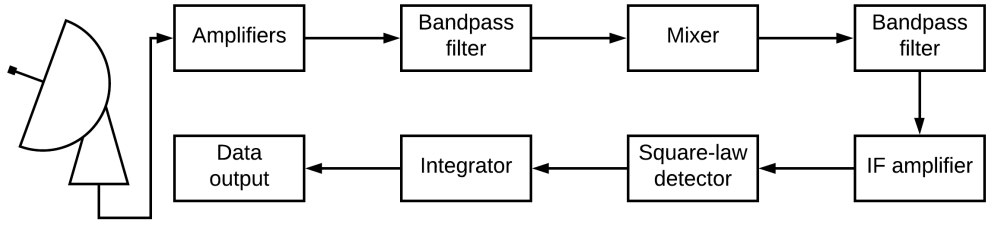


Figure 2.1: Simple heterodyne receiver system. In modern systems digitisation is usually present after the first amplification stage and all the other processing stages can be implemented in software.

another filter that lets only one sideband through. In analogue systems, the square law detector produces an output which is the square of the input voltage, proportional to the power. This is then passed to the integrator which averages the power over long periods of time. At this point only a steady component of power with small variations is recorded. This data can then be sent to the data recording or processing devices which depend on the requirements of the project.

In modern systems a different approach is used. The incoming signals, which are analogue outputs of the receiver system, are digitised, commonly after the first amplification stage. This avoids the need to use the analogue components which lack flexibility and are more sensitive to environmental changes. The input signal is continuous in both amplitude and time and an Analogue to Digital Converter (ADC) has to be used to convert it to the digital signal, which is its discrete representation. Amplitude quantisation is achieved by representing the amplitude using a finite number of bits, with N-bit quantisation providing 2^N levels, or bins, which can be used to approximate to original voltage with a finite number of digital values. Systems making use of a larger number of bits can be used to produce digital signal which more closely resembles its analogue counterpart. There is however a cost of increasing component complexity due to the higher number of bits that have to be processed. The incoming signal, limited to a bandwidth B can be represented exactly (after interpolation) in its digital form when it is sampled at a Nyquist rate[3] of $2B$. The highest sampling time that can be used is therefore $\frac{1}{2B}$. If a smaller sampling rate is used, the signal is then undersampled, resulting in the introduction of aliasing (e.g. a sine wave sampled one

[3]Depending on the application, the Nyquist rate can be defined as half of this value, and then the signal is said to be sampled at twice the Nyquist rate

every period, can be interpreted as a signal with a constant amplitude, instead of varying periodically). The final processing stage usually depends on the project requirements. In pulsar research, the digital signal usually has to be divided into a number of channels. This can be achieved by using Digital Autocorrelation Spectrometers, FFT spectrometers or filterbanks. Currently these steps can all be performed in software, with dedicated hardware being phased out, with conversion of the signal to its digital form now present at early stages of processing and not only used as means of storing the data. This allows for greater flexibility of the system, e.g. sampling time or channel bandwidth can be easily adjusted, and lowers the development costs as most software can now run on off-the-shelf hardware and does not require custom-build solutions.

2.2 Candidate Detection

2.2.1 Dedispersion

Before any pulsars or other radio transients can be detected, the effects of dispersion, described in Section 1.4 have to be removed. The approach to dedispersion can first be divided into two main categories: coherent and incoherent.

Incoherent dedispersion can be implemented in a number of ways, using the final channelised signal, with the power in each channel as input. The simplest algorithm applies the correct delay, calculated using Equation 1.8, at each of the channels and sums them, resulting in a dedispersed time series, according to the equation

$$S = \sum_{c=0}^{N_c-1} F_{c,t+\delta t'}, \tag{2.1}$$

where c is the channel number, N_c is the total number of channels in the file, t is the time sample at the reference channel and F is the input array representing the channelised data. $\delta t'$ is the discrete version of Equation 1.8, i.e. it is divided by the sampling time t_{samp} and rounded to the nearest integer, to obtain the index in the array where the value is read from. When looking for new objects, the dispersion measure is not known beforehand, and therefore a large number of DM values, often counted in thousands, have to be tried out and the resulting dedispersed time series searched for the presence of signals of interest. When a simple brute-force approach is used, Equation 2.1 is used to obtain the time series for each of the DM trials without implementing any optimisations and simplifications that would speed up the execution. It simply relies on the

raw processing power of the available architecture to achieve the required processing speed.

A number of algorithms have been developed over the years, that instead of fetching each time sample and calculating each sum separately for every dispersion measure trial, implement more efficient data reuse approaches. Inspired by the speed-up offered by the Fast Fourier Transform algorithm, tree dedispersion was introduced in the 70's (Taylor, 1974). It first simplifies the problem by assuming the delay is linearly proportional to the frequency. The dedispersed time series is then obtained by adding linearly delayed time samples and also reusing the data from previous steps. In its most basic form, the tree dedispersion has some significant shortcomings. Due to the linear function of delay, it is ill-suited for processing data recorded over a large bandwidth, low frequencies and high dispersion measures. This limitation can be overcome in two ways: the dispersion trail can be approximated by a number of linear segments, or the frequency coordinates can be changed to make the dispersion curve linear (Manchester et al., 2001). The tree dedispersion implementation allows for the reduction of the computational complexity from $O(N_t N_{DM} N_c)$ in the case of the direct algorithm down to $O(N_t N_{DM} \log N_c)$, where N_t is the number of time samples and N_{DM} is the number of trial dispersion measures (Barsdell et al., 2012).

Another approach uses the so-called subband dedispersion. It works by first dividing the incoming frequency dimension into a number of subbands. Each subband is then dedispersed using a reduced number of nominal dispersion measures. With channels in each subband dedispersed to the highest channel in a given subband, the output has the form of a small filterbank, with each dedispersed subband contributing a single channel in the new filterbank. Finally, the resulting subbands are dedispersed around the nominal DMs using the standard brute-force approach, this time working on data with significantly lower number of frequency channels. Using a smaller set of nominal dispersion measures, and approximating the total dispersion, this method introduces an additional smearing as compared to the brute-force approach. This however can be limited by carefully selecting the size of the subbands and the separation between the nominal dispersion measures (Barsdell et al., 2012).

As incoherent dedispersion is performed after the signal detection, all the phase information is lost and cannot be recovered. This can be avoided by using coherent dedispersion, which is applied to the original voltage and therefore can make use of the phase information. In the implementation of incoherent dedispersion, the dispersive interaction between photons and electrons in the ionised medium can be expressed

as the modification of the incoming radiation with a transfer function. This transfer function is a complex-valued, phase-only function that does not change the amplitude of the Fourier components of the signal, but only their phases, effectively applying a delay in the time domain. It has the form of

$$H(f_0 + f) = \exp\left(\frac{i2\pi\mathcal{D}f^2}{(f+f_0)\,f_0^2}\mathrm{DM}\right). \tag{2.2}$$

The above transfer function depends on the centre frequency f_0 and runs for a frequency range $\frac{-B}{2} \leq f \leq \frac{+B}{2}$ (Hankins & Rickett, 1975). The incoming voltages are then multiplied by its inverse H^{-1}, which allows for the original, non-dispersed signal to be recovered. The resulting dedispersed time series has a dispersion smearing proportional to $\frac{1}{B}$, as coherent dedispersion is run using original voltages and not the channelised data which has reduced time resolution. This smearing is much smaller than in the case of the incoherent dedispersion, where the pulse is distorted as described by Equation 1.12.

Coherent dedispersion is computationally expensive, considerably more than the incoherent approach. This is caused by the fact that the transfer function has to be generated for every single dispersion measure trial, which can be as high as a few thousand for standard pulsar and transient surveys, and it is applied to raw voltages, which have higher data rates than the channelised and often time and frequency averaged detected signals. So far, no viable system has been implemented that allows real-time fully-coherent dedispersion using thousands of DM trials. Any practical applications are limited to blind searches within a limited DM range or dedispersing signals from a source with a known DM, previously obtained using the incoherent approach, to improve the resolution of the dedispersed pulse profile.

2.2.2 Periodic Signals Detection

As pulsars emit at regular intervals, a number of techniques exploiting their periodic nature can be used that allow us to discover these signals. One of the most common techniques used for periodic signal detection is calculating the Discrete Fourier Transform (DFT). A number of efficient algorithms, collectively called the Fast Fourier Transform (FFT) algorithms, exist that significantly reduce the time required to obtain the DFT. One of the most commonly used FFT algorithms, is the Cooley-Tukey algorithm (Cooley & Tukey, 1965).

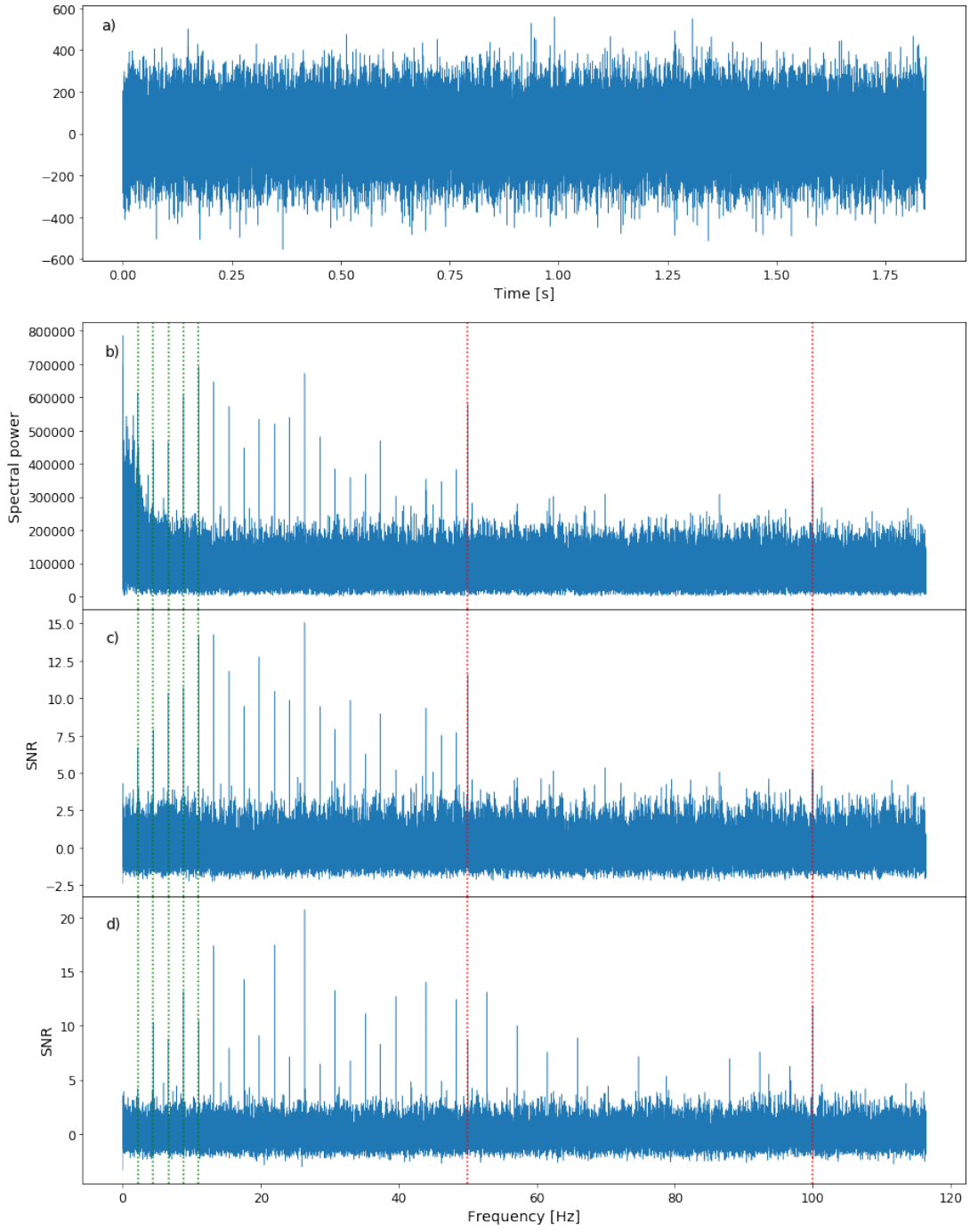


Figure 2.2: Stages in the periodic signal detection. From top to bottom: a) time series of pulsar J1514 − 4834, with periodic signal not clearly visible; b) power spectrum with fundamental and first few harmonics marked in green, in red marked 50 and 100 Hz signals from power lines, the effect of red noise can be seen as the increase of noise power at frequencies lower than ∼10 Hz; c) dereddened and normalised power spectrum; d) first harmonic summing applied.

In its most basic form, the Discrete Fourier Transform converts the complex-valued signal of length N samples in the time domain $f(t)$, into a different complex-valued signal in the frequency domain $F(k)$, by computing

$$F(k) = \sum_{n=0}^{N-1} f(t_n) \exp\left(\frac{-2\pi i n k}{N}\right). \tag{2.3}$$

The 'fast' version of this algorithm works by splitting the incoming time series into smaller portions recursively to reduce the computational complexity. For example, one of the most commonly used algorithms, Radix-2, recursively splits the input data sequence in half and therefore works the best for signals that are power-of-2 in length. The full DFT can then be obtained by reusing the data when combining the halves, reducing the computational complexity from $O\left(N^2\right)$ to $O(N \log N)$ in the best-case scenario.

The resulting Fourier series can then be further examined in terms of the power spectrum, $P_k = \mid F_k \mid^2$. Figure 2.2 shows the time series, with a periodic signal present and its corresponding power spectrum. It can be clearly seen that the power spectrum suffers from increased noise at lower frequencies, so called red noise, arising from the long-period variations, such as RFI and receiver temperature fluctuations. This can become problematic during searches for low frequency pulsars. In such a situation, the signal can be easily buried in the noise, as can be seen in panel b) in Figure 2.2, and its recovery can be made very difficult if not outright impossible, as any techniques currently in use can also significantly reduce the power in the first few harmonics present at low frequencies when removing the effects of red noise. This difficulty in detecting low-frequency, i.e. long-period pulsars can result in the underestimation of the pulsar population counts at such low frequencies. Several dereddening techniques exist, such as median filtering, where a running median is subtracted and the signal is normalised to zero mean and unit rms. It is however important to note that this only reduces the amount of red noise and does not eliminate it completely and can also have a negative effect on the pulsar signal.

When a strong and narrow pulsar is detected, the power is not constrained to its fundamental frequency only, but is also spread over the multiple harmonics. Harmonic summing can then be used to increase the signal-to-noise ratio (S/N). The original power spectrum is stretched by a factor of 2 and added to the original spectrum, therefore combining power from fundamental and second harmonics. Summing the noise

increases its contribution by a factor of $\sqrt{2}$, but more importantly the powers of harmonics add up as well, resulting in an overall increase in the S/N. When adding two harmonics of equal power, the resulting signal-to-noise ratio can be $\sqrt{2}$ greater than the original one. The effect of the harmonic summing can be seen in the bottom panel of Figure 2.2, where the SNR even after just one iteration of the algorithm is significantly improved. The signal can be stretched multiple times which results in the fundamental and lower frequency harmonics being added to the harmonics at higher frequencies, increasing the S/N even further. It is however important to note that this procedure can increase the SNR of unwanted signal, especially the 50 Hz RFI, when 100 Hz, 150 Hz and higher harmonics are added, therefore extra case has to be taken to properly filter out signals in these regions.

The presence of red noise means that pulsars with long rotational periods can be missed in the FFT-based periodicity searches, as the signal can be obstructed by the noise. These destructive effects can be decreased when a different technique, folding, is used, which is less sensitive to this kind of noise. This makes it possible to find low-frequency pulsars, allowing us to better understand the constraints on pulsar periods in the region that FFT is not sufficiently sensitive. Simply described, the folding algorithm relies on taking the time series, dividing it into chunks at regular intervals, corresponding to the pulsar period and adding these chunks to form the final pulse profile. The process is then repeated for a large number of trial periods, which allows us to find the period that results in the highest S/N. The folding algorithm is computationally expensive in its simplest form. There are however a number of implementations, fast folding algorithms, that can reduce the computational time, even at the cost of increased code complexity necessary to fully-optimise the code and offer a significant improvement over the standard FFT-based pulsar search methods (Cameron et al., 2017). They again reuse the data, as certain time samples used for folding at one period can later be used for folding at different periods.

2.2.3 Single-pulse Detection

A single pulse searching procedure starts similarly to the usual pulsar search - with dedispersion. As there are no periodicity searches involved, the Discrete Fourier Transform is not calculated and all of the work is performed on the time series. One of the methods currently used for detecting single pulses is that of matched filtering. This is achieved by convolving the time series with the shape of the burst. Theoretically, any shape can be used, but as generalised convolution can be computationally expensive,

even when run using the FFT of the time series and the shape function, exploiting the convolution theorem. To reduce computational resources and time required, a simple boxcar function is convolved with the times series, which can be achieved by using a simple running sum of the data and subtracting the copies of it, shifted by a distance corresponding to the width of the boxcar function. As the duration of the pulse is usually not known before the data has been processed, a number of different boxcar widths have to be tried. A correct filter width maximises signal-to-noise ratio in the output and the candidates that are found above a set S/N threshold are selected. A single pulse can usually be detected at multiple time samples and DM values close to the true properties of the burst. The candidates found away from the true DM will usually be wider, as they are not dedispersed properly - either too much or too little, which in turn results in a lower S/N of the signal detected in the time series. These multiple candidates can later be combined into a single candidate, with the true properties estimated from the highest S/N results. As the available computing power grows all the time and new processing techniques are developed, it may soon be viable to use other shapes in real time than just a simple boxcar function, which should help us to better match the shapes of true bursts and also identify RFI.

2.3 Phased Array Feeds

In order the increase the number of radio transient discoveries, new radio telescopes and receiver systems can be built to look deeper into the Universe, allowing us to pick up fainter objects and to cover larger portions of the sky at any given time, increasing our chances of detecting rare and unpredictable events, such as FRBs. The survey speed, SVS, can be approximated by (Fisher et al., 2009)

$$\mathrm{SVS} \propto N_b \Omega_b B \left(\frac{A_{\mathrm{eff}}}{T_{\mathrm{sys}}} \right)^2, \tag{2.4}$$

which depends on the number of beams, N_b, the solid angle covered by a single beam Ω_b, the receiver bandwidth B, the system temperature, T_{sys}, and the effective collecting area of the antenna, A_{eff}, which depends on the geometric area of the antenna and various efficiency factors. This provides the metric that depends on both the amount of sky that is covered at any given time and the sensitivity of the system.

To improve the chances of detecting a radio transient, this value should be as high as possible. To achieve that, we can either increase the instantaneous field of view

(FOV) or increase the sensitivity and look for fainter objects. Both of these approaches result in a larger surveyed volume of the Universe, but the usefulness of either of them depends on the characteristics of the studied objects (e.g. bright objects where the sensitivity is not that important). Ideally, both the FOV and the sensitivity can be improved in future radio telescopes, but there are practical limitations on what can be made better. For an existing antenna it would be impossible to change its collecting area. Building large radio telescopes is a challenging task and currently only two fully-steerable telescopes with a diameter equal to or more than 100 m exist in the world. It is also important to note that although the larger effective area can help to detect fainter signals, it also means smaller beam size, which has the full width at half power (FWHP) for the antenna with diameter D, operating at the wavelength λ can be approximated by

$$\Omega_b = \pi \left(\frac{\theta}{2}\right)^2 = \pi \left(\frac{\lambda}{2D}\right)^2. \tag{2.5a}$$

$$\theta \approx \frac{\lambda}{D} \tag{2.5b}$$

$$\mathrm{FOV} = N_b \times \Omega_b \tag{2.5c}$$

FRBs are relatively bright phenomena and it may therefore be possible to sacrifice the sensitivity in order to obtain a larger area on the sky. The beam size however cannot be too large, as it would make any accurate localisation difficult. The bandwidth can be increased, but this has its practical limitations as well, both in the size of the receiver systems and cost-effectiveness. The system temperature can be decreased, especially when a cryocooled system in introduced, but this can also involve significant investments in terms of development time and costs. There are also limitations on how much radio astronomy systems can be cooled down and it is not possible to fully eliminate all the noise components introduced into the receiving system.

Therefore, one of the most practical and easily achievable ways of increasing the survey speed is using a larger number of beams. This exploits the fact that the off-axis rays are not focused in the same point as on-axis rays, but are instead focused in a plane around the axis of the dish. In order to observe a larger portion of sky at any given time, multiple receiving elements can be placed in different parts of the focal plane, which will then collect the waves arriving at different points (Baars & Swenson, 2007). Traditionally, this has been achieved by building receivers with multiple feed horns. This solution quickly becomes impractical with the increasing number of

beams, as due to their size and weight, only a limited number of waveguides can be installed. The most notable multi-beam systems include the Parkes Multibeam receiver with 13 beams (Staveley-Smith et al., 1996) and Arecibo L-Band Feed Array (ALFA) capable of generating 7 beams on the sky (Cordes & Goldshith, 2001). The number of beams is also limited by the loss in gain and various beam aberrations, such as coma and astigmatism, which tend to increase for beams further away from the axis (Baars & Swenson, 2007; Padman, 1995). Additionally, the standard feed-horn based receivers do not provide the flexibility in terms of beam separation or adaptive RFI mitigation, where the shape of the beam can be changed during the beamforming process to decrease the beam response in the direction of known RFI (Landon et al., 2010). Multiple pointings with small offsets have to be used to form a mosaic with a continuous sky coverage, as horns cannot be packed more closely than their physical size allows, often resulting in significant gaps between the beams on the sky.

Phased Array Feeds (PAFs) are an attempt at overcoming these limitations. Instead of large horns they use a large number of closely-packed small antennas, e.g. dipoles, that can fully sample the radiation in the focal plane. All of these elements see a different portion of the sky and their individual inputs can then be combined to form multiple beams on the sky. PAFs have been in use for decades now in various systems including radio communications and radar. Their use in radio astronomy has been limited due to more stringent requirements, such as lower system noise temperature and large operational bandwidth. The last couple of decades have seen a rapid development and great improvements in the radio astronomy PAF receiver technology, mainly thanks to the involvement of multiple large research institutes, such as NRAO in the USA and CSIRO in Australia. Even after this period of rapid development, some problems still remain and they will have to be solved before PAFs can fully replace the traditional receivers. These drawbacks include the difficulty of developing a cryogenically cooled system, which can significantly increase the development cost and the size and weight of the final receiver, but a large amount of research is currently being invested in this area specifically, with a cryocooled PAF currently being commissioned on GBT (Roshi et al., 2015) and new design under development at CSIRO and JBO (Liu & Grainge, 2017). Recent improvements should soon bring the system temperature down from 50–75 K, to levels comparable with the feed-horn based receivers currently in the operation at 20–25 K (Staveley-Smith et al., 1996). The small spacing of the array elements is also a significant problem, increasing the noise levels as the thermal radiation from each element leaks into the surrounding elements.

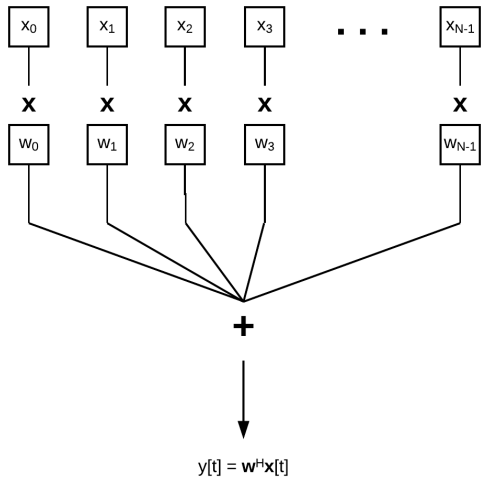


Figure 2.3: Path of single time sample signal x_n. Signals from each element are multiplied by their corresponding weights w_n and summed together, resulting in the beamformed output y.

Despite these problems, Phased Array Feeds already offer a great alternative when the benefits coming from the increased instantaneous field-of-view outweigh the reduced sensitivity. Large projects such as the Australian SKA Pathfinder (ASKAP) (Johnston et al., 2007), consisting of 36 12 m diameter antennas, each with a PAF mounted in its focus, are currently surveying the sky to prove that this technology can be efficiently deployed on larger systems (McConnell et al., 2016). With the first FRB already discovered with ASKAP (Bannister et al., 2017) it appears that the Phased Array Feeds will indeed revolutionise radio astronomy by simply increasing the sky coverage and providing larger snapshots of the sky.

2.3.1 Theoretical Description of Beamforming

As mentioned above, signals originating at each of the receiver elements have to be properly combined during the beamforming operation to obtain the desired beam footprint. The signal at time t recorded by N array elements can be approximated by a vector

$$\underline{\boldsymbol{x}}[t] = \underline{\boldsymbol{v}}_s \underline{\boldsymbol{s}}[t] + \underline{\boldsymbol{n}}[t], \tag{2.6}$$

where $\underline{\boldsymbol{v}}_s$ is the normalised array response to a unit amplitude point source in the far field direction, $\underline{\boldsymbol{s}}[t]$ and $\underline{\boldsymbol{n}}[t]$ are signal and noise vectors respectively, and each of these vectors have N components - one for each receiving element as shown in Figure 2.3.

The beamforming procedure requires a vectors of weights to be calculated which are then multiplied with the incoming signal, resulting in the output

$$\underline{\boldsymbol{y}}[t] = \underline{\boldsymbol{w}}^H \underline{\boldsymbol{x}}[t], \tag{2.7}$$

where the superscript H denotes the conjugate transpose of a matrix. One commonly used beamforming process is applied to determine a set of weights that maximise the signal-to-noise ratio of the recorded signal. A different approach can also be used that prioritises the control over the beam pattern, to better accommodate for known RFI sources, which results in a generally lower sensitivity.

For the case of a stationary random process with zero mean and statistically independent signal and noise contributions, the array covariance matrix (ACM) can then be defined as

$$\underline{\underline{\boldsymbol{R}}} = E\left\{\underline{\boldsymbol{x}}\,\underline{\boldsymbol{x}}^H\right\} = \underline{\underline{\boldsymbol{R}}}_s + \underline{\underline{\boldsymbol{R}}}_n = \sigma_s^2 \underline{\boldsymbol{v}}_s \underline{\boldsymbol{v}}_s^H + \sigma_n^2 \underline{\underline{\boldsymbol{I}}}, \tag{2.8}$$

where $E\left\{\underline{\boldsymbol{x}}\,\underline{\boldsymbol{x}}^H\right\}$ denotes the expectation value, $\underline{\underline{\boldsymbol{R}}}_s$ and $\underline{\underline{\boldsymbol{R}}}_n$ are the signal and noise covariance matrices respectively, σ_s^2 and σ_n^2 are the signal and noise powers respectively, and $\underline{\underline{\boldsymbol{I}}}$ is the identity matrix. In practice, the estimate of $\underline{\underline{\boldsymbol{R}}}$ is calculated over an L-samples long integration

$$\hat{\underline{\underline{\boldsymbol{R}}}} = \frac{1}{L} \sum_{t=0}^{L-1} \underline{\boldsymbol{x}}[t]\, \underline{\boldsymbol{x}}^H[t]. \tag{2.9}$$

Depending on the intended use, the integration length can vary. It can be calculated over the span of seconds in the case of adaptive beamforming, which is used to reduce the negative effects of rapidly varying RFI by dynamically changing the shape of the beams on short time scales to decrease the gain in the direction of the offending RFI source. If such rapid changes are not required, and a more constant and stable beam pattern is preferred, the integration time can stretch over minutes or even hours at a time.

Following a rigorous mathematical derivation (Applebaum, 1976; Jeffs et al., 2008), it can be shown that the optimal weights, resulting in the maximum S/N, can be obtained by computing the ACM while pointing the receiver at a strong source, $\hat{\underline{\underline{\boldsymbol{R}}}}_{n+s}$ and at an empty patch of sky, $\hat{\underline{\underline{\boldsymbol{R}}}}_n$ and solving for an eigenvalue equation

$$\left(\hat{\underline{\underline{\boldsymbol{R}}}}_{n+s} - \hat{\underline{\underline{\boldsymbol{R}}}}_n\right) \hat{\underline{\boldsymbol{v}}} = \lambda \hat{\underline{\boldsymbol{v}}}, \tag{2.10a}$$

$$\underline{\boldsymbol{w}}_k = \underline{\underline{\hat{\boldsymbol{R}}}}_n^{-1} \underline{\boldsymbol{v}}_{1k}, \tag{2.10b}$$

where $\boldsymbol{v}_{1k}$ is the dominant solution to the above eigenvalue equation. The number of weight vectors depends not only on the number of beams, but also on the number of frequency channels processed by the beamformer, as each coarse channel will have its own weights. In total, there will be $2 \times N_b \times N_{ch}$ weight vectors provided, where the factor of two comes from the fact that 2 polarisations have to be processed separately for dual-polarisation system. The weight vectors for each pointing then have to be adjusted to phase changes over the entire band

$$\underline{\boldsymbol{w}}_p = \exp\left(-i\phi\right) \underline{\boldsymbol{w}}, \tag{2.11a}$$

$$\phi = arg\left(\underline{\boldsymbol{w}}^H \cdot \underline{\boldsymbol{w}}_r\right), \tag{2.11b}$$

where $\boldsymbol{w}_r$ is the weight vector for a reference channel (McConnell et al., 2016).

2.3.2 Practical Beamforming Procedure

The practical beamforming procedure for PAFs can be summarised as follows:

1. Select the directions of the beams.
2. Obtain $\underline{\underline{\hat{\boldsymbol{R}}}}_{n+s}$ by pointing the antenna towards the bright source, for each of the beam directions.
3. Move the antenna to a cold portion of the sky, preferably in the azimuth, to avoid changing contributions to noise from the ground, and record the $\underline{\underline{\hat{\boldsymbol{R}}}}_n$.
4. Use Equations 2.10a and 2.10b to obtain the weight vectors and modify them according to Equation 2.11a.
5. Load the resulting weight vectors $\boldsymbol{w}_p$ into the beamformer for subsequent observations.

A bright astronomical source is required to perform reliable beamforming. Even when no rapid RFI mitigation is being used, it is recommended to repeat the beamforming process over the course of days or weeks to account for variations such as gain instability and changes in the receiver electronics systems.

Figure 2.4 shows the example of beam weights in 2.4a. For this particular setup, only a single beam was generated and is shown in Figure 2.4b. The shape of the beam was confirmed by scanning a bright point source, in this case quasar 3C268. It is important to note that Figure 2.4 shows only a single 'cross-section' of the entire frequency domain, as weights are generated separately for every channels. This in turn results

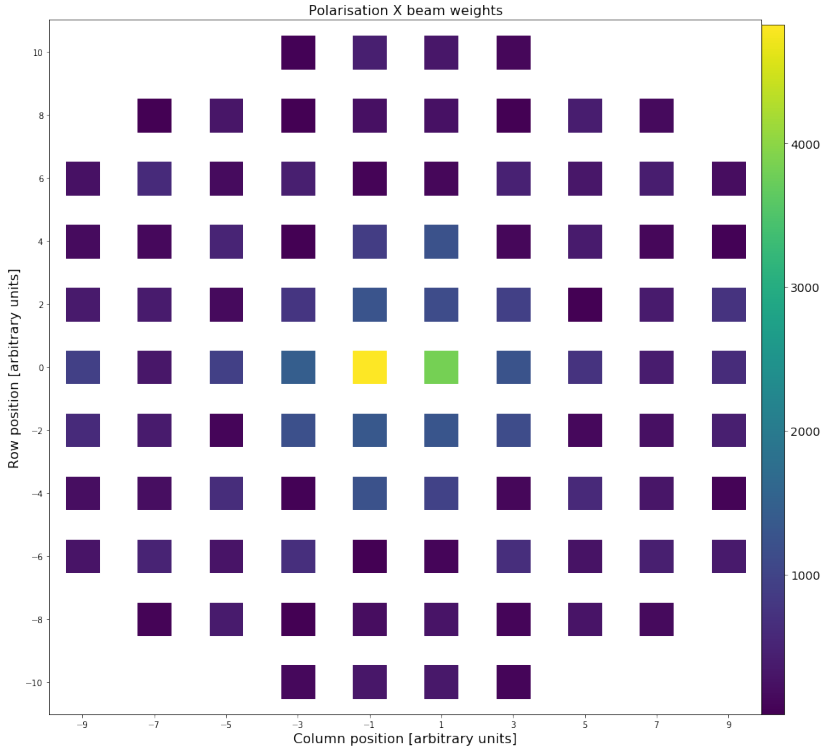
in a changing beam pattern, which always changes with changing wavelength, as described in Equation 2.5b, but now also depends on the computed beam weights, which can significantly influence the shape of the beam. This is especially important if the adaptive RFI mitigation is employed and only certain directions for certain channels have to be masked. The overall shape of the beam and its variations with frequency become important when an accurate localisation of detected source is necessary.

2.4 GPU and Parallel Computing

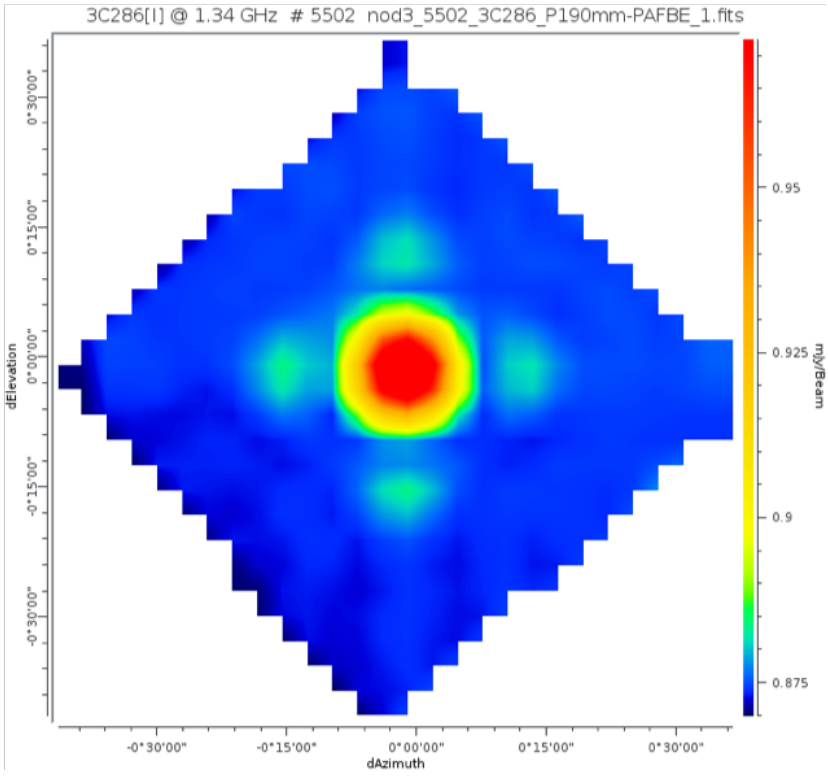
The history of the GPU computing begins with the introduction of the first graphics chips in the 1970s. At that time, consumers had access only to cheaper and simpler designs, as chips capable of handling 3D graphics were reserved for professional use only due to their complexity and resulting high prices. The first affordable 3D graphics cards were introduced to the PC market in the mid-1990s. At that time, these were still relatively simple, with a large number of fixed-function stages, each responsible for performing a single, specific task. Programmers were able to configure certain parameters, but functions themselves could not be programmed and offered a limited flexibility. This severely limited the usefulness of GPUs beyond the task of performing graphics calculations, and the general Central Processing Units (CPUs) were used for any complex computations. The fixed-function pipelines consisted of a number of stages, each of them being a task-specific piece of hardware (Kirk & mei Hwu, 2010).

The first programmable functions were introduced in 2001 and 2002 to the vertex and shader stages respectively. The general idea of a pipeline with each stage devoted to a specific task however remained unaltered. The first major change came in 2006, with the introduction of Nvidia GeForce 8800. It was the first GPU which combined all the stages of the programmable graphics pipeline into a collection of unified processors, all with the same capabilities (NVIDIA Corporation, 2006). These have lower clock speeds than current generation CPUs (up to 1.5 GHz compared to more than 4.2 GHz per core in consumer CPUs), but the strength of GPUs comes in the large number of parallel cores, greatly exceeding 8, 16 or even 32 present in the high-end CPUs currently available.

The GPUs of today are massively parallel processing units, with the number of cores increasing all the time and now exceeding 3000 per GPU. The instructions are executed and managed on Nvidia GPUs using the Single-Instruction, Multiple-Threads architecture (SIMT), similar to the more common Single-Instruction, Multiple-Data



(a) Absolute values of complex beam weights for single polarisation, calculated using all elements for a single frequency channel at 1340 MHz.



(b) The resulting beam shape obtained by scanning a point source, in this case a quasar 3C286.

Figure 2.4: Application of the beamforming procedure, with the resulting beam weights used to form a single beam.

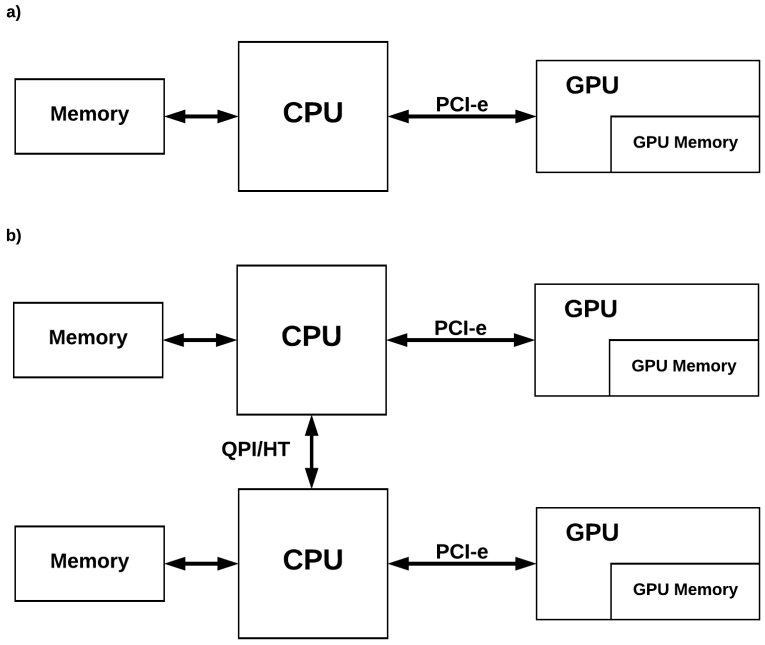


Figure 2.5: A simplified view of a) a single-CPU, single-GPU system, b) a NUMA system with two CPUs and two GPUs. Quick Path Interconnect (QPI) or HyperTransport (HT) have to be used to transfer the data between different CPUs in the NUMA-aware configuration.

(SIMD). In simple terms, SIMT can be thought of as the extension of SIMD, with the emphasis shifted slightly towards increased flexibility, which incurs a small cost in efficiency. In SIMD, multiple data points are processed simultaneously in parallel (Multiple-Data), using the same operation (Single-Instruction). SIMT expands on this idea and incorporates the SIMD model into the massively parallel environment of GPUs. The instructions are 'bound' on a thread-by-thread basis which are then dynamically executed in a SIMD fashion. This is especially important when writing code with branches. In the most recent architecture used by Nvidia, threads are executed in groups of 32, called warps. If, e.g. an `if ...else` statement, results in divergent branches, threads are scheduled to be executed in groups in a SIMD manner, each group following a given branch, in a sequential manner, i.e. the `else` branch threads have to wait for `if` branch threads to finish first before they are allowed to run. This can lead to significant losses in performance and it is important to ensure that all threads within a warp can take the same data path, effectively ensuring a warp-wide SIMD execution and maximum performance gains.

In most practical applications, the GPU does not have direct access to the host

RAM as it is treated as a separate device, as seen in Figure 2.5. Even in the advent of the Unified Memory Addressing, when the memory addresses can be shared between the host and device memories, the GPU memory is still physically separated from RAM, and any data flowing in or out of the GPU has to be transferred over the PCI-e bus. At the time of writing most modern systems use PCI-e 3.0, capable of delivering the theoretical throughput of 16 GBps for a 16 lane configuration, with the real-life limits closer to 12 GBps. This can be a problem for certain applications that consume large amounts of data and do not perform too many complicated operations, especially when coupled with a relatively small amount of memory available on the GPU (up to 12 GB at the time of writing). The situation is complicated even further when the Non-Unified Memory Architecture (NUMA) is in use. Here, the memory can be split between multiple CPUs within the system and the memory access time depends on whether the CPU is accessing its local memory. If the memory being accessed is local to a different CPU, it has to be transferred through the Quick Path Interconnect (QPI) in terms of Intel and HyperTransport (HT) for AMD processors, directly connecting the CPUs, which can incur performance penalties. GPUs can also be connected into different NUMA nodes as seen in Figure 2.5 and the data transfer rates can see a significant impact when the additional CPU-to-CPU transfer is necessary.

2.4.1 Compute Unified Device Architecture - CUDA

Even after introduction of the unified processing unit architecture, performing scientific calculations on GPUs was still a challenge. All the problems had to be translated into language of graphics operations which made writing programs long and complicated. CUDA was introduced by NVIDIA in 2007 alongside Tesla C870 - the first GPU aimed directly at the General-Purpose computing on Graphics Processing Units (GPGPU). It was the first time when a GPU was treated like a general-purpose processor, not just a circuit that can only generate computer graphics. The most popular version, CUDA C/C++, makes use of the standard C/C++ with added extensions, to enable GPU programming. In software making use of CUDA, work can be done on the host, which is a standard CPU, or device, which in this case is a GPU. Execution on the device is performed by the means of kernel launches. Each kernel can access the processing cores and memory available on the GPU and use them to run parallel computations.

The generalised processing path during the concurrent CPU and GPU utilisation is show in Figure 2.6. The memory is first allocated on the GPU and then copied

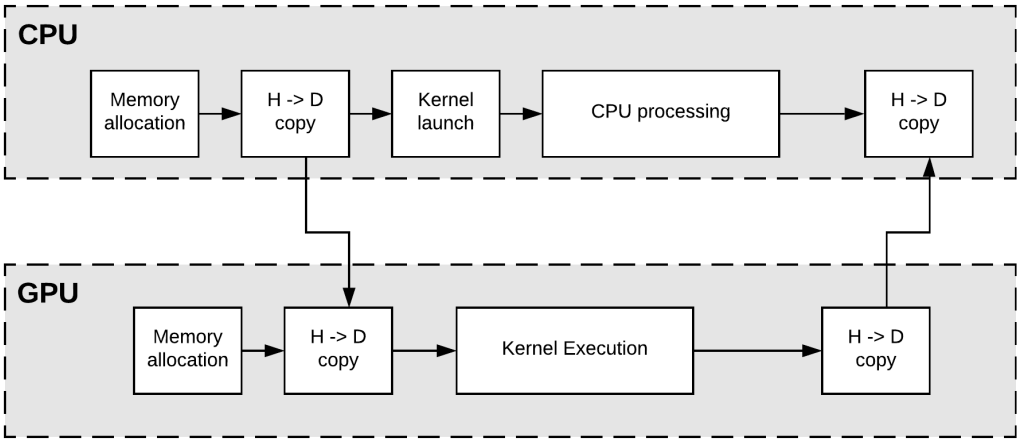


Figure 2.6: The basic workflow for the GPU processing with a concurrent CPU execution.

across from the host memory to the device memory. The kernel then does the work and is called in the same way as ordinary C++ functions, with the additional kernel parameters, such as the number of threads and blocks. The kernel launches are asynchronous, which means the control is returned to the CPU as soon the GPU driver is done with starting the job. This allows for concurrent processing by the GPU and the CPU, meaning that as the data is being processed by the GPU, the CPU can spend this time preparing another piece of data for the processing and launching another kernel. One important performance consideration is that it takes a non-negligible amount of time to launch a kernel, around 5ms, and this concurrent execution scheme does not offer any performance benefits if the time it takes to process the data is shorter than the time required to launch the kernel. In such a scenario, the GPU spends some time being idle, waiting for another kernel launch. Calls to `cudaMemcpy()` are implicitly synchronised which ensures that the data will be processed and made available before it is copied back to the host RAM. Together with CPU/GPU concurrency, modern CUDA allows for concurrent processing of multiple kernels on the GPU as well. As can be seen in Figure 2.7 spreading the processing over many kernels makes it possible to overlap the memory copies and processing, effectively hiding the data transfer latency. With the current architecture, up to 16 kernels can be run concurrently and this number will most likely increase in the future.

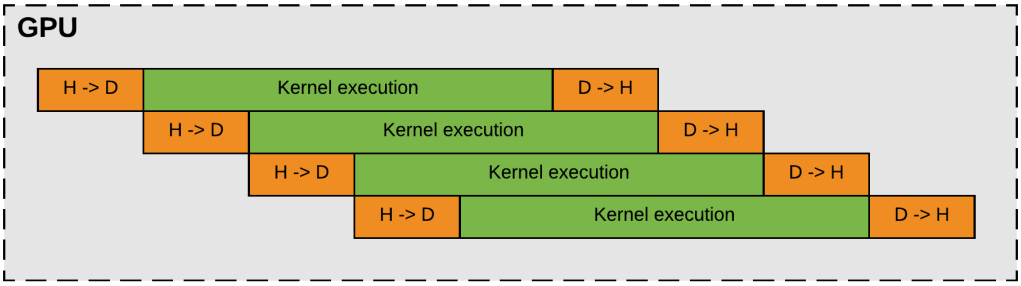


Figure 2.7: Multiple GPU kernels (green) can run concurrently, overlapping with host to device and device to host data transfers (orange).

Chapter 3

PAFINDER - Phased Array Feed FRB Finder

3.1 PAF at Effelsberg

The phased array feed installed at the Effelsberg radio telescope uses the modified ASKAP Mark II design. This particular design contains 188 chequerboard feeds (Brown et al., 2014). The signal is amplified and filtered inside the PAF frontend and is then transported to the backend for beamforming, with the main components of these two stages shown in Figure 3.1. The main elements of the backend include the digital receivers (DRXs) and beamformers (BMFs). The digital receivers are tasked with channelising the incoming data into 336 oversampled '1 MHz' channels. While the centres of these channels are separated by 1 MHz, they are sampled at a frequency of $^{32}/_{27}$ MHz. This causes the channels to overlap at the edges of the band, which provides a flatter frequency response across the band and also reduces the negative effects of aliasing (Tuthill et al., 2012). This coarse filterbank is then passed into the beamformer. Each beamformer chassis contains 6 Field Programmable Gate Arrays (FGPAs) and each FPGA receives 7 channels. This means that each beamformer is responsible for processing 42 '1 MHz' channels and forming all the beams across this part of frequency band. It is therefore necessary to combine the outputs of all 8 beamformers to obtain the full 336 MHz frequency coverage. The beamformed output is then streamed to a network switch which is responsible for distributing the data throughout the network and towards the compute nodes.

In total, the Mark II system is capable of generating up to 36 dual-polarisation beams on the sky. That gives us an instantaneous FOV of 2.06 deg^2 at the frequency

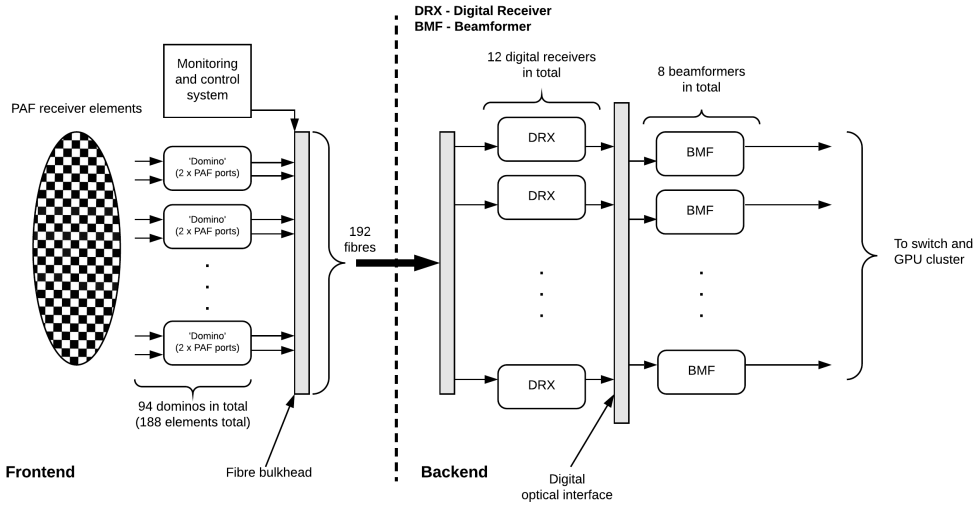


Figure 3.1: The overview of PAF frontend and backend.

of 1400 MHz, when the centres of the beams are separated by FWHP which is ~ 1.15 times the FOV achieved using the 13-beam Parkes Multibeam system. This is the upper limit on the area that can be covered by 36 beams, and the FOV will be smaller in practice, as the beams can and usually will overlap, with their centres separated by less than FWHP.

3.1.1 Processing Facilities

The processing is run on multiple nodes, housing off-the-shelf hardware to allow for efficient and cost-effective processing. It is important that these facilities meet the strictest requirements and are able to process and store immense volumes of data from all the beams in real time. The nodes specification is shown in Table 3.1. With the current hardware and software implementations of the pipeline, we are capable of processing the data from a single beam per NUMA node. As each of the compute nodes has 2 NUMA nodes, we were therefore capable of running the pipeline over 18 beams, using 9 compute nodes that were installed and available at the time of the initial commissioning.

CPU	2 × Intel(R) Xeon CPU E5-2630 v4, 2.20 GHz, 10 physical cores
RAM	128 GB, 64 GB per NUMA node
GPU	2 × Titan X (Pascal), 12 GB memory
NIC	2 × Mellanox 40 Gbps

Table 3.1: Specification of PAF processing nodes.

3.2 PAFINDER Pipeline Design and Implementation

The PAFINDER pipeline has been developed to enable real-time processing of the PAF data. Even though the main focus is enabling a real-time FRB detection and localisation, the pipeline has been written in different possible uses in mind, such as pulsar search, which can have a different set of requirements. Clear separation of different processing modules and the use of tools, modules and libraries commonly used in radio transient data processing and searches allows for easier modifications and upgrades to the existing pipeline.

The work done by the pipeline can be divided into 3 major stages: receiving the data, creating the filterbank files and searching for single pulses. The generalised workflow is shown in Figure 3.2. From this layout it can be seen that most of the work in creating the actual filterbank file, including unpacking, the FFT, scaling and averaging, is done by the GPU. The CPU is responsible for receiving the data from the beamformer, managing the pipeline, which includes dispatching the work from internal buffers to the GPU, monitoring the state of the pipeline and reporting back on any errors and problems that can occur during the processing.

Due to the limited work done by the CPU on the actual data, the CPU/GPU boundary is crossed only twice and the data stays on the GPU for the majority of the processing and only the final averaged and scaled filterbank product is sent back to RAM to be saved to disk and sent for further processing. The initial version of the pipeline utilised a more monolithic approach, where the single pulse processing was embedded into the pipeline and the data stayed on the GPU the whole time. This approach was however abandoned to allow for greater flexibility in terms of search backends. This means that different software can be used to process the data, e.g. a pulsar search software can now be used instead of the single-pulse search pipeline if needed and if it supports the data format used by the pipeline. More modular approach also resulted in a less complex pipeline design, which made it easier to track and fix any possible bugs in the code.

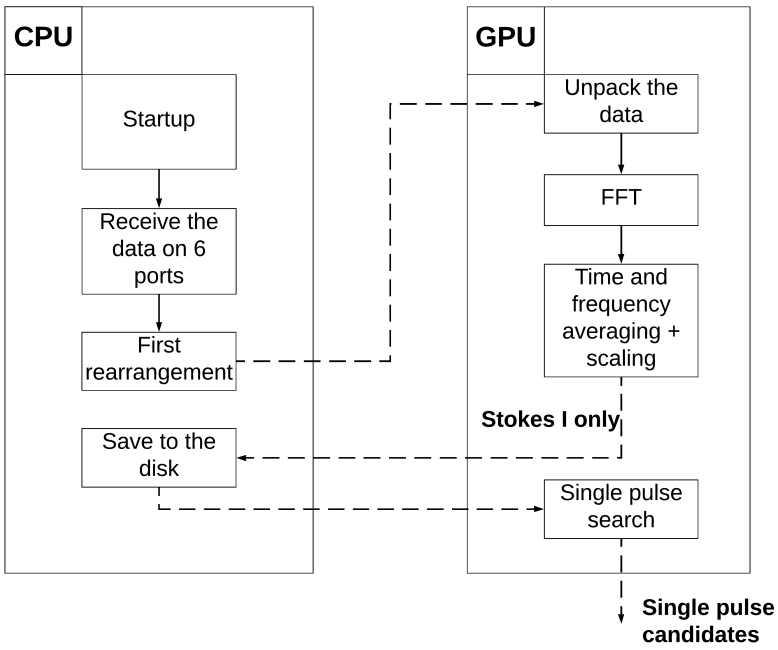


Figure 3.2: Basic components of the PAFINDER pipeline. Dashed lines represent boundaries where the data has to be copied from the host memory to device memory and vice versa. Single-pulse search part of the pipeline is performed using the first Stokes parameter only - the absolute power, with the future plans to store raw voltages to enable full polarisation analysis.

Each stage has different processing and memory requirements and has to be carefully designed, benchmarked and optimised to achieve enough data throughput that will allow for FRB surveys to be run in real time. Figure 3.3 shows all the memory buffers used during the processing, including their memory requirements and whether they are allocated on the host or GPU device memory. This proved to be one of the most crucial and challenging steps in the pipeline design process. The size of each buffer, its role in the processing and the arrangement of the data had to be carefully considered and implemented to make sure we efficiently used our resources and introduced as little complexity as possible. As the buffers further down the processing line depend on the input they receive from the earlier parts of the pipeline, it is also important that we are able to move the data between them in an efficient manner and coordinate all the code execution to prevent the data corruption and stalls caused by data races. The correct ordering of the data and execution of various parts of the pipeline is achieved with the help of multiple techniques, depending on the requirements of the processing stage and described in more details in the following sections.

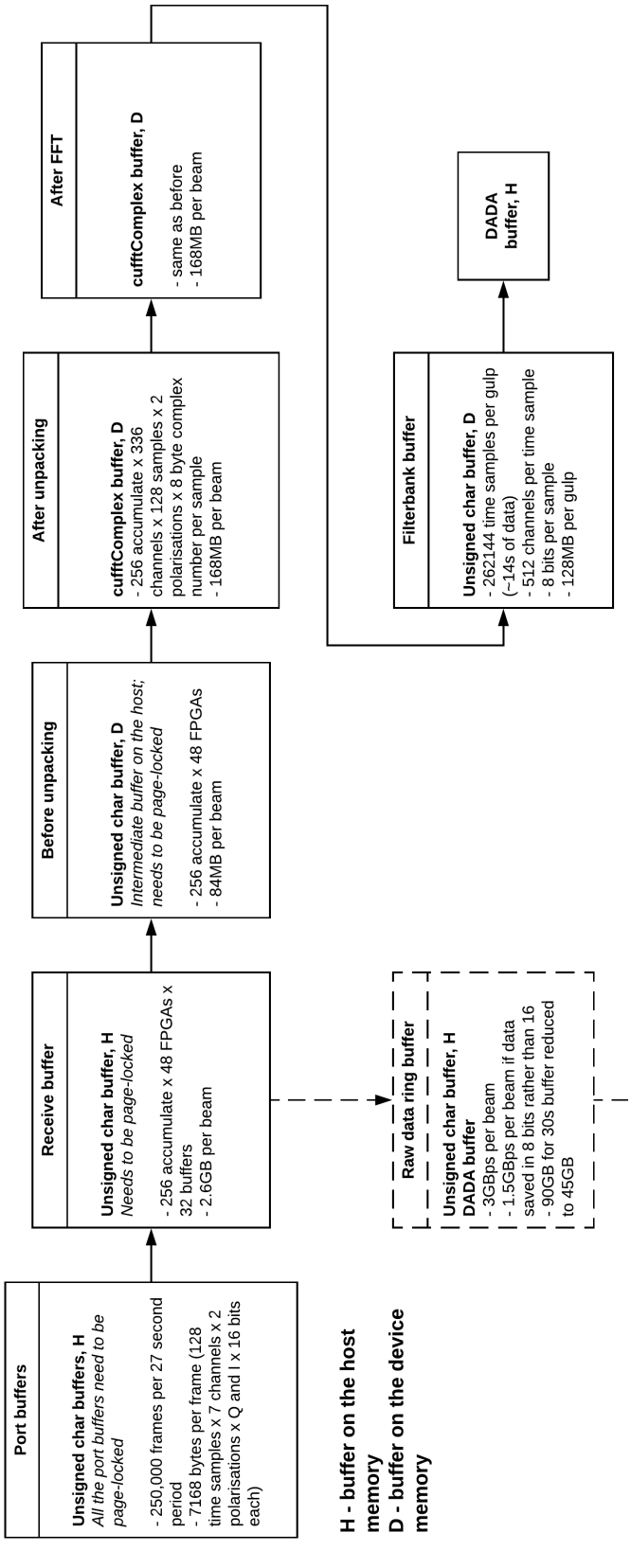


Figure 3.3: Buffers used for different stages of the processing pipeline. Dashed lines represent the raw data buffer portion of the pipeline that currently cannot be implemented due to the lack of a sufficient amount of host RAM.

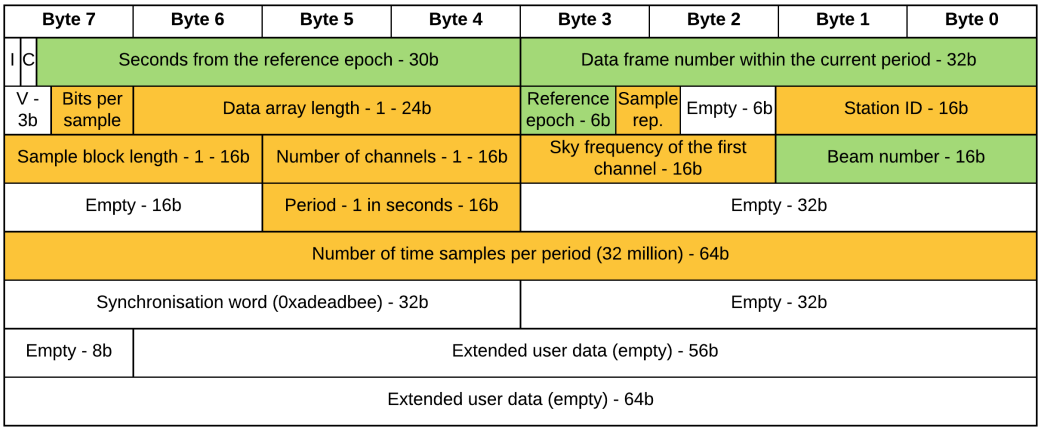


Figure 3.4: Header of the CODIF data frame. Highlighted in green are the parts that are crucial to running the pipeline and recalculated for every packet received. In yellow are parts that are used during the processing, but do not change during or between the pipeline executions and are therefore hardcoded. Credit: CSIRO Astronomy and Space Science

3.2.1 Receiving the Data

The data for a single beam arrives on 6 network ports, each receiving channels from 8 FPGAs. A CSIRO Oversampled Data Interchange Format (CODIF) has been designed and implemented specifically for the task of transporting the data from the ASKAP Phased Array Feeds. Each packet contains a 64 byte-long header, shown in Figure 3.4, followed by a data frame containing 128 time samples for each one of the 7 channels, shown in Figure 3.5. Each time sample contains two polarisations and each polarisation is composed of two complex parts, resulting in the 32 bits per time sample per polarisation - 64 bits for a complete time sample that has to be received and processed. With 6 ports, 8 FPGAs per port and 7 channels per FPGA, we receive the total bandwidth of 336 MHz. Together with the high 16 bit sampling, this results in a combined data rate of 25 Gbps per beam.

Only some information contained in the header is used in the processing, as shown in green in Figure 3.4. Some fields, shown in yellow, do contain important information, but they do not change when running the pipeline as part of the same system and therefore can be hardcoded, which allows us to skip the explicit calculations of these values and to speed up the processing. The most important variables that are used and recalculated for every frame include the reference epoch, the number of seconds from the reference epoch and the frame number within the current period, which combined

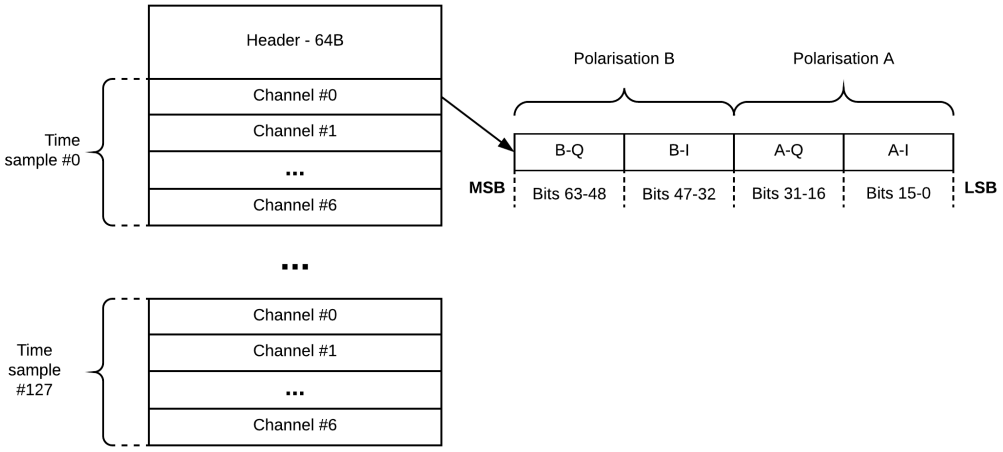


Figure 3.5: Packing of the data into the CODIF frame. Each of the 128 time samples contains data from 7 beamformed channels. Including the header, the size of a single data frame is 7232 bytes. Credit: CSIRO Astronomy and Space Science

together give us the absolute time of samples contained in the data block. The frequency order is obtained from the FPGA ID, with each one of the 7 MHz wide blocks assigned a unique number between 0 and 47 and obtained from the third and fourth octets of the sender's IP address. Having the arrival time and frequency of each data block allows us to place these in the correct order in the receiver buffer. Additionally a beam number is obtained at the start of the observation, as due to the networking and compute node issues we cannot guarantee that nodes will receive data from the same beam every time.

After the packet has been received, and time and frequency offsets have been calculated, the data are placed in a buffer, where they undergo the preliminary rearrangement which puts them in the required time and frequency order. At the data receiving stage, the pipeline uses 6 threads, bound to 3 CPU cores for optimum performance and lowest possible packet loss. Having multiple threads saving the data to an array shared between all of them can cause unexpected results to occur when not enough care is taken to safeguard against the effects of data races. To avoid any problems of this kind, the position of the data within the buffer is determined from the frame number and the FPGA ID, which means that each thread uses only the portion of the buffer that 'belongs' to it and other threads do not write any data there. This removes the need for any explicit blocking mechanisms, such as mutexes, which can incur a non-negligible performance penalty and should be avoided at this crucial stage.

To avoid too many expensive copies, the data is sent into the GPU memory for processing in chunks of 128 or 256 data frames for each FPGA. With 128 time samples per data frame, each 256 frame-long data chunk spans a time of ≈ 27.6 ms. An additional buffer for keeping the track of received packets is used where each time sample has a 64 bit atomic variable associated with it. After the data from a given FPGA is received, the corresponding bit in this variable is set to 1, marking the data as received. The data chunk is added into the consumer queue after at least 75% of all the FPGAs have sent their data, i.e. at least 36 out of 48 bits that we consider are set to 1. This allows the pipeline to work correctly even if large packet loss of 25% is present, which can happen if the beamformer FPGAs are not initialised properly, but also ensures we receive the majority of the data, which reduces the possibility of insufficient amount of data and/or corrupted data. The end of the current buffer and the quarter of the next buffer is checked for being filled to accommodate for any out-of-order packets. Any packets that arrive after their buffer has been sent for processing are simply discarded.

After the producer thread which receives the data puts it in the queue, the thread responsible for running the work on the GPU ('the worker thread') is woken up, as shown in Listing 3.1. Access to the queue is protected by a mutex and a condition variable. Wrapping the whole operation in a mutex means that only one thread can modify the data queue at a time - if not protected this can lead to an undefined behaviour as the order of operations on the queue cannot be guaranteed. The condition variable `workready_` allows the thread waiting for the data to sleep when the queue is empty and there is nothing to be done. It can then be woken up by the producing thread with the `workready_.notify_one()` call. The working thread then resumes the operations and checks whether the data has really been added to the queue (unexpected spurious wakes, even when the queue is empty, are a possibility) and whether the pipeline is still working (this is necessary to make sure the worker thread does not wait indefinitely for the data after the processing is finished). The data is then copied into the GPU memory - this is the last place the data is stored in the host machine's RAM. The next time the data 'leaves' the GPU and is copied back to the host memory is when the final filterbank is ready.

Listing 3.1: Mechanism for sending the data to the GPU for processing using a simple STL queue protected by a mutex.

```
// Thread consuming the data
    std::unique_lock<mutex> worklock(workmutex_);
    workready_.wait(worklock, [this]{return (!workqueue_.empty() ...
        || !working_);});

    if (working_) {
        bufferinfo = workqueue_.front();
        workqueue_.pop();
        worklock.unlock();

// Thread producing the data
    workmutex_.lock();
    workqueue_.push(std::make_pair(hinbuffer_ + istream * ...
        inbuffsize_, refframe));
    if (workqueue_.size() > 1) {
        cerr << "WARNING: GPU is not keeping up with the work! (" ...
            << workqueue_.size() << ")" << endl;
    }
    workmutex_.unlock();
    workready_.notify_one();
```

To avoid unnecessary copies, not the data, but the pointer to the position in the buffer where the data is stored is placed onto the queue. This is necessary, as copying the contents of the buffer has been found to be inefficient and did not meet the real-time requirements during the testing. Passing the pointer instead of the actual data poses a risk that if the GPU processing slows down, there might be different data in the buffer by the time the worker thread copies the data to the GPU. To avoid this situation, the receiver buffer is sufficiently large as to accommodate for a reasonably large stall in the processing, which during multiple tests have been found to be as large as 0.5 s when the GPU driver is restarted. The current implementation uses 32 buffers, combined into one long ring buffer, meaning that the GPU processing can have an instantaneous lag of more than 880 ms before the data starts being overwritten. During the extensive testing, we found that in the most exceptional cases, there are less than 20 elements in the queue at most and larger values (up to a few hundred) occur only during node or network failure, at which point the processing has to be restarted anyway.

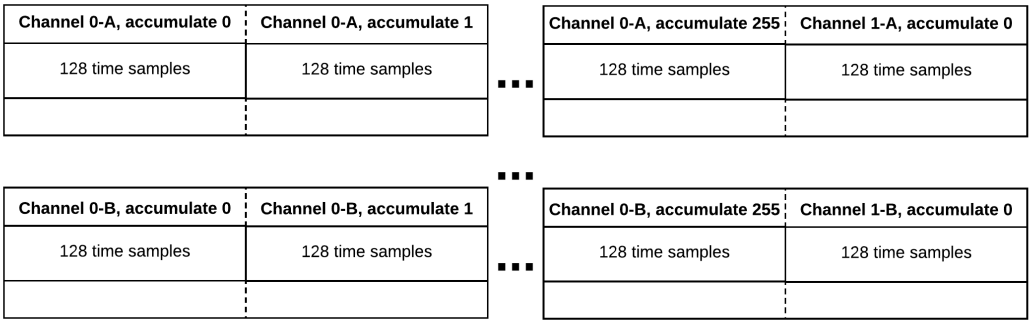


Figure 3.6: Data representation after the unpacking.

3.2.2 Unpacking

The data is packed into the CODIF frame as shown in Figure 3.5. It has to be unpacked and combined into single-precision, floating-point complex numbers, which can later be passed into and processed by the CUDA cuFFT functions for calculating the FFT. The data structure has an unfriendly stride of 7 64 bit words between the successive time samples within the channel. This can cause problems with dividing the work between the threads and extra care has to be taken to avoid non-coalesced memory access (please see Section 3.4 for optimisation details of the unpacker kernel).

The single data stream is split into two, one for each polarisation. The data still resides in a single array, but with each half of the array storing separate polarisation. It is arranged in the order that maximises the performance of the processing stages further down the line which allows for easier optimisation and is shown in Figure 3.6. The data is arranged with the 128 samples from a single data frame, followed by all the time samples for a single 1 MHz channel for a data buffer containing 256 time accumulates. This is then repeated for all 336 1 MHz channels and both polarisations. After the unpacking, the size of the output buffer is twice the size of the input buffer, as the 16 bit values are cast into 32 bit single-precision floating-point numbers.

3.2.3 Filterbank

The process of creating the filterbank file is split into two stages: first a Fourier Transform in the form of FFT is run on the incoming raw voltage to obtain the channelised time series and then the data is ‘detected’, i.e. the total power is obtained. The resulting filterbank file can then be optionally time and frequency averaged, which decreases the resolution, but can reduce the output data rate, therefore enabling faster processing

and using less disk space when the data is stored offline. We run a batched version of the FFT, where a large number of independent FFTs is run at the same time. A complex-to-complex 32-point FFT is run on each of the 336 1 MHz channels. This means that for 32 input values, we receive 32 unique fine channels. The structure of the unpacking stage output buffer means that after the FFT we now have 4 output time samples, 32 channels each, for every 128 input time samples next to each other.

The detection kernel is responsible for performing 3 main tasks: it removes the unnecessary channels (oversampled channels obtained after the FFT of the original $^{32}/_{27}$ MHz channels), as well as reducing the final number of output channels to 512) and changes the frequency ordering, calculates the signal power and scales the data. In the first step 5 channels are dropped for every 32-point FFT to remove the initial oversampling. Currently we drop the first 2 and the last 3 channels and these values can be changed if necessary, but tests have shown that this results in the best frequency response. It is also necessary to change the frequency ordering in two cases. Firstly, for the individual 1 MHz channels, the halves of the spectrum have to be swapped, to place the negative frequencies in the right place. Secondly, after all the averaging is performed, the original frequency ordering in the data array, with the lowest frequency stored first, has to be swapped in order to follow the convention of placing the highest frequency first and using the negative channel bandwidth. The first Stokes parameter, the total power, is calculated for every time sample and channel - at this stage we lose all the polarisation information, as two polarisations are combined. An implementation that calculates all 4 Stokes parameters has been developed and tested in the past, but it was decided to use only the total power during the commissioning and first science observations.

A future upgrade to the pipeline has been drafted that will add the ability to store the raw data in memory as it is originally packed inside the CODIF data frames and save it when an FRB candidate is detected, making the calculation of all Stokes parameters unnecessary and making it possible to fully extract polarisation information during the post-processing. It is however impossible to implement this solution at the moment because of the RAM constraints, as the 30 s-long raw voltage buffer, the minimum we need to be able to store 14 s of data and allow for 14 s of processing, requires 180 GB of RAM for two beams, where only 128 GB is available on each compute node. One possible solution is to truncate the incoming data from 16 bits per sample down to 8 bits, but this can lead to the considerable loss of signal and will have to be investigated further, once the changes to the frontend gain levels are finalised and tested.

In the final step of the detection stage the power is averaged, both in time and frequency, resulting in the final time resolution of 54μs and the frequency resolution of 0.59MHz. Having the time samples next to each other, we can easily average in time first, and then in frequency. The frequency-averaged filterbank has 567 frequency channels, but that value is truncated to 512 in the final output filterbank. We drop 28 channels at the bottom of the band and 27 at the top, which allows us to discard frequency channels where the bandpass starts to drop off significantly, as can be seen in the top left panel of Figure 3.7. Removing 55 channels therefore results in the total bandwidth reduction from 336MHz down to 303MHz - a close to 10% loss. This is however necessary, as a lot of processing software has been written and optimised to work the best with power-of-two number of channels and some can even fail for various reasons, such us memory alignment requirements, when this condition is not met.

3.2.4 Scaling

The data has to be then scaled in order to reduce the dynamic range, and fit it into an 8 bit number, which means all of the power values have to be brought down into the range between 0 and 255. This is achieved by applying a simple scaling operation

$$x^O_{c,t} = \left\lfloor \frac{x^I_{c,t} - \bar{x}_c}{\sigma_c} \times 24.0 + 64.5 \right\rfloor, \tag{3.1}$$

where $x^O_{c,t}$ and $x^I_{c,t}$ are the output and input values for channel c and time sample t, $\bar{x}_c$ is the mean for the channel c, σ_c is the standard deviation and $\lfloor \ldots \rfloor$ is the floor operation. After this scaling is applied, the data has a mean of 64 and a standard deviation of 24. A number of time samples with values outside the desired range will always be present. In such a case, they are simply clipped and set to the minimum or maximum allowed value. The effect of scaling on the bandpass and the range of input and output values can be seen in Figure 3.7. We can see that both the scaled and unscaled data values follow an approximate normal distribution, with the unscaled distribution having a longer tail towards low values caused by the bandpass dropping off at the edges. The excess of 0's in the unscaled values' distribution is caused by the channels with missing data, which in turn results in a sharp increase in the scaled values' distribution around 0, as whenever there is missing data, these channels are automatically set to 0 in the scaled version. This slightly disturbs the properties of the

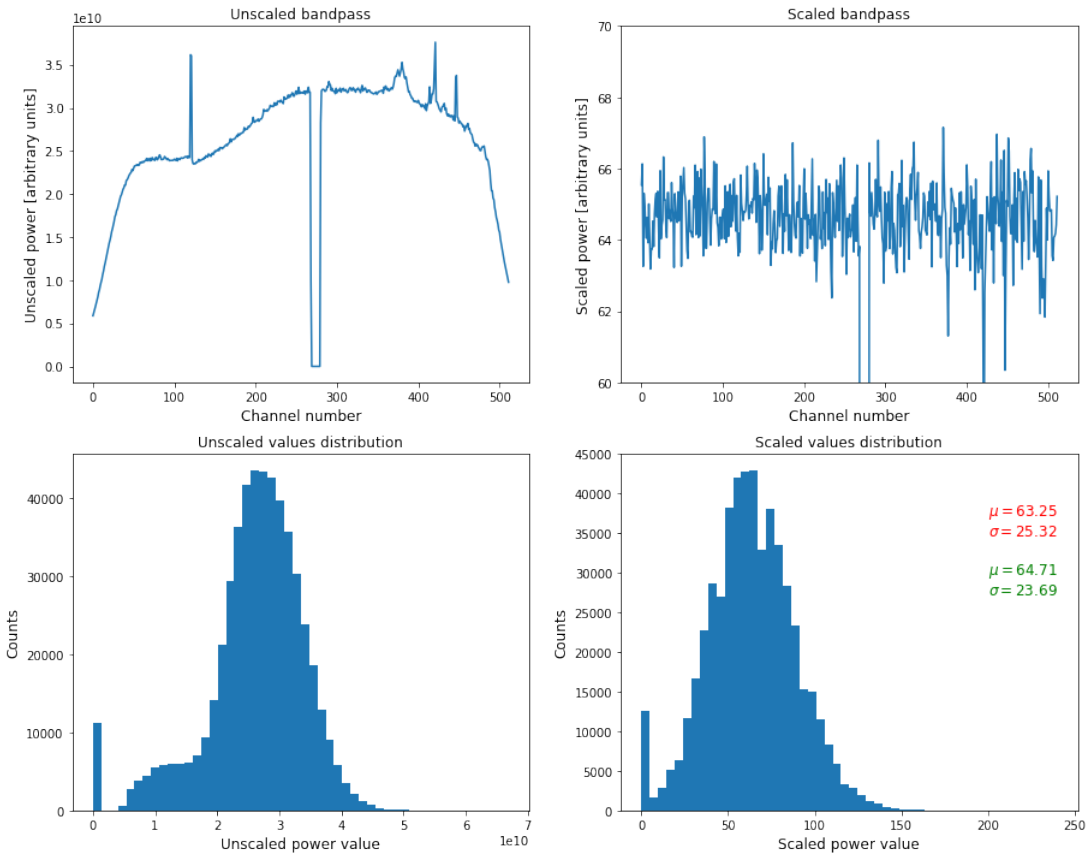


Figure 3.7: The effect of scaling on recorded signal. In the bottom right panel in red are shown properties of the distribution with missing data channels included and in green with missing data channels removed.

distribution, as can be seen from a slightly lower mean and higher standard deviation for when all the 0's are included compared to when they are removed, when the mean and standard deviation are closer to the expected values of 24 and 64. The number of 0's introduced due to clipping of low values is relatively low and corresponds to only 5% of all 0's in the scaled distribution, meaning that the excessive clipping will not disturb the data.

The scaling factors σ_c and $\bar{x_c}$ are obtained by calculating the running mean and variance at the start of the observation, for the amount of time specified by the user during the startup. In most situations mean and variance can be calculated using a 'naive' approach of double-pass algorithm, where the mean of the data is calculated first and then the variance of the sample of size N is obtained from

$$\sigma^2 = \frac{\sum_{i=0}^{N-1} (x_i - \bar{x})^2}{N-1}, \tag{3.2}$$

with the standard deviation defined as the positive square root of this value. The 'double-pass' nature of the above algorithm means that each data point has to be accessed twice: first to calculate the mean and then the standard deviation itself and does not allow for running variance calculation. By expanding the above equation we can obtain a simple expression for a single-pass algorithm,

$$\sigma^2 = \frac{N\sum_{i=0}^{N-1} x_i^2 - \left(\sum_{i=0}^{N-1} x_i\right)^2}{N(N-1)}. \tag{3.3}$$

In this case we can simply accumulate the sum and the sum of squares of the values in the data as they come in and obtain the final variance when the last required time sample has been received, removing the need to store all of the data. Even though mathematically simple, it cannot be used reliably in numerical calculations as it is prone to catastrophic cancellations where not enough precision is available when subtracting two very similar numbers, especially when these numbers are large. This is the case for this pipeline, as after the averaging the power is at the level of $10^9 - 10^{11}$ and the values within the channel are similar. Using this approach can then lead to unreliable and often mathematically wrong results, such as negative variance - which by definition should always be a positive number.

As we are dealing with large numbers and the running mean and variance approach is preferred to achieve the best performance, this stage of the pipeline uses Welford's algorithm (Welford, 1962) combined with the generalised pairwise algorithm (Chan et al., 1982). Welford's algorithm can be used to calculate the mean and standard deviation in a single pass and is much less prone to the errors described above. With both mean and standard deviation set to 0 at the start of the calculations, adding the n^{th} value from the data set allows us to update new values for the mean, $\bar{x}$ and standard deviation σ using

$$\bar{x}_n = \bar{x}_{n-1} + \frac{x_n - \bar{x}_{n-1}}{n}, \tag{3.4a}$$

$$S_n = S_{n-1} + (x_n - \bar{x}_{n-1}) \times (x_n - \bar{x}_n), \tag{3.4b}$$

$$\sigma_n = \sqrt{\frac{x_n}{n-1}}, \tag{3.4c}$$

for n in the range $1 \leq n \leq N-1$, where N is the total number of samples.

The parallel generalisation of this algorithm allows for the problem of computing mean and variance of a large number of samples N to be split into smaller problems,

using subsets of the original data stream. First, simple sum T and the sum of the square of differences S for the subset of data between the a$^{\text{th}}$ and b$^{\text{th}}$ samples are defined as

$$T_{a,b} = \sum_{i=a}^{b} x_i, \tag{3.5a}$$

$$S_{a,b} = \sum_{i=a}^{b} \left(x_i - \frac{1}{a-b} T_{a,b} \right)^2. \tag{3.5b}$$

With N then divided into subsets of sizes m and n, where $n+m=N$, the total mean and variance can be obtained by combining the partial values for each of the subsets from

$$T = T_{1,m} + T_{m+1,m+n}, \tag{3.6a}$$

$$S = S_{1,m} + S_{m+1,m+n} + \frac{nm}{N} \left(\frac{T_{1,m}}{m} - \frac{T_{m+1,m+n}}{n} \right)^2, \tag{3.6b}$$

$$\bar{x} = \frac{T}{m+n}, \tag{3.6c}$$

$$\sigma = \sqrt{\frac{S}{m+n-1}}. \tag{3.6d}$$

The parallel implementation of the running variance algorithm can have two uses. Firstly and most importantly, by using it we can easily combine the means and standard deviations calculated for each data chunk into values covering the N time samples requested by the user. In this case we simply use the updating formula in Equation 3.6 to combine the mean and standard deviation of the current chunk with the results of previous computations. This means the scaling factors can be calculated for an arbitrarily large number of time samples without the need to store the whole buffer of interest, which can span many seconds and use a substantial amount of memory, allowing us to store only the calculated mean and standard deviation after every iteration. Secondly, if necessary we can use it to speed up the computation of variance and mean for a chunk of data, distributing the work between the threads on the GPU and then combining pairs of results, halving the number of threads doing the work after every iteration.

The scaling factors are used throughout the whole observation, which can have a negative impact on the data in the presence of the gain drift or 'corrupted' scaling factors if strong RFI is present in the first few seconds of the observation chosen for the scaling factors calculations. Previous tests, run with offline scaling factors calculations,

have shown that they do not change dramatically over the course of many hours of observations when calculated every $\sim$4 minutes. If necessary, a future update can be easily implemented that will recalculate the scaling factors in the pipeline every few minutes and make it more resilient to changing RFI environment.

3.2.5 Detection

As described above, after the filterbank stage the data is copied back from the GPU into the RAM buffer. The copied buffer has a large number of time samples, typically 2^{17} or 2^{18}, which corresponds to 7.1 and 14.2 s of data respectively. This data is copied into a DADA shared memory buffer[4], which can have multiple programs attached to it that read and process the data. Our pipeline uses the Heimdall single-pulse detection software[5], which relies on the standard boxcar filtering, as described in Section 2.2.3 is used to search the data for any burst candidates. As already mentioned, the use of the DADA buffer means we can attach any program that supports this kind of data format to it and efficiently swap the processing backends, e.g. to enable pulsar timing, if necessary. We also connect the reading software `dada_dbdisk`, which is used to save the data from the buffer to the disk in case any additional offline processing is required.

The single pulse search pipeline generates a number of candidate files, which contain a number of parameters describing the detected events, the most important ones including the signal-to-noise ratio, the time of the event, the DM for which the S/N was the highest and the beam number in which the event was detected. This information can later be used to correlate the candidate lists from different beams to exclude events detected in more than a certain beam threshold and therefore reduce the number of false-positives caused by RFI. An example plot showing data for 13 beams from the observations of RRAT J1819 $-$ 1458 (please see Chapter 4 for the details) is shown in Figure 3.8.

As the vast majority of candidates are false-positives, additional plots are generated for each of the candidates that help to establish their true nature. There are 4 plots per candidate generated and these are shown in Figure 3.9. First one is the simple dedispersed time series, which is usually not enough to establish whether the signal is a real detection or was caused by RFI. This distinction can be made from the presence of a dispersion curve and therefore the remaining three plots contain the data that

[4] http://psrdada.sourceforge.net/

[5] https://sourceforge.net/projects/heimdall-astro/

has been not dispersed at all or dispersed only partially. The original filterbank is first time-averaged, so that the averaged burst has a width of 4 time samples. The same filterbank is also partially dedispersed into 16 subbands to increase the S/N of any potential signal. The last plot is the combination of the previous two: partially dedispersed and time-averaged data. Figure 3.9 illustrates why additional plots have to be used to examine the candidates with even the highest reported S/N as they can easily turn out to be caused by RFI. With these plots a more informed decision can be made on which candidates should be examined further and which can be discarded.

3.3 Deploying on the Cluster

The pipeline is launched and managed on multiple nodes with the help of Bash scripts. The core processing components are built inside Docker[6] images, which allow us to quickly deploy the software across a large number of compute nodes and efficiently apply any upgrades or bug fixes by simply rebuilding a single image containing the new code and distributing it amongst all the processing nodes. The launch procedure creates a number of configure files which serve as input to the individual pipelines allowing us to quickly respond to any hardware failures and limit the disruption by restarting the processing only for the affected beams.

Initially, each individual pipeline was designed to start the receiving and subsequent processing as soon as the beam and current reference epoch, seconds from reference epoch and frame within the period were obtained by one of the receiver threads to serve as an entry point. This was found to cause problems, as there can be a difference of up to 10 s between the first and last launched beam processing. This complicated the coincidence procedure as all of the times in the candidate files had to be offset to start at the same point in time, causing problems when converting back to the time within a given filterbank. Rounding errors in both steps could introduce an offset of up to 500 ms from where the candidate was originally reported to be, effectively removing possible candidates in the additional diagnostic plots.

To remove the need for introduction of any inter-node communication, such as MPI, a much simpler solution was implemented. During the launch procedure, before any Docker container is started, a Python script is started as a daemon. This supervisor collects the information saved into a file by each of the pipelines about the time of the first received frame. After all of the beams report back, the highest common start time

[6] https://docker.com/

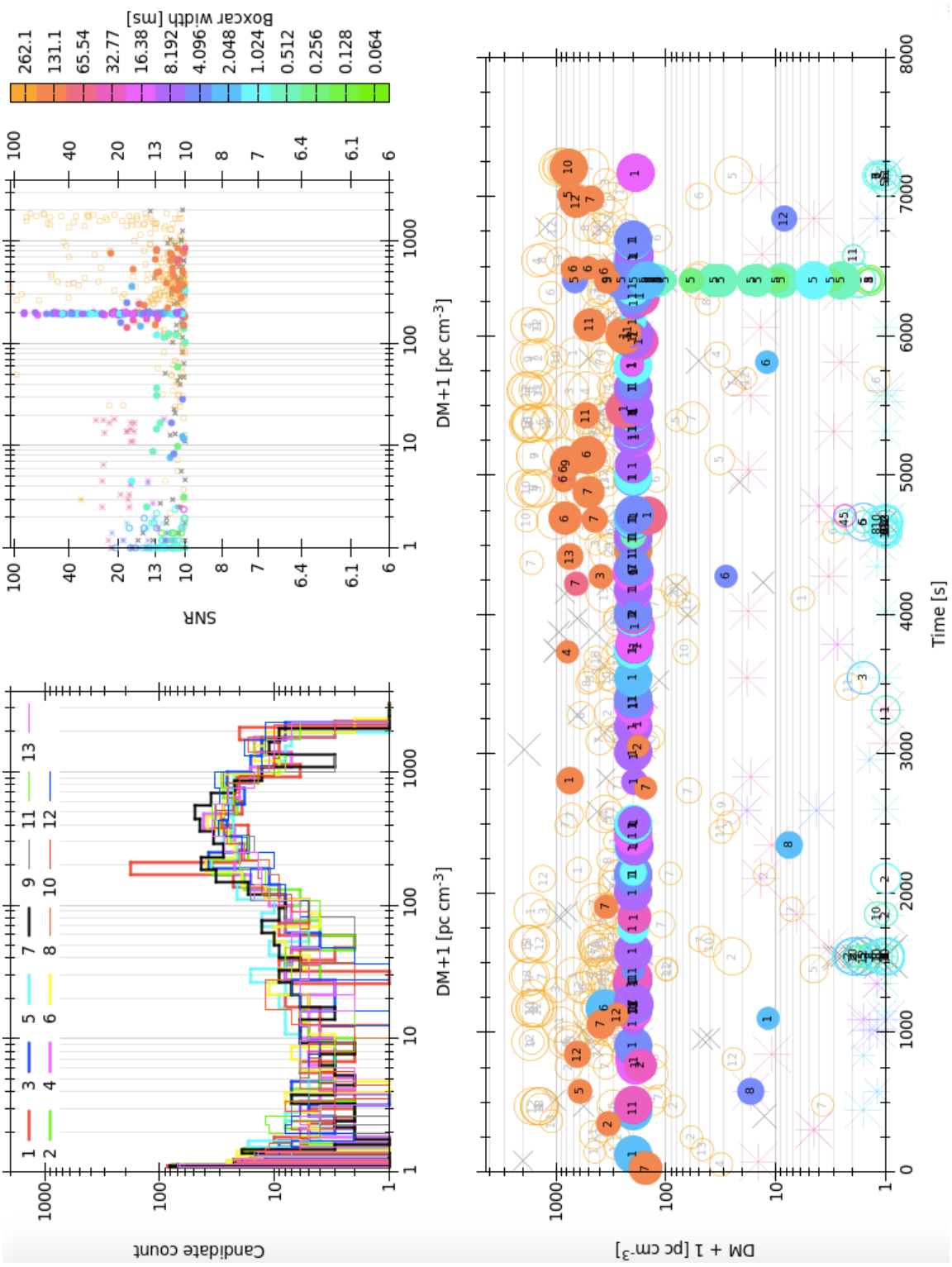
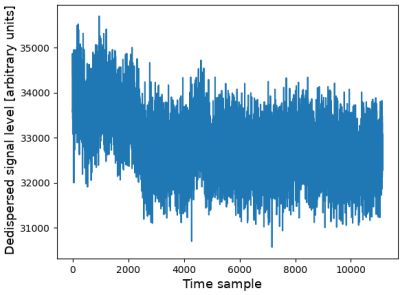
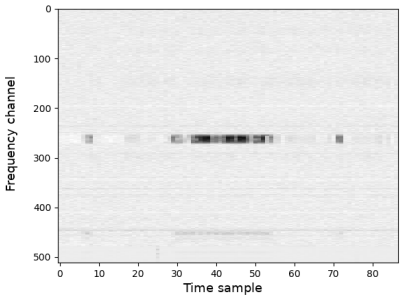


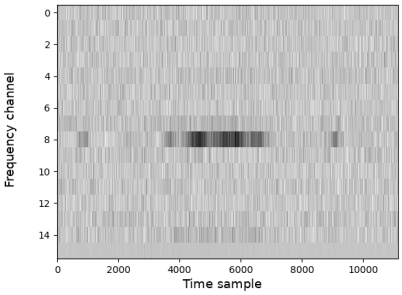
Figure 3.8: Example single-pulse detections from 13 beams pointing towards RRAT J1819 – 1458. The upper left plot shows the number of detections as a function of DM for different beams - beam 1 (the central beam) stands out as the one containing most detections around the DM of 200 pccm^{-3}. The upper right plot shows the SNRs of different detections at different DMs - repeating signal is seen across a range of SNRs at similar DMs. Plot at the bottom contains all the bursts classified as true-positives, mainly coming from the central beam, with a small number of bursts also found in the adjacent beams. Black and grey numbers within filled and unfilled circles represent beam numbers. Filled circles are used to represent candidates, and other symbols are used to represent RFI: low-DM RFI using unfilled circles with black numbers; wide-RFI using unfilled circles with grey numbers; multibeam RFI using crosses; groups with low number of coincident candidates indicating RFI using stars.



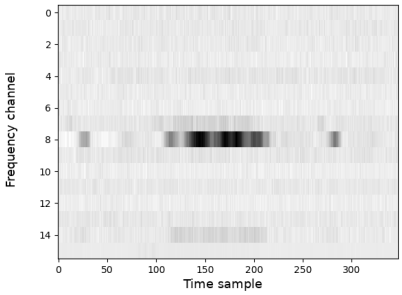
(a) Dedispersed time series.



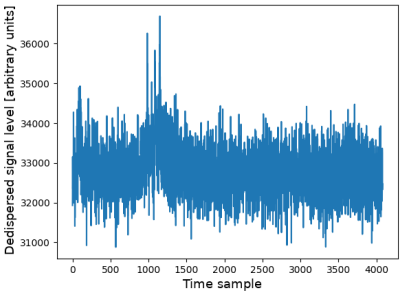
(b) Time-averaged filterbank.



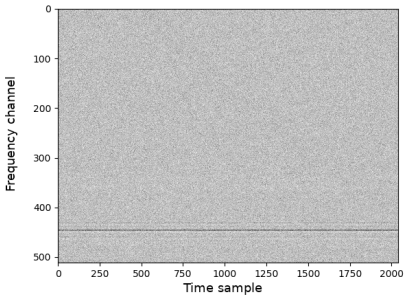
(c) Filterbank dedispersed into sub-bands.



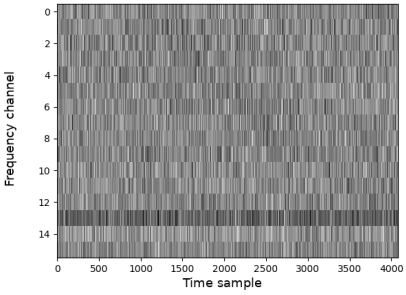
(d) Same as **(c)** but also time-averaged.



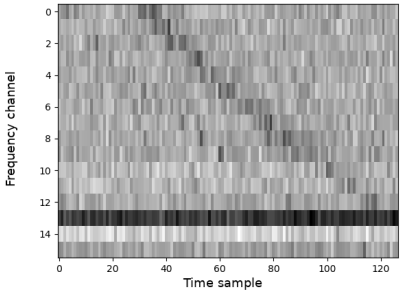
(e) Dedispersed time series.



(f) Time-averaged filterbank.



(g) Filterbank dedispersed into sub-bands.



(h) Same as **(g)** but also time-averaged.

Figure 3.9: Additional PAFINDER candidate plots. The top four panels show a candidate detected at a DM of 545.394 pc cm^{-3} , with a S/N of 27.6. Closer examination showed it was caused by the burst of RFI. The bottom four panels show a confirmed single-pulse detection with the S/N of 7.8, with a dispersion curve clearly visible in the bottom right panel. This shows we cannot rely on just the single-pulse detection logfiles and have to examine the candidates individually. The time samples are on the horizontal axis and the channel numbers on the vertical, with the highest frequency channel on the top.

is chosen and a time buffer of 50,000 frames is added, which corresponds to $\sim 5.4\,\mathrm{s}$ and the information is saved into a file, which is then read by each of the processing nodes. The pipeline then waits until that start time is reached and all the beams can start adding the data to the buffers and processing it at the same frame.

The state of the processing for every beam, including any errors, is saved into log files. Due to a large volume of information, especially during long observations, a monitoring software has been written in Python that combines all the information collected by all the pipelines into single, simple GUI. It allows us to quickly assess the state of the pipeline and react to any errors that may occur during the processing. An example output of the monitoring software is shown in Figure 3.10. The information collected includes the number of warnings generated by each beam processing, which are raised when the GPU cannot process the data in real time and there is more than one element present in the producer-consumer queue, and when there are discontinuities in the time frames of the buffer being sent to disk/further processing, which indicates large packet loss and problems with the networking. Crucially, information on errors is also collected, which are always an indication of more severe problems and are raised on two occasions: when the size of the producer queue exceeds the number of independent buffers in the receiver ring buffer, meaning the pipeline is overwriting the data, and when the time between subsequent large filterbank data chunks is longer or shorter than is should be, which again is usually a sign of severe network problems or when the processing on the GPU stops completely, which can be caused by someone else running a computationally-intensive job. When the real-time processing part of the pipeline is enabled, the user can also get the information on how fast the data is being processed, with green bars indicating real-time processing, yellow less than 1.1 times real-time, orange less than 1.25 time real-time and red for all the values above 1.25. This is crucial information as the amount of time spent on the single-pulse processing is heavily dependent on the number of candidates detected and can vary significantly between beams and sources observed, e.g. during pulsar observations, the central beam is expected to have a larger number of detections and spend more time on processing than the outer beams or when and empty patch of sky is observed during FRB surveys. If the real-time processing requirement is not met for long enough, this will eventually lead to the intermediate DADA buffer not being emptied fast enough and the pipeline being stalled until next filterbank chunk can be placed into this buffer.

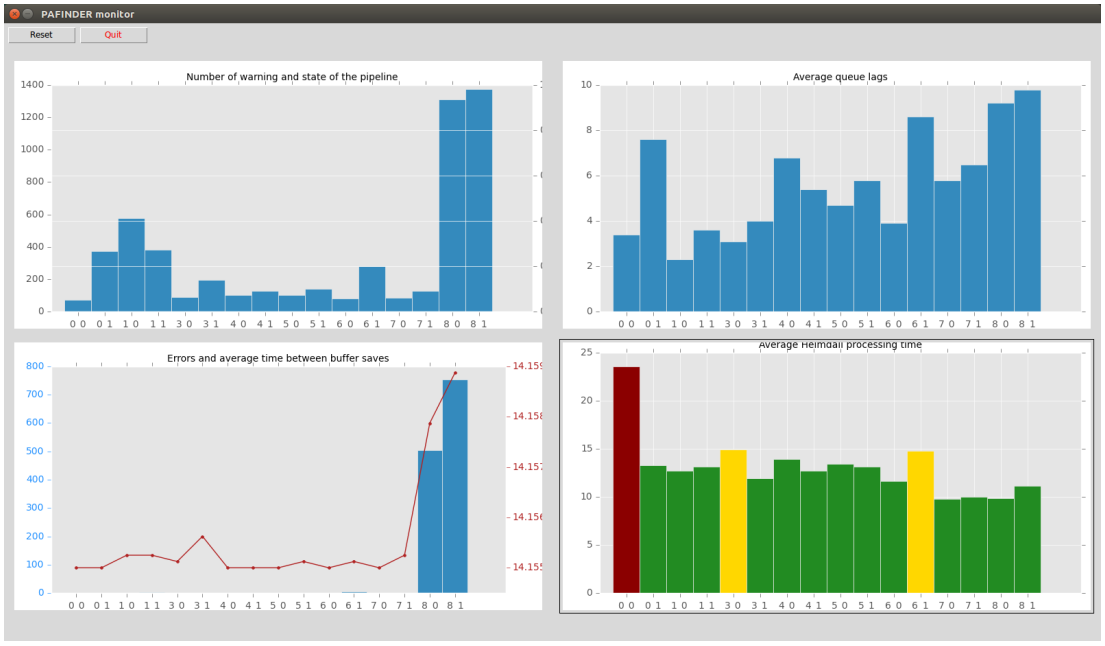


Figure 3.10: Pipeline monitoring software. The horizontal axis shows NUMA nodes numbering, following the convention (*compute node ID*)_(*NUMA node ID*). A large number of errors can be seen on both NUMA nodes of processing node 8 in panels on the left. This was caused by hardware problems on this node and this output helped us to identify the problem. A relatively large number of warnings on all nodes is caused by the GPU filterbank processing slowing down when the single-pulse search algorithm is running and the data queue has more than one element. In a future software iteration, the warning condition will be relaxed and will be issued only after the queue size exceeds half of the maximum allowed value.

3.4 Tests and Benchmarks

All of the benchmarks were run on the same node, with setup as shown in Table 3.1 and using a single GPU only. Various parts of the pipeline were profiled using the same simulated dataset, simulating 128 data frames for each of the 336 channels. The current version of the pipeline processes data in chunks of 256 data frames, but the results of benchmarking apply to a broad range of these values as the execution times were found to scale linearly within a considered range and therefore the achieved speedup will remain the same.

3.4.1 GPU Kernels Optimisation

Listing 3.2: The optimised and final version of the unpacking kernel making use of shared memory to avoid non-coalesced accesses to global memory.

```
1    \\ Read the data into the shared memory array
2    for (int ichunk = 0; ichunk < 7; ++ichunk) {
3    line = ichunk * blockDim.x + threadIdx.x;
4    chan = line % 7;
5    time = line / 7;
6    accblock[chan * NSAMP_PER_PACKET + time] = in[skip + line];
7    }
8
9    __syncthreads();
10
11   \\ Move data from shared to global memory array
12   for (chan = 0; chan < NCHAN_PER_PACKET; ++chan) {
13   polint = accblock[chan * NSAMP_PER_PACKET + threadIdx.x].y;
14   cpol.x = static_cast<float>(static_cast<short>( ((polint & ...
        0xff000000) >> 24) | ((polint & 0xff0000) >> 8) ));
15   cpol.y = static_cast<float>(static_cast<short>( ((polint & ...
        0xff00) >> 8) | ((polint & 0xff) << 8) ));
16   out[outskip + threadIdx.x] = cpol;
17   ...
18   }
```

Unpacking. Initially the unpacking was designed to be run on the CPU. As each packet contains 108 μs of data and signals from 8 FPGAs are sent to a single port, the whole receiving operation, including reading of the data from the network card, unpacking it and putting into a buffer, has to take less than 13.5 μs. At the early stages

of the development, it was the receiving thread that was also responsible for the task of unpacking and saving the data into the first-stage buffer in the already unpacked state. This was the only time the unpacking and subsequent benchmarking was done on per-packet basis and not using the large data buffer described at the start of this section. Initial tests showed that the unpacking part implemented in such a way could not be run in real time, with the processing time close to 25μs - almost twice the required maximum processing time of 13.5μs. This was the case even after the CPU code was optimised with the help of the Intel vectorisation instruction set. At that point a decision was made to move the unpacking to the GPU.

As shown in Figure 3.5 the data is packed in such a way that can cause problems with non-coalesced memory accesses on the GPU. The non-coalesced memory access occurs when the threads within a warp read from memory addresses that fall into two separate cache lines. This most commonly happens when two threads access non-neighbouring memory addresses. The performance hit varies between the GPUs with different architectures and has been lower in recent years, but it is still advisable to coalesce the global memory accesses as often as possible. Using the simple approach either writes or reads to the global memory during the unpacking in a non-coalesced manner, resulting in the suboptimal performance. To avoid this, the first GPU implementation was using texture memory and is used as a baseline for our benchmarking purposes and in Figure 3.11. Texture memory uses the same memory pool as the usual data stored in the GPU, but different array addressing techniques allow for increased data locality, but the memory addresses the threads read from are still required be close together to achieve the best performance. In principle, this could reduce the penalty imposed by the non-coalesced memory accesses, but the practical limits depend on the intended use and can vary greatly.

The second implementation removed the texture memory in favour of the shared memory. Unlike the texture memory, shared memory is separate from global memory and placed directly on the chip. That means it can offer a much higher bandwidth than the global memory. However, it comes at a price of reduced size - only 64KB of shared memory are available on most chips nowadays, compared with multiple GB of global memory. Shared memory therefore has to be used sparingly and carefully. Its main benefit in our case is that it does not have the same memory coalescing constraints as the global memory. The common scheme, used in this kernel optimisation, is to load the values from the global memory in the coalescing manner into a shared memory array, and then save them into the global memory in the coalescing manner again. At any

of these stages, the neighbouring threads can read from non-neighbouring addresses in shared memory and all that matters is that they access neighbouring addresses in the global memory. Even though this generally increases the number of operations, the penalty they introduce is greatly outweighed by the performance gains from the properly implemented memory access scheme. The partial implementation of the unpacking GPU function is shown in Listing 3.2. In the first `for` loop, the threads read the 64 bit words from the incoming data buffer in order as they appear in the global memory array. However, when they are saved into the shared memory array they are arranged in a uniform order with channels in the outer dimension and time in the inner dimension. The second `for` loop is then used to extract the relevant information from the shared memory array and to save the data into the global memory. In this case, the threads save to the neighbouring addresses in the global memory. An important feature is the `__syncthreads()` call. It is placed to ensure that all threads within the block reach the second `for` loop at the same time. As described in Section 2.4 all threads within a warp with 32 threads run in the SIMD fashion, but not all warps in the block have to do so and therefore `__syncthreads()` is used as a barrier to ensure all the threads are done saving the data into the shared memory array before any reads from it occur. Updating the processing to use the shared memory shows significant improvements in performance, resulting in the code running $\sim$2.5 times faster than when the texture memory was used.

Additional optimisations concerned the use of variables passed as function parameters versus hardcoded as `#define` statements and changing the way the input array is handled. In the original shared memory code, the array was passed using a pointer to an `unsigned char` which was later cast to `int2` - a 64 bit integer structure with x and y components for every array element. In the final version of the unpacking kernel, the array is passed using a pointer to `int2` straight away. Together with the hardcoded values, this results in a negligible increase in performance (2.58 speedup up from 2.57) that can be attributed to measurement uncertainties. Nevertheless this version of the optimised kernel is used in the final pipeline implementation.

Filterbank. The part of the code responsible for obtaining the absolute power of the signal and time and frequency averaging also sees great improvements with the use of shared memory. The original version was using a stride that was causing a non-coalesced memory access on either the input or output data. It was also inefficiently using the data when trying to truncate the fine number of channels, obtained

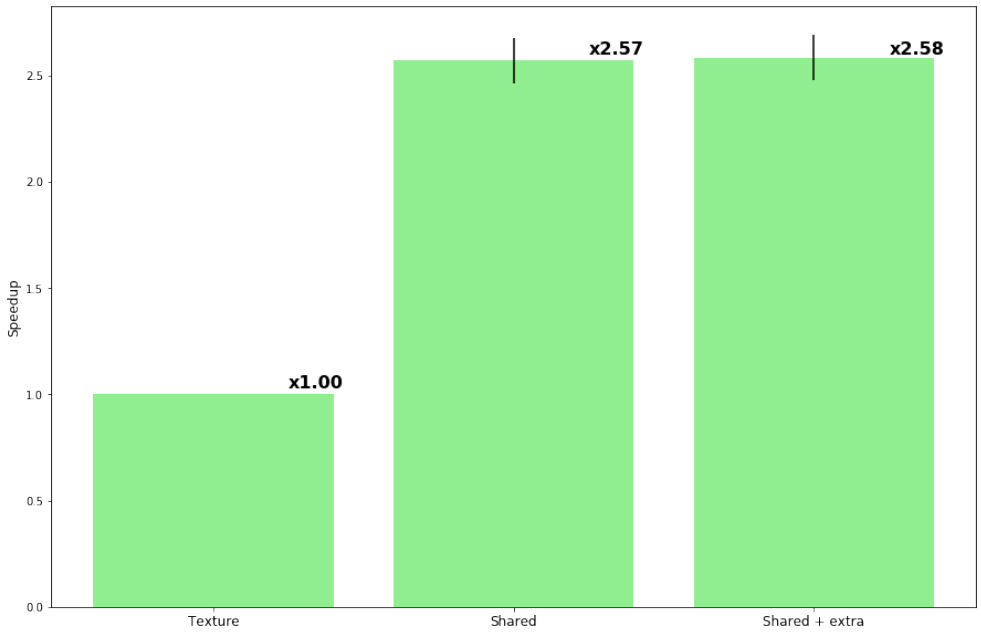


Figure 3.11: Unpack kernel benchmarking. 2.5 times improvement is seen when the the texture memory is replaced with the shared memory, despite the overall increase in the number of memory operations. Additional optimisations, which allowed us to reduce the number of variables passed into the kernel did not visibly improve the performance. Black vertical bars represent errors.

after running the FFT on the coarse '1 MHz' channel, from 32 to 27. All the improvements allowed for almost 25 time better performance, which put the filterbank part well within the real-time pipeline requirements. The unoptimised version was not meeting them and was 3–3.5 times slower than the real-time version should have been. This was hidden by the use of multiple threads bound to multiple CPU cores and multiple GPU streams running concurrently as shown in Figure 2.7. This solution introduced a number of new challenges in the form of processing synchronisation between the different streams and updating the scaling factors which now had to be protected by a mutex. The inefficient use of the available power also meant we were causing the GPU to do more work than necessary and waste the processing cycles. The optimised version helped us to make better use of the available resources and allowed us to redirect the freed processing cores and memory to other parts of the pipeline, such as the single-pulse detection.

Another important question was whether the introduction of scaling and reversing the frequency ordering would introduce any significant penalties (the scaling factors are stored in the global memory). As can be seen in Figure 3.12, only a marginal performance hit is observed after the introduction of scaling. Adding the frequency reversal also does not change the processing time significantly. Using a smaller number of channels introduces no significant speedup, as for the most part all 567 channels are processed - it is only the final averaging, which is the least intensive procedure from a computational and memory use point of view, that is performed on a smaller portion of the data. Overall, adding additional steps to the optimised filterbank kernel does not alter the performance in any significant way and they can therefore be used without loss of the real-time processing capabilities.

Scaling. The main drawback of Welford's running variance algorithm is the apparent number of division operations for calculating the mean. Equation 3.4a uses the current number of samples to obtain the estimate of the new mean and subsequently standard deviation. In a simple interpretation, this means dividing through by a new value for every loop iteration, which in the case of the code shown in Listing 3.3 will result in `2 * NACCUMULATE` (256 or 512, depending on the pipeline setup) divisions. Floating point division is much slower than multiplication and should generally be avoided and replaced with multiplication whenever possible. In the case of this kernel, we use a simple approach of pre-calculating the multiplication factors using a short GPU function that is run during the pipeline setup and runs concurrently with the CPU, where normally the GPU would be inactive. We can then replace the

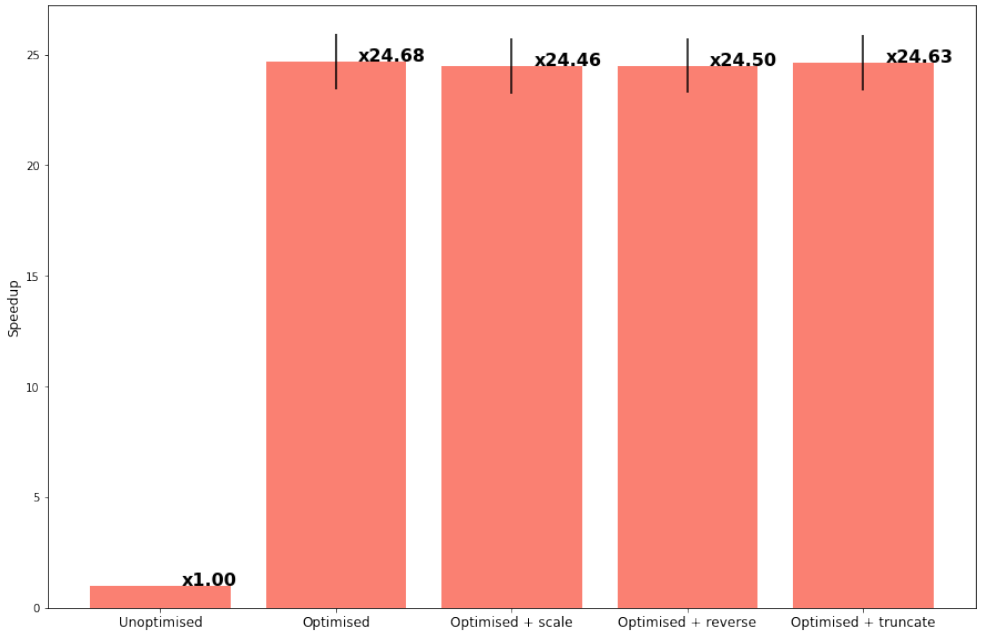


Figure 3.12: Power kernel benchmarking. This kernels sees the greatest improvement after the optimisation with almost 25$\times$ increase in performance. Additional operations, such as scaling and reversing and truncating the frequency channels did not results is any detectable performance changes and are well within the error bars, meaning they were not as computationally intensive as the parts optimised at the beginning.

division with multiplication, as implemented in Listing 3.4. This has a visible effect as shown in Figure 3.13 with the multiplication-based version running almost 4 times faster than the one making use of division. Overall, the best performance that we reach with the double-pass algorithm is almost identical to the single-pass case and we reach the real-time performance requirements in both cases. In this case there is no clear benefit of touching the data only once, as the benefits of the lower number of memory accesses are offset by the increased number of operations performed in order to obtain the estimate of mean and variance. The running single-pass algorithm is nevertheless still the best choice, as it allows us to calculate the mean and variance independently of the number of samples that have been requested and does not require the pipeline to store any long pieces of data, which would have been required with the use of the double-pass algorithm and also provides the necessary numerical stability.

Listing 3.3: Running mean with division

```
for (int isamp = 0; isamp < 2 * NACCUMULATE; ++isamp) {
    val = indata[isamp * nchans + threadIdx.x];
    diff = val - chmean;
    chmean += diff / (isamp + 1);
    chestd += diff * (val - chmean);
}
```

Listing 3.4: Running mean with multiplication

```
for (int isamp = 0; isamp < 2 * NACCUMULATE; ++isamp) {
    val = indata[isamp * nchans + threadIdx.x];
    diff = val - chmean;
    chmean += diff * factors[isamp + 1];
    chestd += diff * (val - chmean);
}
```

Real-time detection. Once the pipeline was confirmed to produce the data in real time it was crucial to test the real-time detection capabilities. As mentioned previously, the pipeline uses Heimdall single-pulse detection software connected to the DADA buffer for single-pulse searching. Additional software, `dada_dbdisk`, is also connected to the same buffer and used to store the data to disk for further examination. The time spent on the processing is heavily dependent on and increases with the increasing number of candidates. Two extreme scenarios were therefore tested. First, the data was

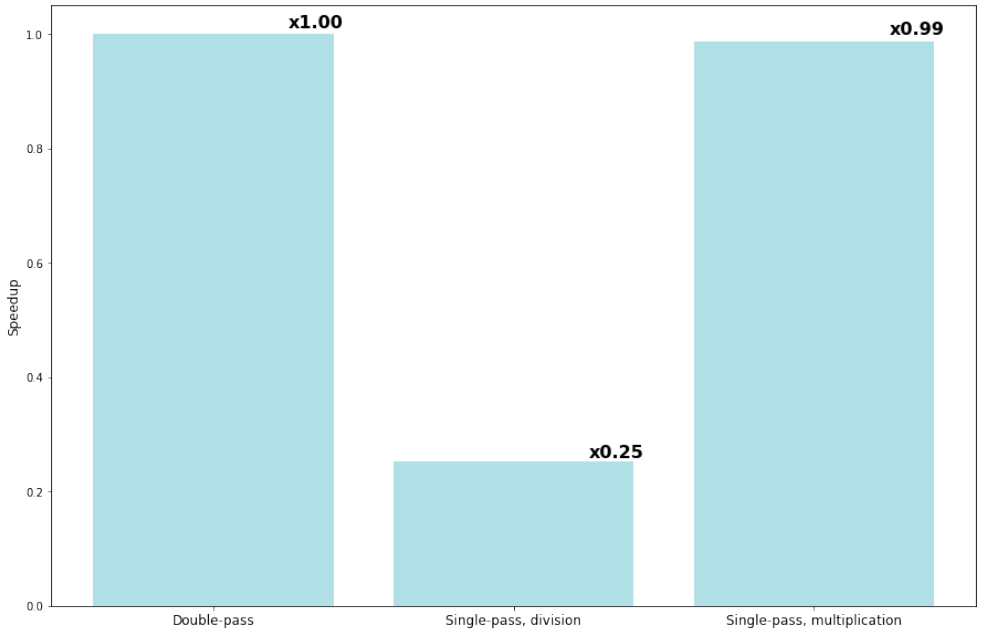


Figure 3.13: Scaling factors kernel benchmarking. A clear performance hit is visible when the single-pass algorithm is used in its most basic form with division. Using multiplication brings the performance up back to the original level.

taken when pointing at a bright pulsar. B0355+54 (J0358+5413) was selected as it is one of the brightest pulsars in the northern hemisphere at the frequency of 1400 MHz (Teoh, 2015), with a flux density of 23 mJy (Lorimer et al., 1995). This pulsar has a period of 150 ms, which means that multiple candidates can be expected in every buffer processed, which can slow the processing down. The example buffer from the DADA buffer saved to the disk is shown in Figure 3.14. This allows us to validate that there are no significant missing portions of the data, the buffers are not corrupted/overwritten and the data is stored in correct order in both time and frequency dimensions.

Figure 3.15 shows the results of tests from two pointings: one with the pulsar in the field of the central beam and the second one pointing at the empty patch of the sky. It can be clearly seen that when the pulsar is observed, the number of detected individual pulses causes the pipeline to slow down significantly and the real-time processing constraints are only met for 3 out of 7 beams. Most importantly, the central beam which, as expected when the correct beam weight are obtained and the observed source is in or near the middle of the central beam, has most of the detections, lags behind more than all the other beams from the inner ring and in the best-case scenario is still around 2 times slower than the real-time requirement. The situation changes when the receiver is pointed at the empty patch of the sky with no known pulsars that

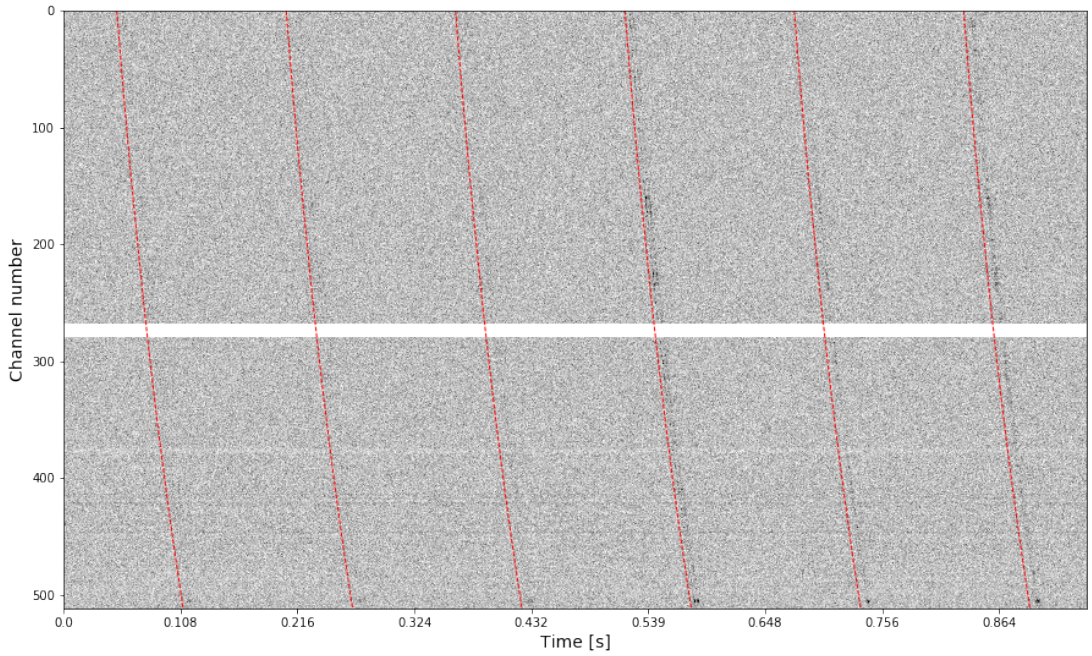


Figure 3.14: B0355+54 data as recorded using the real-time pipeline. Six individual pulses can be seen. Plotted in red are the expected dispersion curves for this pulsar, separated evenly by the period of the pulsar and placed to the left to the darker dispersed pulses actually recorded by the pipeline. The separation and shape of the pulses shows that the recorded data is indeed correct. The white horizontal line close to the middle of the band is caused by the missing data from one of the FPGAs.

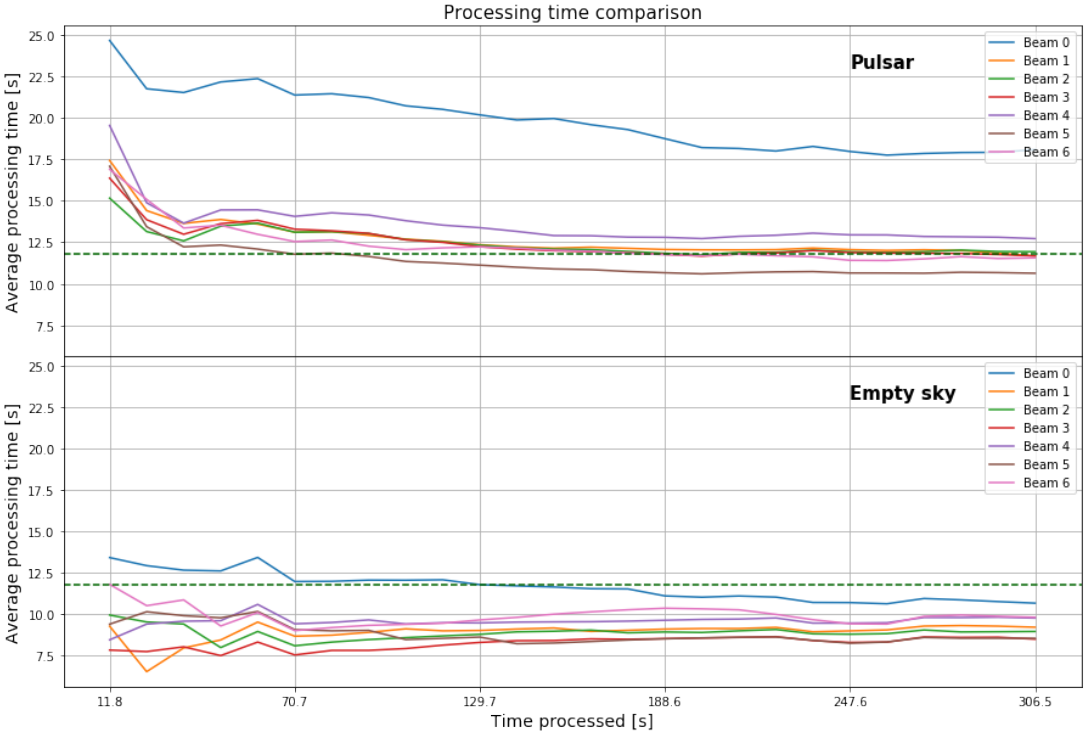


Figure 3.15: Real-time processing comparison for two pointings: one with a pulsar in the field of the central beam and the other with all the beams pointed at the empty patch of sky. The green dashed line represents the processing time limit required for real-time processing - all of the measured times would ideally be below that line for real-time processing. Real-time performance can be reached for beams which do not have too many single-pulse candidates.

would result in a large number of candidates. Although the beams in the inner ring do not have their data processed much faster - the number of candidates was not expected to drop significantly, the situation is most improved for the central beam. Now the data from all of the beams can be processed faster than it is produced, meaning the pipeline can detect single pulses in real time. It is important to note that even in this case, the average processing time meets the requirements by a small margin. This means that a sudden burst of candidate detections, caused for example by bright RFI, can increase it above the allowed value and eventually bring the pipeline to a halt as it will not be able to drain the buffer fast enough.

As the pipeline is run as two separate processes, one producing the filterbank data and the other one running the detection, sharing the same GPU, we can use existing resources that could have the potential to speed the pipeline up. One such solution, NVIDIA Multi-Process Service (MPS) enables a better resource management when different processes are run on the same device, and schedules the device code to be executed concurrently. Better sharing of the available resources should improve the processing time, as currently only around 45% of the available processing power is used when the data production and processing parts run concurrently. The deployment of the MPS was not possible during the tests described above, as it was not supported by the NVIDIA Docker software installed on the cluster and will be tested in the future.

Another option is to carefully tune the dedispersion and search parameters. Simply increasing the step between the trial DM values used during the dedispersion stage can have a large impact on the processing speed as it reduces the number of the dedispersed time series that later have to be examined for the presence of any bursts. This however can come at the cost of decreased accuracy and lowered sensitivity and has to be carefully evaluated. The pipeline can also be configured to stop the single-pulse search for a given time period if the number of candidates exceeds a predetermined value. This can help to avoid problems where there is an increased presence of RFI in the data, but if the threshold is too low, this can lead to the genuine candidates being thrown away as well.

3.4.2 Data Quality Tests

Network performance tests. Receiving the data in full is the most crucial part of the pipeline. We have to make sure that the data loss due to the insufficient processing power is kept as low as possible. As mentioned in the description of the unpacking

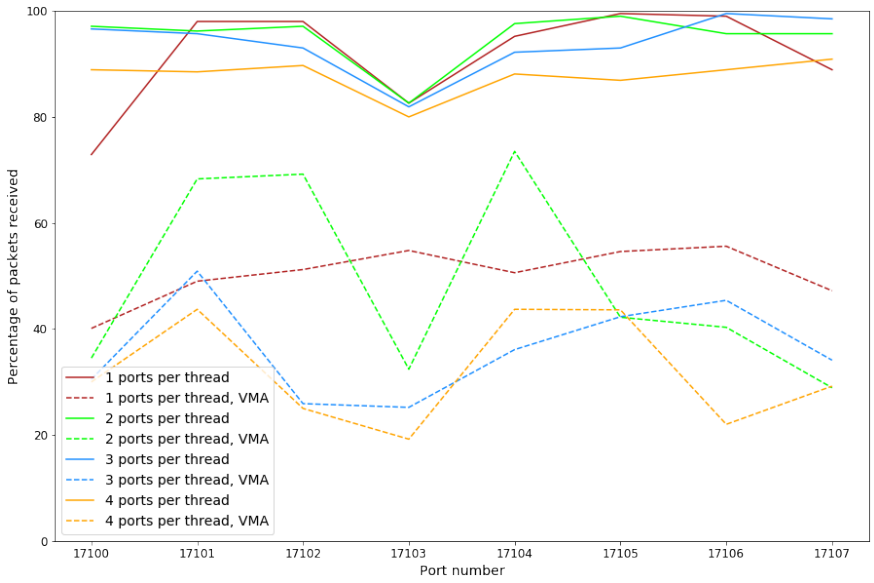


Figure 3.16: Network performance tests results. This test is used to detect any problems with packets loss, as can be seen for port 17103. The use of Mellanox VMA, which is generally meant to increase the performance of network applications, but is difficult to configure properly for multi-threaded applications, running on NUMA-aware systems, introduced even greater packet loss and was abandoned after the testing.

optimisation, this cannot be achieved in real time by the thread responsible for receiving the data. The amount of work done by the receiving threads has to be limited to ideally just placing the data in the right buffer. We have run a series of tests to determine the optimal resource allocation to this first stage of the pipeline. As mentioned above, we use 6 threads bound to 3 physical CPU cores in order to reach the optimum resource utilisation and low packet loss. A smaller number of CPU cores, i.e. 2 cores with 3 threads bound per core, results in significantly increased packet loss. At the same time, using a larger number of cores does not show any evidence of significant improvements in the received data rates, meaning they can be assigned to other parts of the pipeline.

Figure 3.16 shows the results of one of such tests, run during the early design stages, where the data was delivered across 8 ports. This test, run on the older version of the network setup revealed a number of problems, such as an uneven performance depending on the port number (e.g. the dip in the received packet count for port 17003 in Figure 3.16) and the occasional loss of signal from random FPGAs, resulting in the loss of 7 MHz frequency bands. A number of different techniques that often improve the network performance were considered. One of them was the Mellanox Messaging Accelerator (VMA), which implements a kernel bypass to speed up the packet delivery

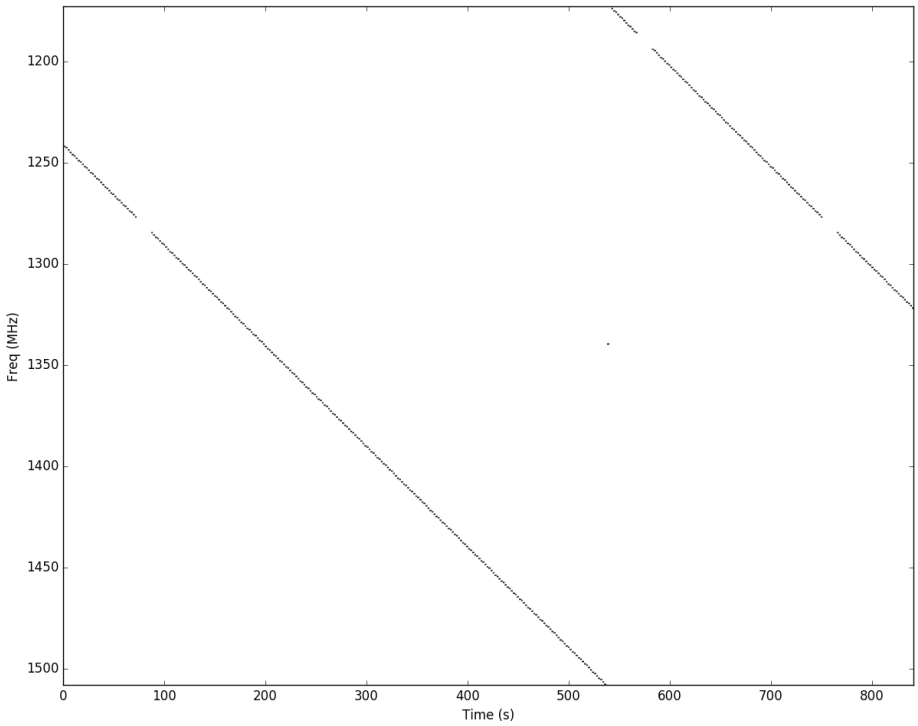


Figure 3.17: Tone sweep test results. With the signal moving through single 1 MHz frequency channels, we can validate the channel ordering and also easily detect any failed FPGAs, manifested as the breaks in the straight black line.

- it removes the relatively slow Linux kernel network packet handling and allows for the direct access to the data on the network card. It was however found that introducing the VMA resulted in a much higher and uneven packet loss, close to 80% in certain situations, as can be seen in Figure 3.16. Further tuning and tests showed that our pipeline would not benefit from this technology. Another approach considered was the use of asynchronous processing in the form of the Boost.Asio library. This was caused by the fact that our software does not see much benefit from concurrent processing when waiting for the receive operation to complete - we need to receive the data to run the necessary calculations for putting the data into the buffers.

Tone sweep tests. It is also crucial to confirm the right order of the frequency channels in the produced filterbank files. This can be difficult if only small errors are present, which may not be detected during regular pulsar observations. During the development of the pipeline we used simulated data injected in the beamformers to validate the correctness of the network setup and the subsequent processing steps.

The beamformers were injecting strong signal into one 1 MHz channel at a time, for a period of 1s, sweeping across the entire frequency band over a period of 336 s. This allowed us to discover problems that otherwise went undetected, such as the need to swap the positions of negative and positive frequencies in the single-channel FFT. This kind of test also helps to easily identify any hardware problems, including broken FPGA connections and networking problems, manifested as the empty bands of data, as can be seen in Figure 3.17.

Scaling tests. During the initial observations, the filterbank files were saved with 32 bit, floating-point numbers used for every sample. If stored this way, the resulting files would use 4 times more space than the scaled, 8 bit version and significantly increase the processing requirements. In order to decrease the disk storage requirements, files were scaled and combined into longer filterbanks outside the pipeline during the initial commissioning with the help of additional scripts. The solution integrated with the pipeline was preferred though. The approach to calculating the mean and standard deviation during the outside processing was different to the one implemented as part of the pipeline. It used a standard, double-pass algorithm. Such a solution is less than optimal when used in the pipeline, due to time and memory constrains, as described in Section 3.2.3. The correctness of the GPU implementation of the parallel algorithm was tested using the scaled RRAT data. All of the central beam filterbank files were processed to remove the scaling and restore the original power. At that stage, the unscaled data did not exactly match the original 32 bit files, due to errors introduced by rounding, and underflow and overflow, but was a close approximation. At first, a quick comparison of mean and standard deviation values obtained using these two solutions offered a rough estimate of the correctness of the new algorithm. Figure 3.18 shows the difference in distributions of power values obtained using the different scaling implementations. It can be seen that they match each other, but do not result in perfectly identical results, which can be expected from the numerical approximations introduced by the Welford's algorithm and the fact we were not using the original 32 bit powers. The differences are however small and are not expected to have a negative effect on the output data. To verify this, all of the filterbank files with new scaling were then fed into the single-pulse search pipeline and the detections were compared to those obtained with the previous implementation. Although small differences can be seen in Figure 3.19, we were able to verify that both approaches allowed us to detect the same pulses from the RRAT, with only small differences in the reported S/N, time or dispersion measure.

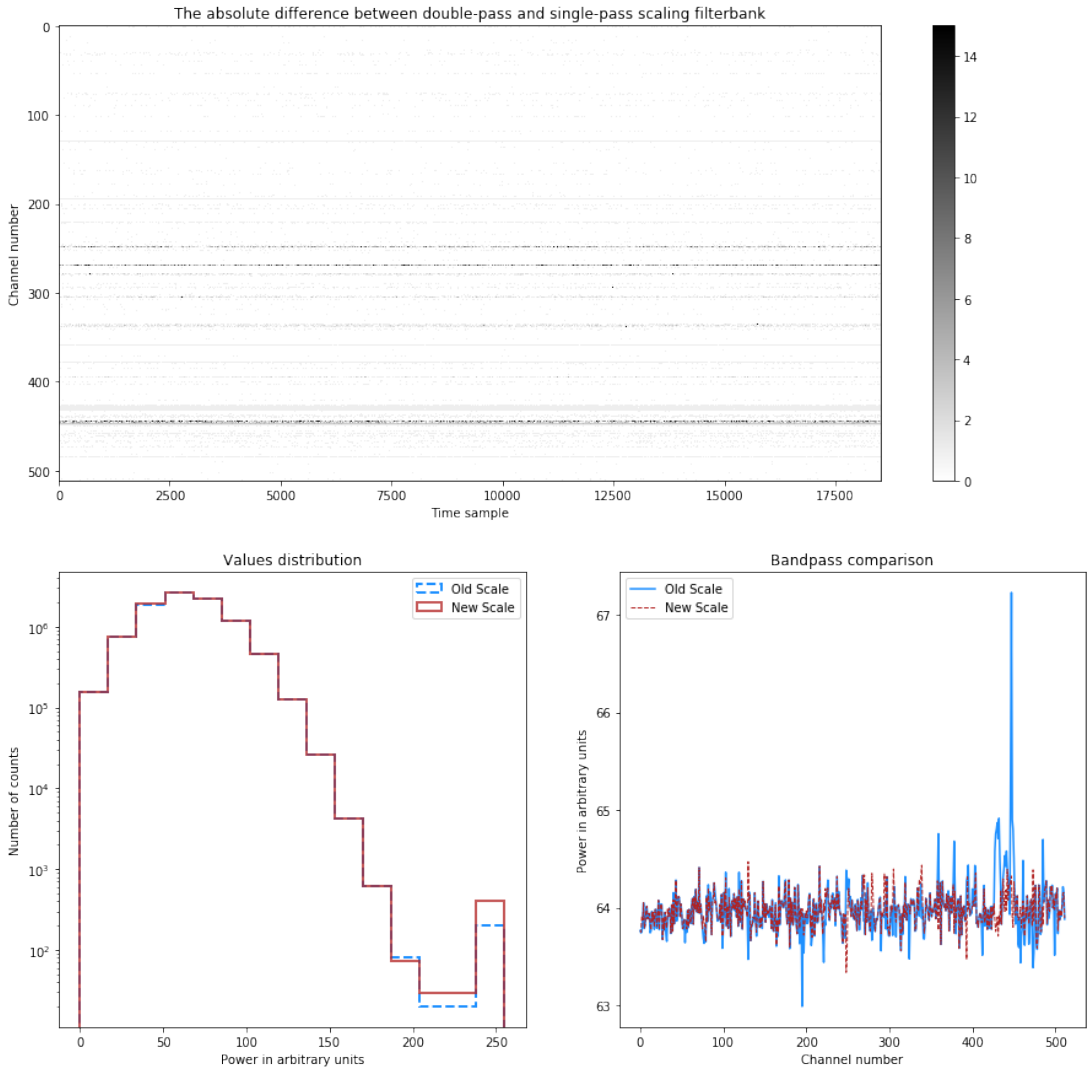


Figure 3.18: Comparison of the scaled data obtained using single-pass and double-pass algorithms. The difference between filterbank files mainly results in small numbers, meaning both files are scaled to similar values. Small differences can be seen in the averaged bandpass and can be attributed to the way the scaling factors are calculated and the number of samples used for these calculations.

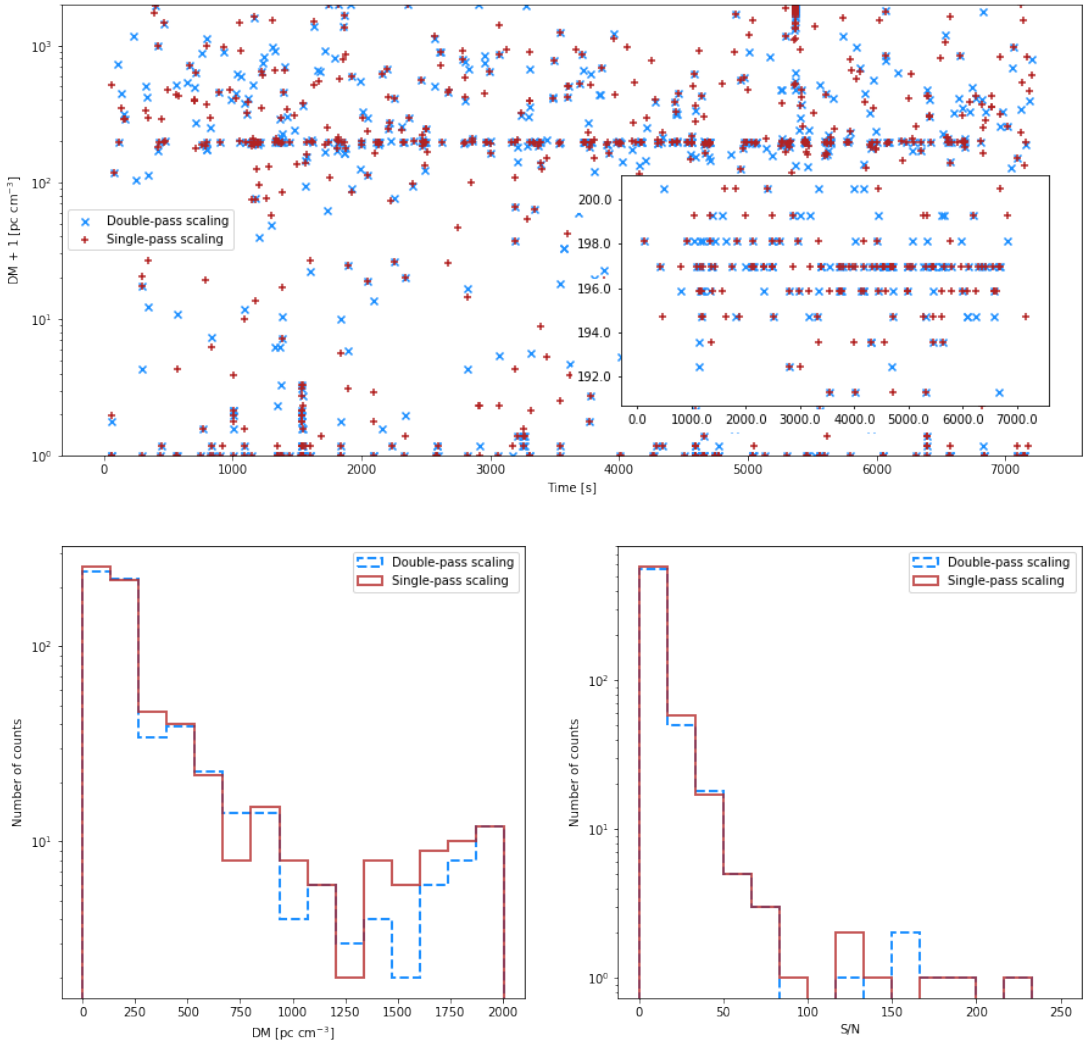


Figure 3.19: Comparison of single-pulse detections using data files obtained with different scaling techniques. The inset in the upper panel zooms in on a DM region around the DM of J1819 – 1458. It can be seen that single pulses are detected in similar positions, with small differences present in some places. These are however not expected to have any significant impact on the performance of the pipeline.

Chapter 4

PAFINDER Observations

After successful confirmation of pipeline performance and data correctness during multiple off-sky tests and initial commissioning runs, we were able to run a series of observations with the receiver mounted on the Effelsberg telescope. These were run between January 19^{th} and 23^{rd} 2018. They allowed us to identify possible problems in the frontend and backend, which can develop over time. This can happen after maintenance and/or updates to both the pipeline code and the PAF management code. We also developed a comprehensive test suite which can now be run before every observing run, which helps us to quickly and efficiently identify problems and fix them before the scientific data has to be recorded. In the following sections we present the results obtained during these observations, including the search for single pulses from known sources, such as RRAT J1819 − 1458 and FRB121102 and looking for new Fast Radio Bursts.

4.1 Metadata

The positions of the beams used during the observations were recorded in a separate stream and used during the offline processing. The right ascension and declination of every beam was saved once every second which enabled a nearly real-time monitoring. In the event of burst detection we could then use this data to obtain the true position of the source on the sky. This approach was preferred during the initial testing and observations for two reasons. First, only the central beam had a constant right ascension and declination as the other beams are not able to rotate together with the sky and therefore cross a range of angles - we therefore had to update their positions on a regular basis in case it was necessary to go back and recalculate them. The second reason

is directly connected to the first one, as the SIGPROC filterbank file format used for data storage does not support changing (right ascension / declination) and (azimuth / elevation) position values, which does not allow us to reliably store these changes in the file header.

4.2 System Temperature Measurements

To obtain an estimate of the system noise temperature $T_{\rm sys}$ and better understand the properties of the receiver, we observed a compact source, quasar 3C286, on January 20$^{\rm th}$ 2018. This allowed us to measure $T_{\rm sys}/\eta$, the system temperature scaled by the aperture efficiency, which quantifies the losses during the signal recording attributed to various factors such as the dish illumination pattern and telescope surface errors. As the efficiency is always less than 1 (it can be anywhere between 0.5 and 0.7 for modern radio telescope (Baars & Swenson, 2007)), the measured values will be greater than the true system temperature. Measuring $T_{\rm sys}/\eta$ gives us the chance to include all of the receiver inefficiencies into a single value, which can later be used to calculate other properties of the system, such as the system equivalent flux density (SEFD). Our source of choice, 3C286, is found at a redshift of 0.85 and at a high galactic latitude of 80.7°.

During the observing, a series of 1-minute long observations were performed, that included all the beams pointed at the source and a reference point at a nearby empty patch of the sky. Using the ratio of the on-source power $P_{\rm on}$, to off-source power $P_{\rm off}$ on the channel-by-channel basis and the definition of the Y-factor

$$Y = \frac{P_{\rm on}}{P_{\rm off}}, \tag{4.1}$$

we were able to measure the system temperature $T_{\rm sys}$ scaled by the aperture efficiency η over the entire operational band (Chippendale et al., 2015)

$$\frac{T_{\rm sys}}{\eta} = \frac{AS_{\rm 3C286}}{2k_b\,(Y-1)}\,{\rm K}. \tag{4.2}$$

In the above equation, A is the geometric area of the antenna ($\pi \times 50^2 \approx 7854\,{\rm m}$ for the Effelsberg radio telescope) and $S_{\rm 3C286}$ is the spectral flux density of our calibration source, 3C286. The positions of the individual beams on the sky have been verified using the metadata, as described in the previous section. The spectral flux density, $S_{\rm 3C286}$ was derived using the following fitted equation and coefficients (Perley

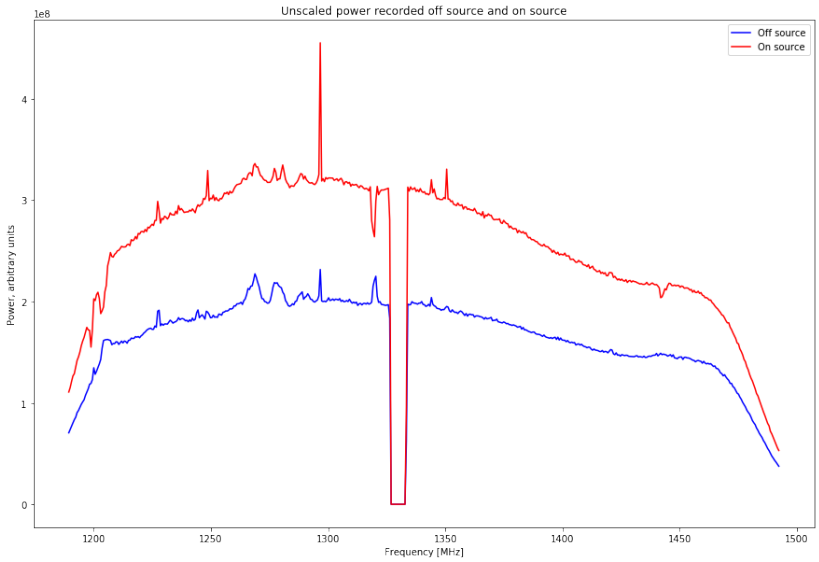


Figure 4.1: Central beam on-source and off-source powers with scaling removed. The ratio of these two powers measured for every beams is used to obtain the Y-factor and ultimately the system temperature.

& Butler, 2017)

$$\log S_{3C286} = a_0 + a_1 \log f_G + a_2 (\log f_G)^2 + a_3 (\log f_G)^3 , \tag{4.3a}$$

$$a_0 = 1.2481, a_1 = -0.4507, a_2 = -0.1798, a_3 = 0.0357, \tag{4.3b}$$

where f_G is the channel frequency expressed in GHz. The modelled spectral flux density of 3C286 was flat over the range of frequencies covered by the PAF receiver, with the average value of 15.1 Jy. To obtain a reliable estimate of the on-source and off-source powers, the scaling had to be removed from the recorded data to recover the original signal. This step is important as any ratio obtained using Equation 4.1 has to include powers that have their true 0 at 0, without any offsets introduced. To obtain a reliable spectrum estimate, the data was accumulated for 30 s, half of the total time spent on the source by each beam. Such a period is long enough to avoid any significant distortions to the measured power from the bursts of short-lived RFI and also short enough to provide a margin for errors inside the log files and the metadata, and the telescope still moving onto the source in the first seconds of the data acquisition. The example on-source and off-source bandpasses for the central beam with the scaling removed can be seen in Figure 4.1, with the signal clearly increasing when the beam is pointing at the source as compared with to reference location.

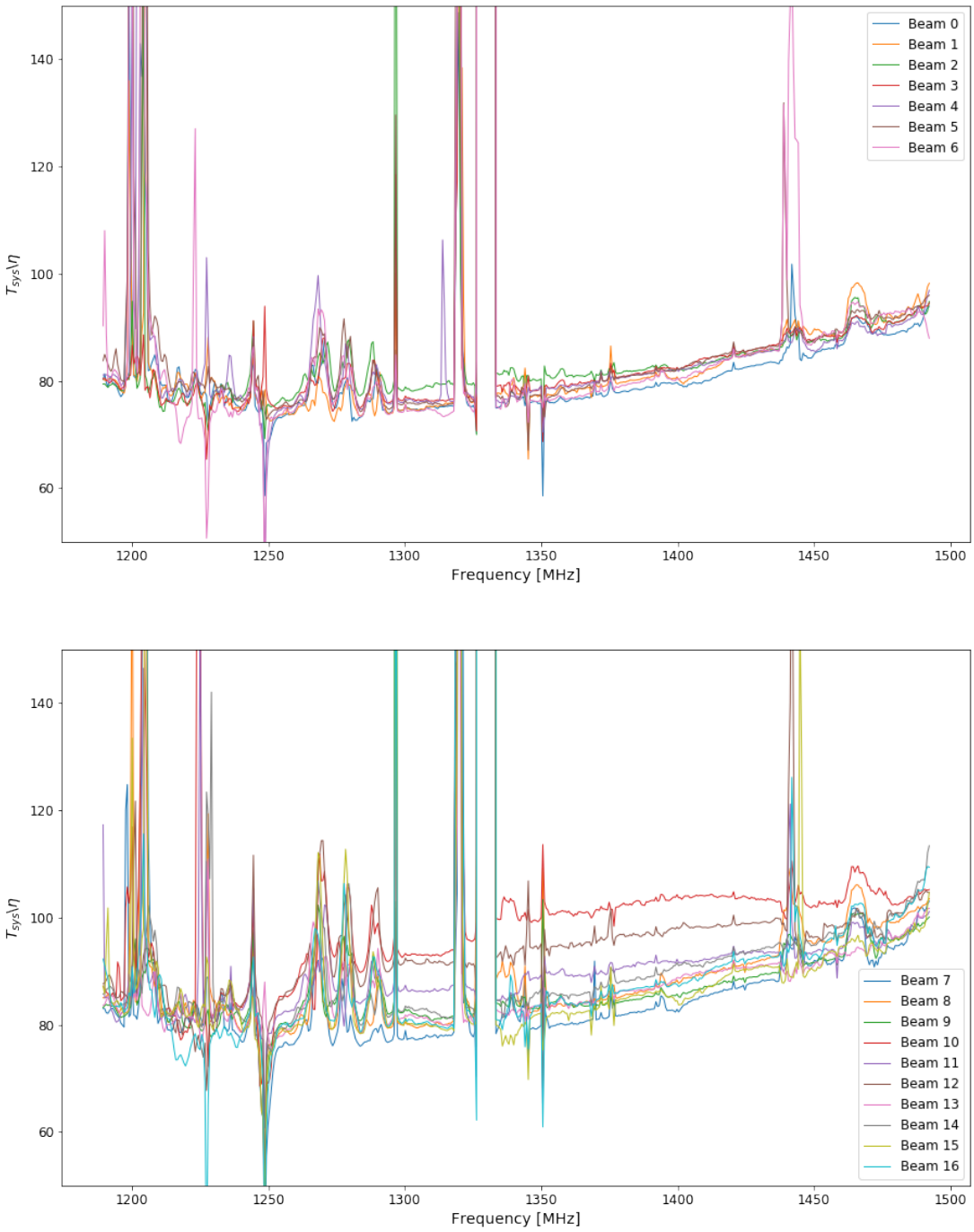


Figure 4.2: System temperature for the inner (top) and outer (bottom) ring of beams obtained over our operational bandwidth with observations of quasar 3C286.

Beam #	0	1	2	3	4	5	6	7	8
	80.93	82.27	83.42	82.32	84.97	85.22	84.41	86.16	89.44
$\bar{T}_{sys}/\eta$ [K]	9	10	11	12	13	14	15	16	
	88.01	100.38	93.28	94.96	87.54	90.35	88.09	87.93	
Beam #	0	1	2	3	4	5	6	7	8
	78.85	79.71	81.42	80.17	80.85	81.21	79.80	82.31	85.98
$\tilde{T}_{sys}/\eta$ [K]	9	10	11	12	13	14	15	16	
	84.78	101.17	89.52	94.91	85.44	87.02	84.88	86.00	

Table 4.1: Mean $\bar{T}/\eta$ and median $\tilde{T}/\eta$ system temperatures for all the 17 beams used during observations.

The system temperature was then calculated using Equation 4.2 for 17 beams processed during our observations, and can be seen in Figure 4.2. 3C286 has a confirmed measured flat spectrum, therefore the spikes in the obtained system temperature estimates cannot be attributed to inadequate modelling of changes in the flux of the source across the frequency band itself. As described in Section 2.3.1, the set of beam weights is generated for each channel separately and then combined ensuring a smooth frequency response. The most probable explanation for the presence of the steep spikes in the power spectrum and the system temperature values is RFI, which if sufficiently strong during the beamforming procedure, can cause the set of beam weights to be created and subsequently used that underestimates the power in some channels. Even in the presence of these spikes and without removing them, the overall baseline and trends in the behaviour in the system temperature can be clearly seen. Table 4.1 shows the derived mean and median system temperature values for all the beams used during our observations. Using these measured values, it is clear that for most of the beams we have expected behaviour and the system temperature is consistent for beams in the inner ring. It is however found to be less consistent for the outer ring of beams (beams 7 and above), with some obvious outliers, especially the beam number 10, which is the only beam with a measured mean and median system temperature above 100 K. This can be an indication of problems with the beamforming procedure, which can be affected by RFI, and might be reviewed in the future to provide a more consistent response for all the beams. Using the average total aperture efficiency of 0.75 (Chippendale et al., 2016), we measure mean system temperatures for all the beams in the range 60–75 K, which is more than double the system temperature achievable with the Parkes Multibeam system (Staveley-Smith et al., 1996).

We then use the approximate system temperature of 80 K for the central beam to

calculate the system equivalent flux density

$$S_{\rm sys} = \frac{T_{\rm sys}}{G} = \frac{2k_B T_{\rm sys}}{\eta A_g} = 28.13\,{\rm Jy}, \tag{4.4}$$

which is similar to the reported SEFD for the central beam of the Parkes Multibeam system at 28.6 Jy (Manchester et al., 2001). This can be attributed to the larger collecting area of the Effelsberg antenna (100 m vs 64 m diameter), which just about offsets the negative effects caused by the higher system temperature.

4.3 RRAT J1819 − 1458 Observations

The sporadic nature of RRATs makes them perfect candidates for the single-pulse detection pipeline tests, as for a number of them, multiple detections are expected every hour, but unlike in the case of pulsars, they do not radiate on the scale of milliseconds and therefore do not swamp the pipeline with a large number of candidates, which a) slows the processing down as described previously in Section 3.4.1 and b) increases the post-processing time if every pulse is examined individually.

To test the ability of the pipeline to detect single pulses, we have therefore chosen to observe a well-known and bright source that emits sporadically - RRAT J1819 − 1458. It is the brightest known RRAT at 1400 MHz, with the peak flux of 3600 mJy. It has a moderate reported burst rate which varies depending on the period when the observations were performed. The first reported rate was $17.62\,{\rm h}^{-1}$ (McLaughlin et al., 2006), but more recent observations put this as high as $68\,{\rm h}^{-1}$ (Keane, 2010). In both cases the burst rates are large enough that we expect to detect a number of single pulses even during a short observation, but small enough that we are not going to saturate our processing pipeline due to the high number of candidates, as the single-pulse detection algorithm used for the processing has the execution time increasing with the number of candidates. The individual pulses of J1819 − 1458 have also been found to undergo changes on short timescales between individual pulses, which gave us a chance to test out ability to detect the candidates from bursts of varying shapes and energies. J1819 − 1458 spends a limited amount of time above the horizon when observed from Effelsberg due to its low declination. The ability to observe some objects using the 100 m telescope can also sometimes be limited by the presence of mountains in the close vicinity of the observatory, which limits the range of elevations to more than $15 - 20^{\circ}$. Combined with various time constraints during the commissioning, it

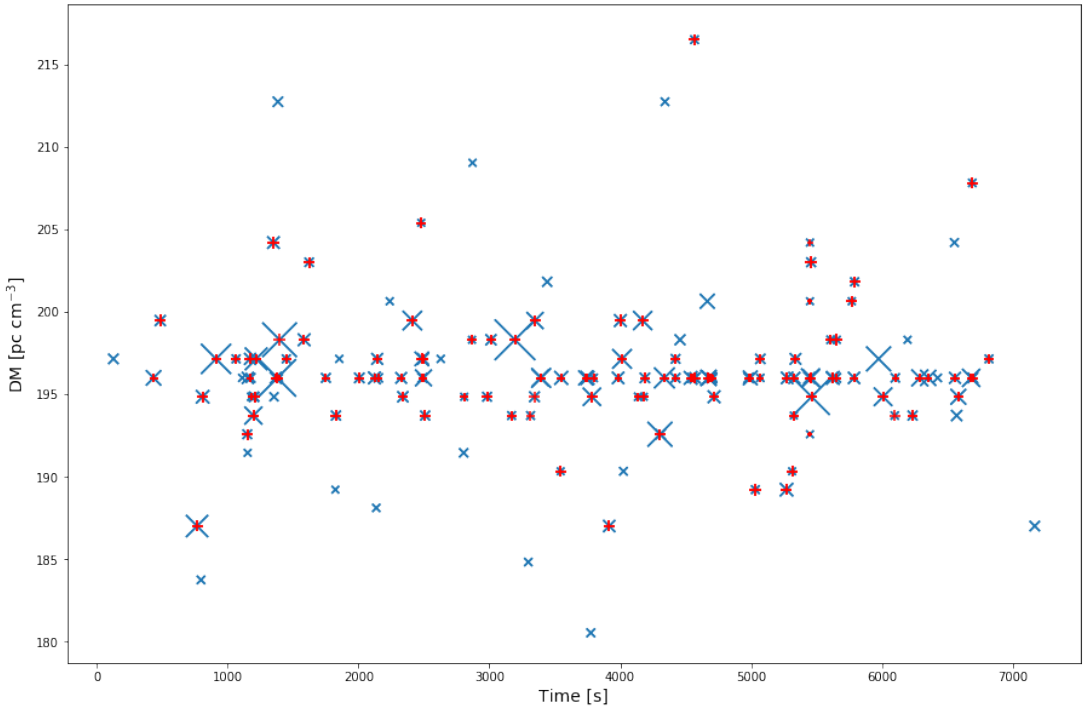


Figure 4.3: Single-pulse detections towards J1819 – 1458. In blue are all the pulses reported by the pipeline, with the size of the crosses corresponding to the S/N. In red are pulses confirmed with additional visual inspection.

meant this object was observed during a single 2 hour-long pointing.

4.3.1 Single-pulse Detections

We have processed all of this data with the single-pulse search pipeline in the offline mode, looking for detections with the S/N greater than 7.5. In total, we were able to detect 108 bursts, resulting in an averaged burst rate of $54\,\mathrm{h}^{-1}$. Figure 4.3 shows all of the single-pulse detections within the DM range of 180 and 220 $\mathrm{pc\,cm}^{-3}$. Each candidate was then inspected separately to reject false-positives by looking at the individual pulses they were found to correspond to, and the additional plots generated for each candidate. Figure 4.4 shows a selection of single-pulse detections as seen in these plots. Using the visual confirmation they provided, we were capable of easily distinguishing between real detections and RFI, as well as identifying and removing multiple detections of the same burst at different DM values and time samples. As can be seen in the bottom 4 panels of Figure 4.4, we were able to detect single pulses which would not normally significantly stand out above the noise level in the simple

dedispersed time series plots and could be improperly rejected, and which become visible when the partially dedispersed and time-averaged dynamic spectrum is examined. As can be seen in Figure 4.3 and the running burst rate in Figure 4.5, we did not detect any pulses for the first few minutes of the observation and only two bursts were detected in the first 10 minutes of the data. This unpredictable behaviour can result in underestimated burst rates if short pointing times are used and happen to occur when the RRAT is going through its quieter phase.

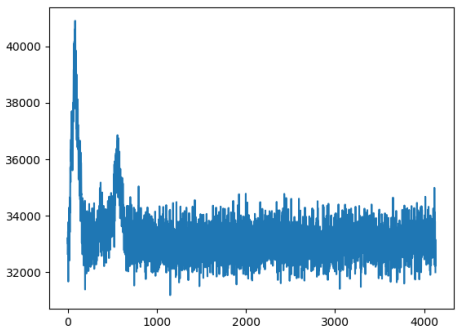
Using the system equivalent flux density of 28.13 Jy as measured previously using the Equation 4.4, we were able to derive peak flux densities for the detected pules from the following equation

$$S = \mathrm{SNR}\frac{S_{\mathrm{sys}}}{\sqrt{W n_p B}}\,\mathrm{Jy} \tag{4.5}$$

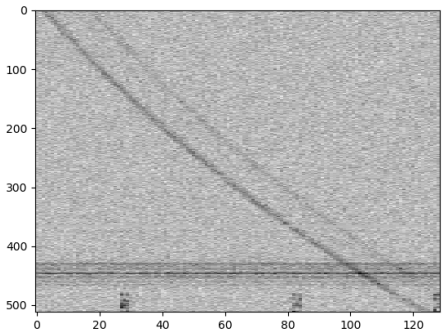
where SNR is the reported signal-to-noise ratio of the detected pulse, W is the width of the pulse as reported by the pipeline, but limited to the integer power-of-2 boxcar pulse widths, n_p is the number of polarisations (2 in our case) and B is the total bandwidth of the system, 303 MHz when 512 channels are used. The bottom left panel of Figure 4.5 shows that the vast majority of pulses are found below 1000 mJy, but some of them can be as bright as 2600 mJy. The overall shape of the peak flux density distribution is consistent with values obtained previously (McLaughlin et al., 2006).

4.3.2 Timing and Profile Evolution

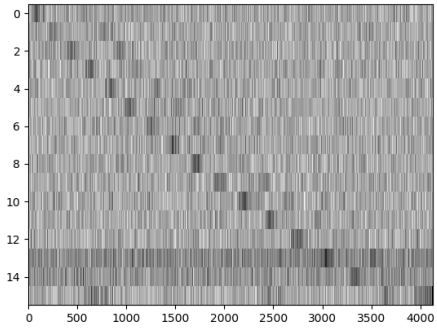
Our burst value of $54\,\mathrm{h}^{-1}$ is higher than the values generally reported during different observational campaigns. On one previous occasion, the increased burst rate was found to be associated with a glitch (Keane, 2010). We have therefore checked for any obvious signs of a recent glitch in our measured arrival times of individual bursts. With more than 100 pulses detected for J1819 − 1458, we were able to obtain a good quality timing solution and residuals. Initial results showed a significant deviation from the predicted timing solution for some of the pulses, of the order of half of the 4.263 s period. That was a strong indication that some of the detected pulses were obscured by strong RFI rather than any systematic errors introduced by the presence of a glitch, which is expected to introduce the same shift in the predicted arrival time for all of the pulses. In total, we have identified 11 such pulses from which the RFI was removed by flagging the offending channels and setting them to 0, and the correct pulse structure was revealed. The timing residuals obtained after the offending RFI has been cleared are shown in Figure 4.6. We do not find any evidence of a glitch happening during our



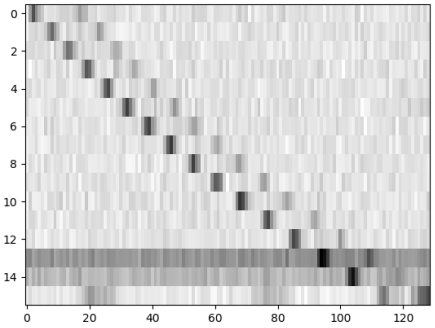
(a) Dedispersed time series.



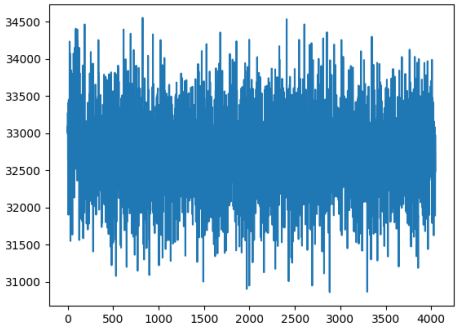
(b) Time–averaged filterbank.



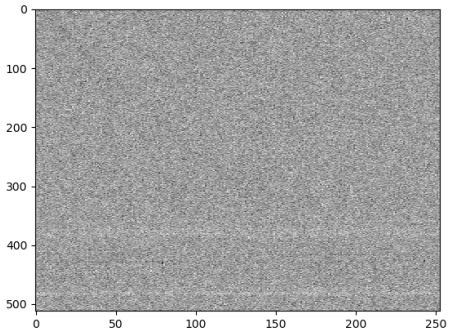
(c) Filterbank dedispersed into subbands.



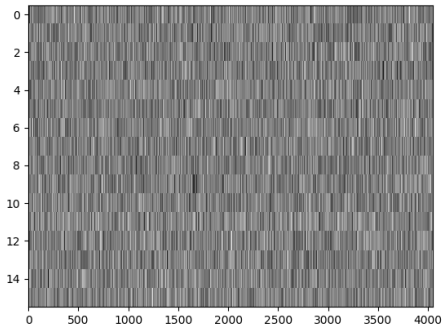
(d) Same as **(c)** but also time–averaged.



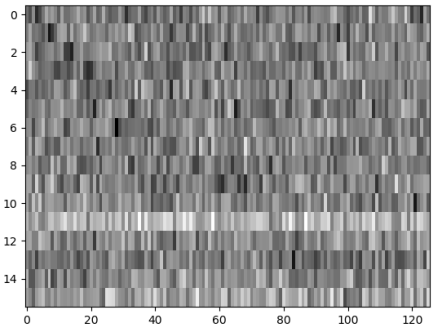
(e) Dedispersed time series.



(f) Time–averaged filterbank.



(g) Filterbank dedispersed into subbands.



(h) Same as **(g)** but also time–averaged.

Figure 4.4: Example of two confirmed single pulse detections with a S/N of 83.12 with two emission components (top 4 panels) and 9.76 (bottom 4 panels).

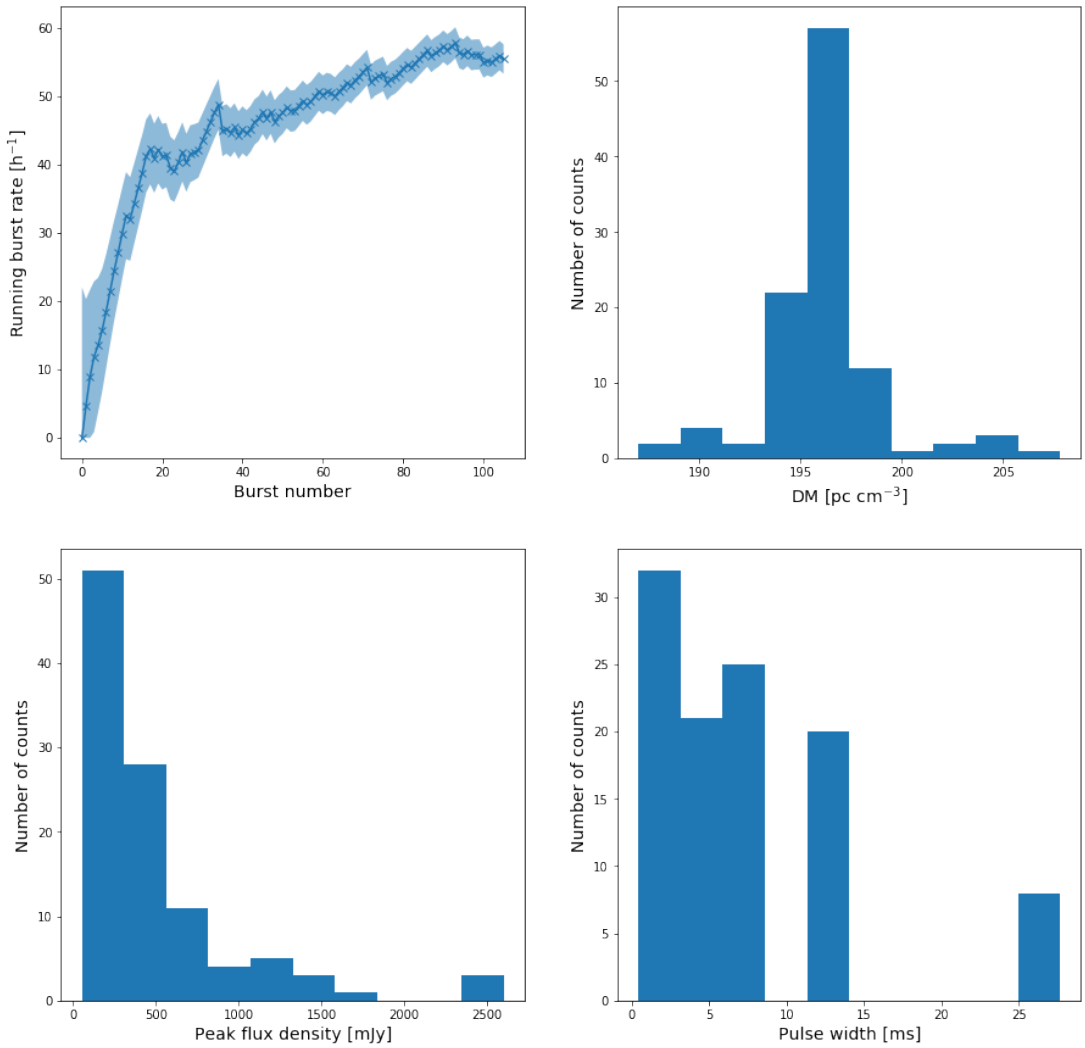


Figure 4.5: Basic properties of J1819 – 1458 measured during observations. The burst rate was significantly lower than the average during the first 10 – 15 minutes of observations.

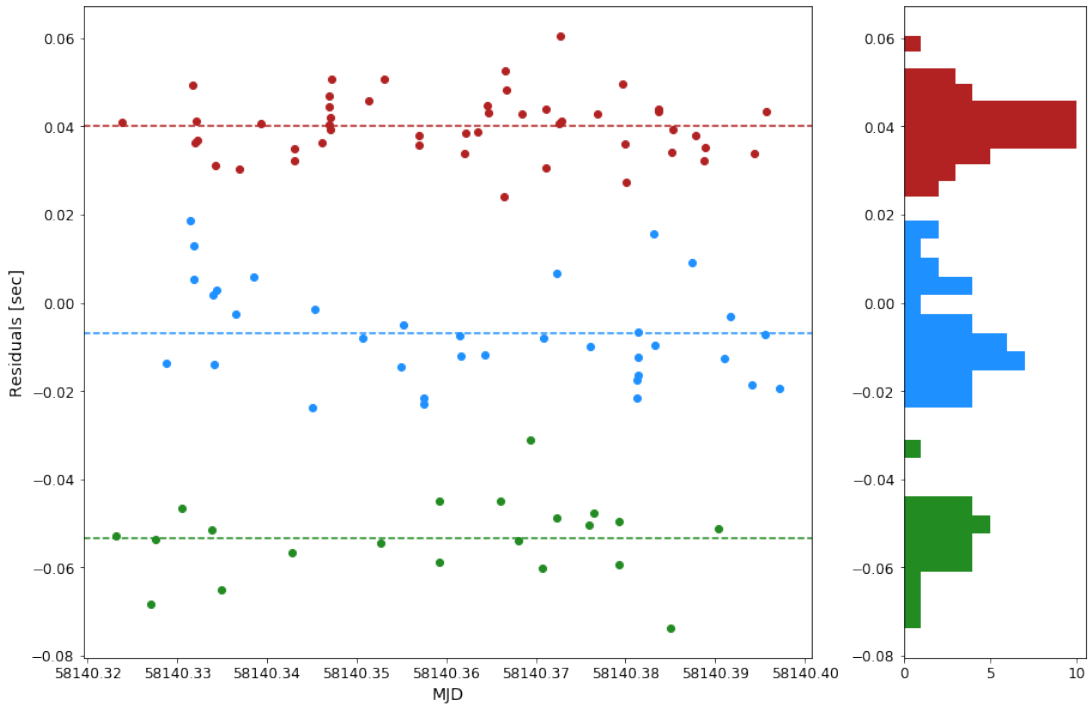


Figure 4.6: Timing residuals of RRAT J1819 – 1458. Three distinct bands which are an evidence for the presence of different pulse components are clearly visible.

observation. As the timing residuals are close to being uniformly distributed around 0, there is also no evidence of any glitch occurring before the observations, as the post-glitch ephemeris is found to fit the times of arrival of individual pulses well. It is important to note that although a single association between glitch and increased burst rate has been found, another event with increased burst rate, similar to the one reported here, was identified without a glitch association (Keane, 2010). It is therefore not unlikely this particular glitch was an one-off event and glitching for this particular RRAT cannot be associated with an increased bursting rate with a high degree of confidence.

J1819 – 1458 shows different emission components, which are first manifested by the timing residuals split into three clearly separate bands, as seen in Figure 4.6. Out of all the recorded pulses, 46.15% are found in the upper, 33.66% in the middle and 20.19% in the lower band. Our observations show a lower number of bursts originating from the middle band as compared to other surveys, which found that J1819 – 1458 spends more than 50% of time emitting in the middle band (Lyne et al., 2009; Bhattacharyya et al., 2018). A closer examination of individual detections revealed that a proportion of pulses had not only emission starting at different rotational phases, but

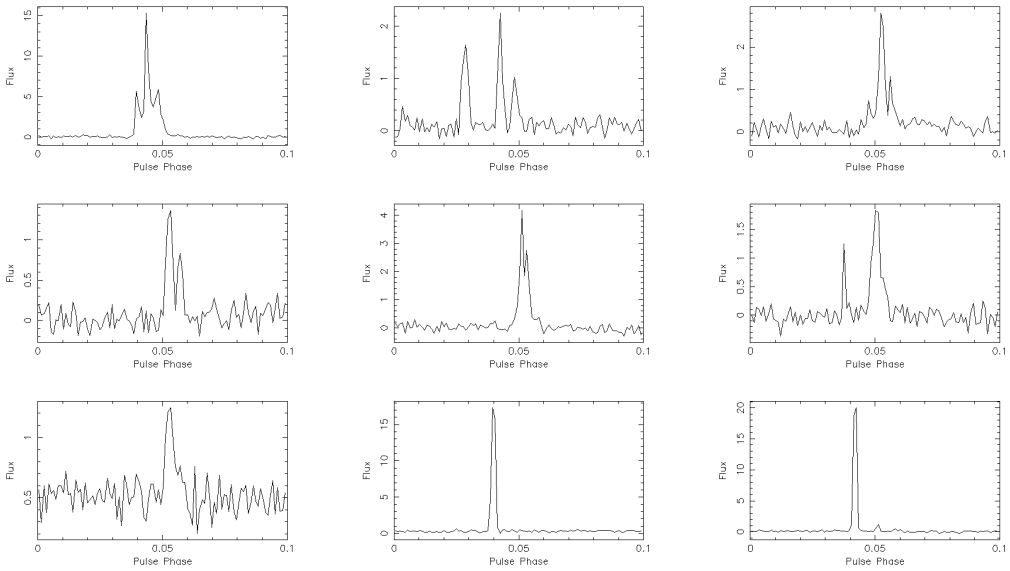


Figure 4.7: Selection of J1819 − 1458 pulses. The triple (top row), double (middle row) and single-component (bottom row) pulses are shown.

they also had multiple components, which were varying with time. Out of 108 detections, 39 were found to have two components and another 5 had three components. A selection of some of the most interesting detections, which show dynamic changes happening on pulse-by-pulse time scales, is show in Figure 4.7. These best represent the complex and changing behaviour of J1819 − 1458 and its multi-component emission. For the triple-component pulses, the central peak tends to drift between the other two components and in some profiles can be found closer to the leading component, while in others it is found closer to the trailing one. This behaviour can help us to interpret a larger spread in of the residual values for the central emission component. What have been classified as single-component pulses also exhibit profile changes, with varying pulse widths and peak flux densities, as well as emitting in the different bands, as shown in Figure 4.6.

While most of the pulses with multiple peaks are found in isolation, we have found a single grouping of multi-component detections in our data, shown in Figure 4.8. Two triple-component pulses are separated by a double-peak burst and the group starts with and ends with a single double-peaked pulse. This is the only time when we find two pulses with three separate components so close to each other and also surrounded by a group of double-component pulses. This group is also isolated from all other pulses. The preceding candidate is found 16 periods ($\sim$68.21 s) before the start of the group.

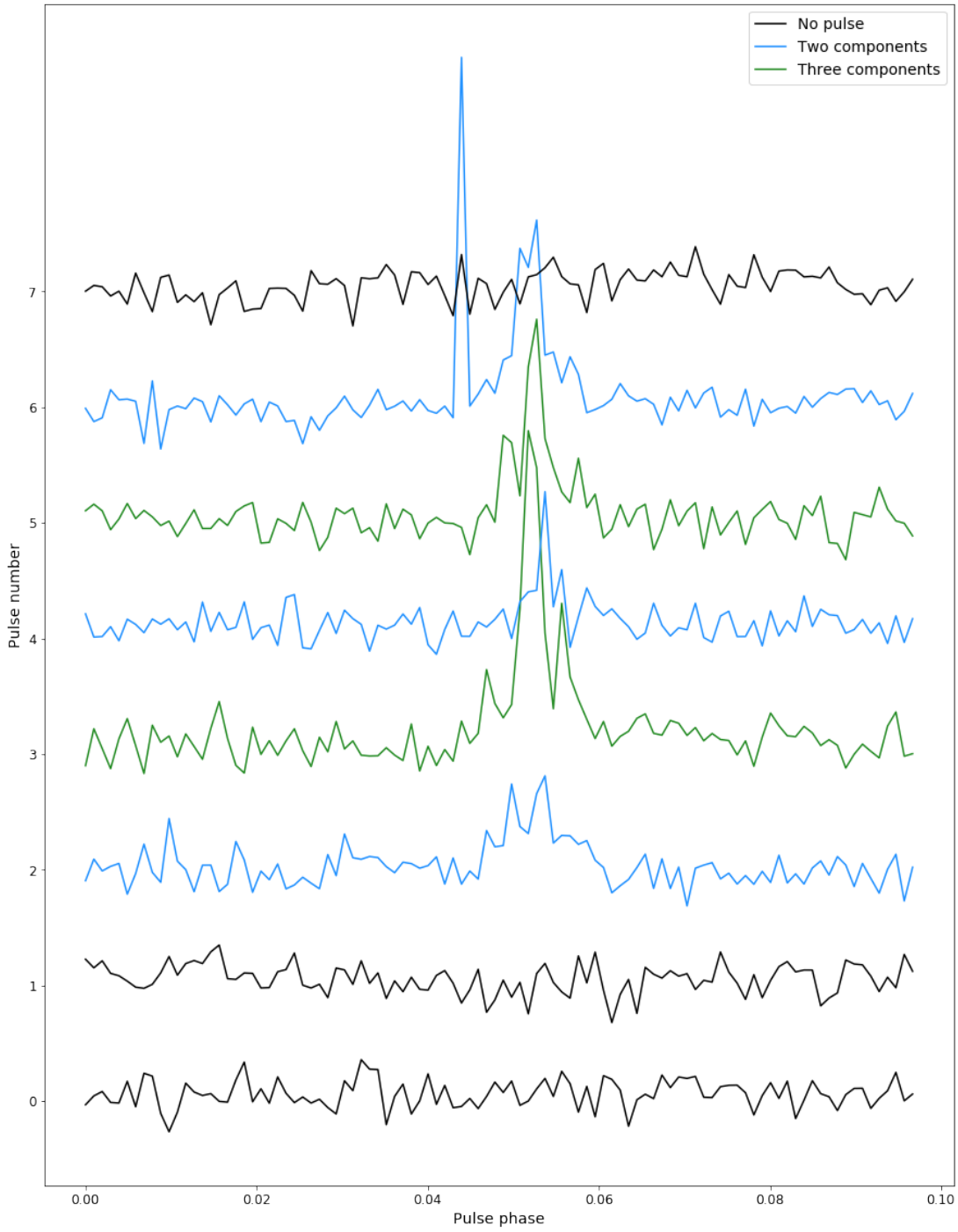


Figure 4.8: A group of multi-component detections. The additional peaks in the triple-component detections are found much closer to the central component than usual. The last double-peaked pulse (number 6 on the plot) is also found with an unusually distant and bright leading component.

While the next single-pulse detection is found only 3 periods away, the one following that is found 73 periods away, which corresponds to more than 310 s from the end of the group. This is the second longest separation between any individual pulses seen during our observations and the longest separation for any group with more than one member. This is not an isolated example, as we were able to find multiple groups of bursts occurring in close succession, the longest one consisting of 5 single-component pulses, with separations between the group and the surrounding bursts measured in tens of rotational periods.

Figure 4.9 shows the separation to the preceding and succeeding bursts for various groups identified in our data. The multi-component group described above is counted as containing 6 single pulses, adding the very last one separated by 3 periods from the main group and even though its total separation is not the greatest (the preceding burst is a modest 68 seconds away), it still stands out as one of the more separated successive detections. It can be seen that for most of the bursts the separation is of the order of tens of rotational periods. Although most emission groups have only a single member, a number of groups can be found, mainly with 3 and more members, which are mostly separated by more than 100 s from all the other bursts, corresponding to more than 20 rotational periods. Although no clear correlation between the group size and its separations from other bursts has been found, Figure 4.9 shows that the majority of pulses are separated from their neighbours by multiple rotation periods. Groups with 3 and more components have also been found to have more consistent large separation. As we were able to identify only a small number of groups with 3 and more components, we recommend similar analysis be performed with a larger dataset, spanning tens of hours of observations. Only when a larger number of emission groups is found, a clear correlation (or lack thereof) between the size of the group and the time between the neighbouring emission can be established. If proven to be correct, the generally large separation could be an indication of J1819 − 1458 going through a period of more violent energy release, followed by a period of relative calmness, when the RRAT 'winds up' and accumulates the energy through still unknown process.

All of the detected pulses were carefully analysed and any RFI contamination has been removed by masking either offending frequency channels or entire time chunks in case of broadband RFI. These bursts and variations in the overall shape and the number of emission components can therefore be classified as being intrinsic to the pulsar rather than to the local telescope environment. They are also consistent with recent observations using a range of different radio observatories (Hu et al., 2011;

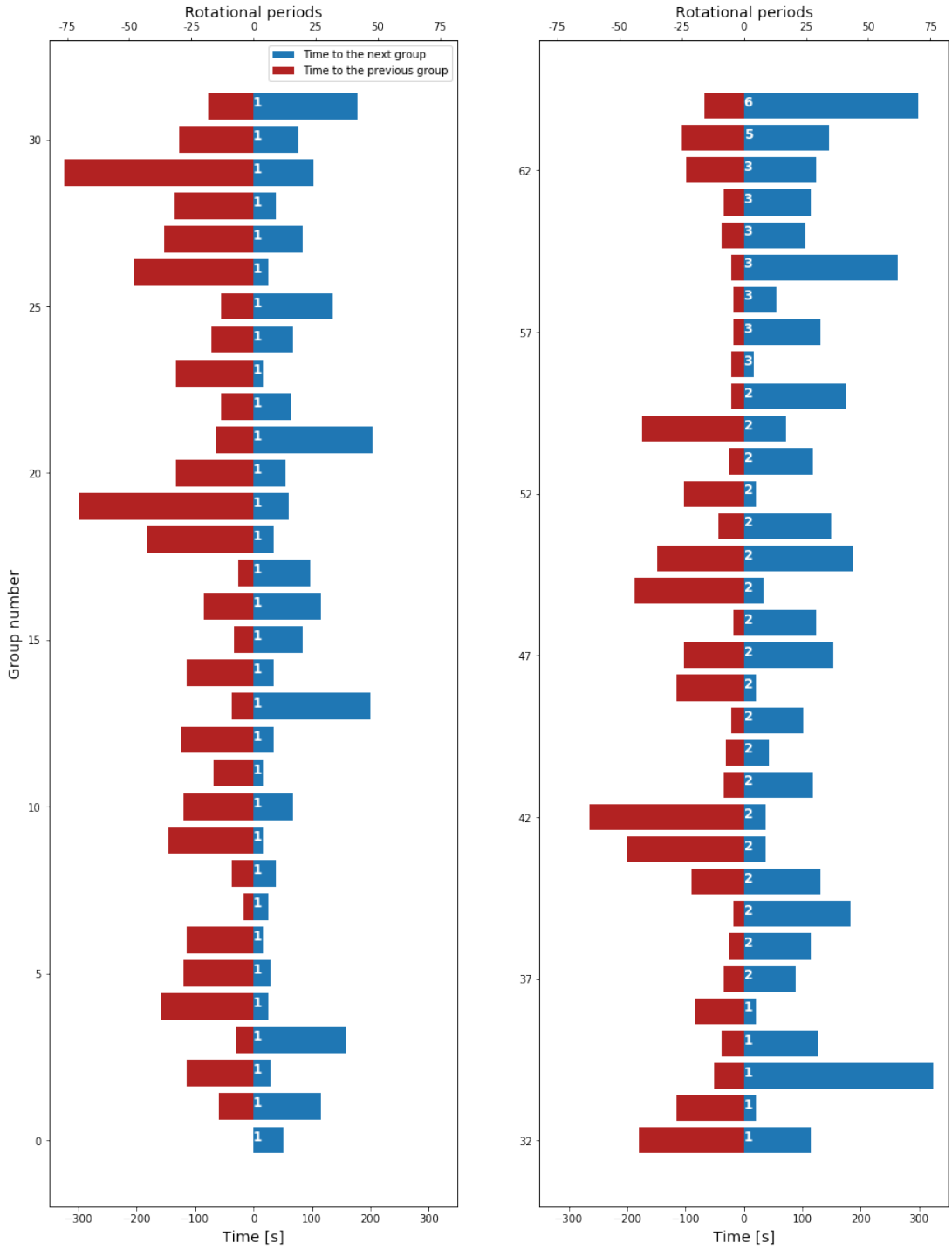


Figure 4.9: Single-pulse groups separation for J1819 – 1458. The size of each group is indicated as a white number in the middle of each bar. The groups are ordered by the group size and then by the time of arrival.

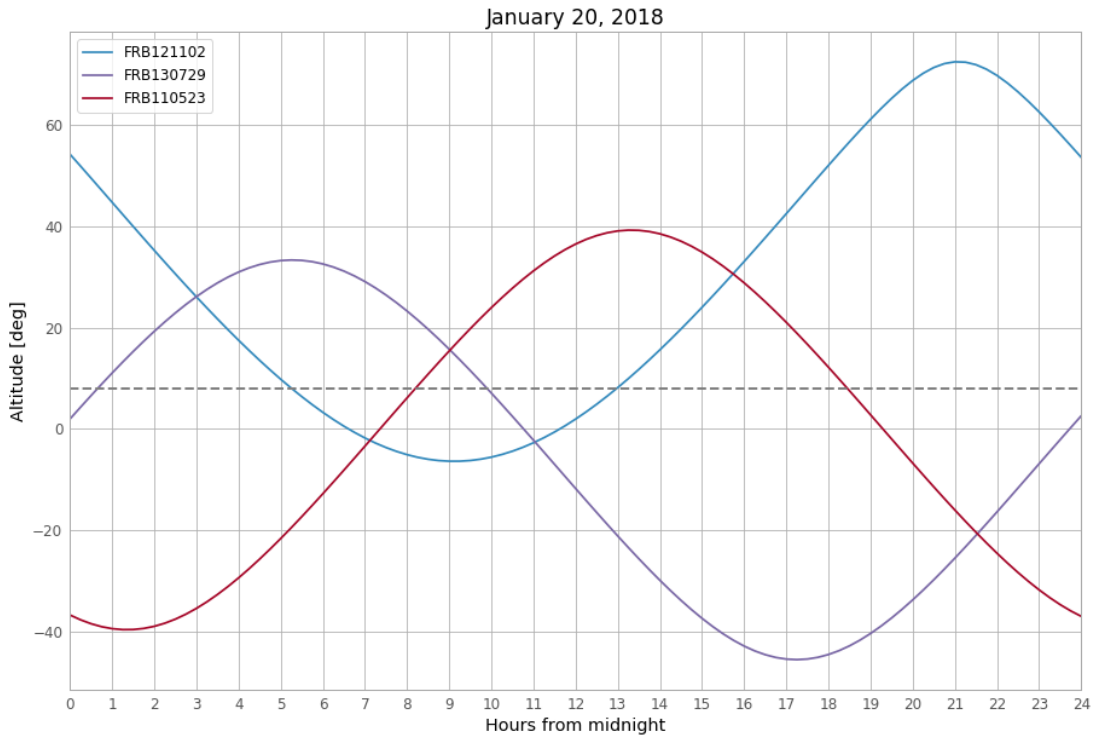


Figure 4.10: The altitude of sources used during the January 2018 commissioning run for the targeted FRB searches. At least one source was visible on the sky at any given time, allowing us to, at least in theory, run continuous observations. The dashed black line represent the elevation limit of the telescope, but this value varies in practice depending on the azimuth, due to the presence of nearby hills and strong RFI sources that had to be avoided during the observations.

Bhattacharyya et al., 2018). It is therefore most likely that these changes in the bursting behaviour are truly the manifestation of the complex and varying emission environment and mechanism of J1819 − 1458.

4.4 Fast Radio Bursts Observations

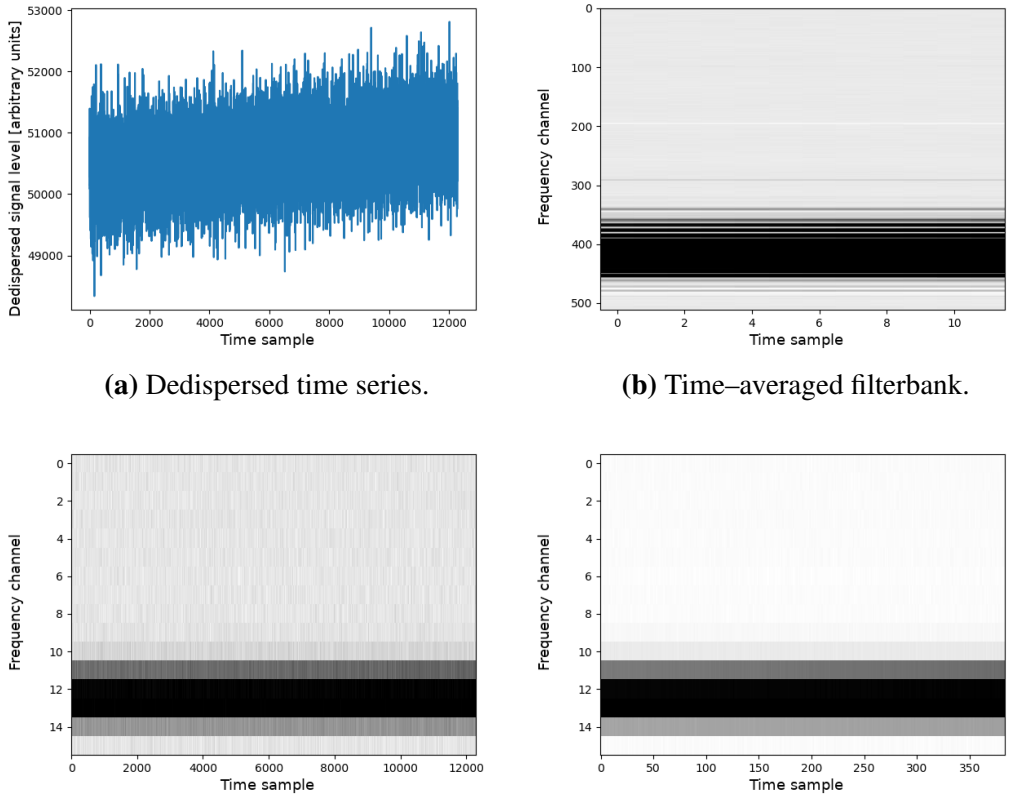
During the commissioning run in January 2018, we observed three positions on the sky where FRBs have been found previously. These were: FRB110523 - the only FRB to date discovered at 800 MHz with the Green Bank Telescope (Masui et al., 2015), FRB121102 - the repeating FRB (Spitler et al., 2016) and FRB130729 - a double-peaked FRB discovered at Parkes (Champion et al., 2016). Each of these sources was visible for 8 or more hours, with the repeater visible for more than 12 hours every day. The selection of sources in principle could have allowed for continuous observations,

Start time	Total hours	On-source hours
FRB110523		
2018-01-20 09:29:23	8.5	4.62
2018-01-21 11:32:26	5	5
2018-01-22 15:14:41	2.5*	2.5*
Total	16	12.12
FRB121102		
2018-01-19 20:22:23	7	5.97
2018-01-20 18:05:18	5	0
2018-01-21 16:38:58	10	5
2018-01-22 17:50:40	9*	9*
Total	31	19.97
FRB130729		
2018-01-20 03:23:28	0.25	0.25
2018-01-20 07:23:15	2	2
2018-01-21 04:39:37	5	5
2018-01-22 02:51:30	3	3
2018-01-23 03:12:02	2.5*	2.5*
Total	12.75	12.75

Table 4.2: Pointings and the number of hours spend on each source. *This may not be an accurate number, as these observation were run after we had lost the metadata stream with beams position information and the times had to be approximated from the telescope log files.

where at least one of the three sources was visible at any given time, as seen in Figure 4.10. In total, we have accumulated close to 45 hours of observations with the telescope on source, divided between the three main sources, as shown in Table 4.2. With a number of observations, we were not able to run the whole planned observing schedule due to external circumstances, such as bad weather conditions and telescope maintenance. For most of the observing we had the pointing information provided by the metadata stream. However, after the unexpected and undetected loss during the telescope maintenance, we had to rely on the telescope logs when analysing the data taken on the last day of observing. These were expected to be accurate, although it meant we did not have an independent pointing confirmation.

The data was recorded for offline processing, as the real-time detection pipeline was still under development and not ready for testing during the January run. We ran the search between the DM values of 0 and 2000 $\mathrm{pc\,cm^{-3}}$ and down to the S/N of 7.5 for bursts with widths between 0.054 and 221 ms. All of the beams were first processed individually and later all the candidates were correlated between different beams, with the detections present in more than 4 beams discarded as RFI. For the



(a) Dedispersed time series.

(b) Time–averaged filterbank.

(c) Filterbank dedispersed into subbands.

(d) Same as **(c)** but also time–averaged.

Figure 4.11: High-S/N FRB121102 candidate, which was found to be RFI. The plot that stands out the most is the time-averaged one, (b) which is meant to have 4 times samples across the whole burst width. The entire time series has been averaged to just 11 time samples which meant a large width (2^{11} samples in this case), a clear indication of RFI.

remaining candidates, additional plots were created similar to those shown previously in Figure 3.9. As at that point the pipeline was not designed to start the processing at the same data frame for every beam, the timestamps of candidates had to be adjusted so that they all had a common starting point. As a result, the MJD of the beam for which the data was recorded first was chosen as the start of the observation.

4.4.1 FRB121102

Additional processing was run over a smaller DM range between 500 and 600 $\mathrm{pc\,cm^{-3}}$, with the known DM of FRB121102 sitting almost in the middle at 557 $\mathrm{pc\,cm^{-3}}$. This was also the dataset that was used to develop the general processing techniques, with the main goal of reducing the number of candidates that had to be examined by hand

	2018-01-19	2018-01-21	2018-01-22
Original	100420	98449	162629
Errors removed	93226	88695	150939
Beam mask	80863	71883	128911
Width	70508	57292	113628
Members	27945	24748	44974
Total data reduction	72.2%	74.9%	72.3%

Table 4.3: Number of single-pulse candidates during each FRB121102 pointing. The cleaning procedure designed to remove as many false-positives due to RFI as possible has resulted in the reduction in the final number of candidates that had to be examined by almost 75% in all cases.

when running the search across the whole DM range and for other pointings. For the continuous 9 hour-long pointing, we detected a total of 21823 candidates across all the beams. With the candidates detected in more than 4 beams discarded as RFI and additional filtering of erroneous detections, such as a S/N of 0 or infinity, which is a result of incorrect normalisation, we were able to lower the number of the candidates by more than 25%, down to 16232, which were later examined individually using the additional plots. During this examination, 42 candidates were found to have a reported signal to noise ratio of more than 100. One candidate, with the highest reported S/N of 575.9, is shown in Figure 4.11, which can be classified as a false-positive with a high degree of confidence. Closer examination revealed that all 42 were caused by RFI and always had a reported burst width greater than 2^{10}, which was found to be one of the characteristic features of RFI. Upon confirmation using multiple datasets from different observations, it was decided the remove the candidates with widths greater than 2^{10} samples from further processing, significantly reducing the number of remaining candidates by additional 24%, down to 12406. Finally, we removed the candidates that had only 1 member in the group after the candidates in the neighbouring time samples and DMs have been merged and classified as a single candidate. Tests run on the RRAT and pulsar data have shown that the real signals always had single pulses detected in adjacent DM values and time samples and therefore more than one group member. At that point, we had only 5322 candidates left, a reduction of 75% from the original number of candidates - a more easily manageable number.

The same processing procedure was applied to all 3 observations of FRB121102 that had useful data recorded, with the telescope pointing at the source, as shown in Table 4.2, with the search pipeline run over the entire DM range described previously. Table 4.3 shows the number of candidates for each pointing after each of the cleaning

stages has been applied. It is evident that the employed procedure helped to remove a large number of candidates, with only around a quarter of the original dataset left in all three cases. Figure 4.12 shows the remaining single-pulse candidates. It can be clearly seen, that even after the extensive cleaning and removal of obvious false-positives, we were still left with a large number of candidates, many of which seem to be caused by bursts of RFI, indicated by the long vertical lines around the same time sample. It is therefore recommended to use more robust techniques to sift the candidates more efficiently in the future, such as machine learning, to speed the processing up and reduce the need for human intervention and examination of the candidates by eye. This can be achieved by a more efficient removal of RFI-caused candidates and more focus on the identification of true-positives, rather than just removing the false-positives.

After the data has been cleaned and each candidate has been examined individually, we report no bursts detected from the direction of the FRB121102 in the nearly 20 h of data collected during the three observations listed above. We consider 4 different possible causes of this situation, discussed below in the order of increasing probability:

Less data than expected. As already mentioned, some observations were run without independent confirmation of beams' positions from the metadata and were estimated from the telescope log files only. A 9 hour-long observation of the repeater was run in such conditions. The probability that the telescope log files were incorrect is low and additionally separate logs were kept on the pipeline starts and stops, the sources processed and conditions during the observing. These logs as well as the telescope logs do not contain any records of the telescope stopping the tracking during the observations that had no metadata information available. We can therefore safely conclude that the pointing information we had was correct and that we indeed spent 20 h observing in the field of the repeating FRB.

Burst loss caused by the pipeline. Here we consider the two processing stages separately. The filterbank stage of the pipeline has been tested extensively and it had been proven on multiple occasions that the version used during FRB observations was behaving correctly and not corrupting/missing any important and detectable chunks of data. The detection side utilises the single-pulse detection software which had been successfully used in the past to detect the majority of the currently known FRBs, including the repeating one. Considering the specific use of our pipeline, we have successfully demonstrated we were capable of detecting single pulses from various pulsars and J1819 − 1458. Although it is possible that a single burst could have been missed in the data, it was considered to be unlikely.

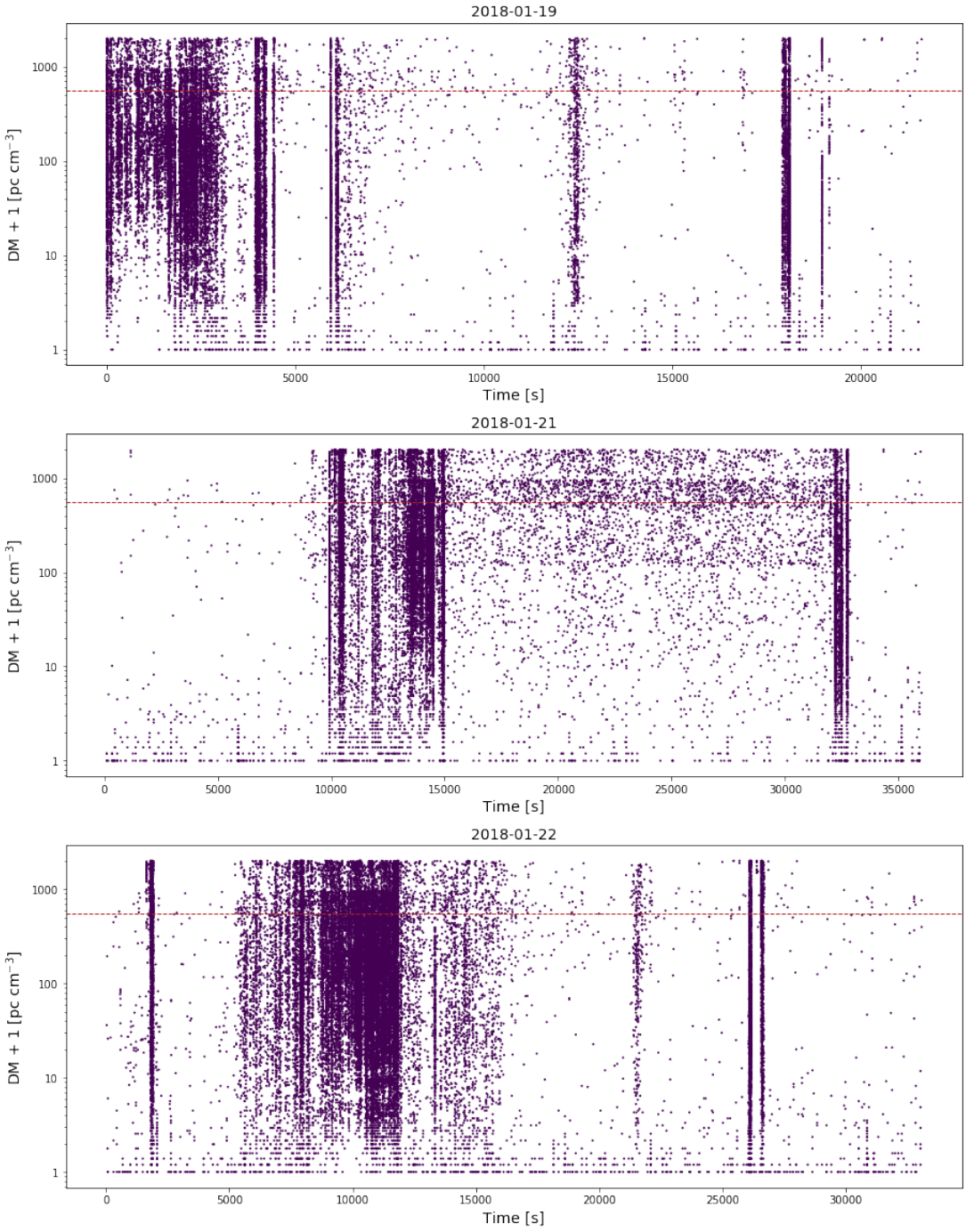


Figure 4.12: Single-pulse detections towards the position of FRB121102. All of them were confirmed to be false-positives. Strong RFI can be seen, manifested as vertical lines of candidates across a large range of DM values, concentrated around similar timestamp.

True-positives removed. As described above, we removed around 75% of the original candidates using a set of conditions that were found to best describe RFI. There is a non-negligible probability that the conditions we have selected were too broad and resulted in the true-positives being removed as well. These conditions were selected after tests were run using tens of thousands of candidates and were used to remove the most obvious RFI-caused detections only. Additionally, all of the 21823 candidates detected during the 9 h observing processed with the smaller DM rage were examined by eye and no detections were confirmed to be real bursts coming from the FRB121102. We are therefore confident that our selection criteria and the candidate reduction performed as expected and that only the most extreme cases of RFI were removed, leaving other, less obviously identifiable false-positive and any possible true-positives behind.

FRB121102 did not burst in that time period. FRB121102 has been found to repeat hundreds of times. During some observations multiple detections during a single pointing were also observed. Long periods of non-detections have already been reported though, including the lack of any bursts during continuous 5 and 10 hour-long observations (Price et al., 2018a). Assuming FRB121102 is like any other FRB, with the probability of detecting the event independent of any possible burst observed previously, its behaviour could be modelled successfully as following Poisson statistics. In such a scenario, the bursts would be expected to show no evidence of clustering and arrive at random intervals. If that was the case, we should be more likely to detect the repeating bursts during longer observations, with the total amount of time spent continuously on the sky being the most important parameter. Recent studies (Oppermann et al., 2018) have however shown that non-Poissonian statistics are a better fit to the emitting nature of the repeater. A Weibull distribution has been proposed, that could be used to better explain the clustering nature of the detected bursts, their irregular occurrences and no apparent detections during long pointings.

The general result suggests that burst from the FRB121102 are more likely to be seen in clusters and that long-duration observations with no detections can be common, much more than if this FRB was to follow Poissonian statistics. Using the derived parameters of the probability distribution function, with the average repetition rate of $5.7\,\mathrm{day}^{-1}$ (Price et al., 2018a), we calculated the probability of not detecting any bursts during the 9 h observation, where we would expect to see 2.14 bursts on average, to be 0.58. If we were to consider a standard Poisson distribution, with the same repetition rate, this probability then decreases to 0.12, which is 5 times less than in the case of the

Weibull distribution. It is important to note that the calculated distribution is not intrinsic do the observed object, FRB121102 in this case, but depends only on the detected bursts. It is therefore highly dependent on the sensitivity of the instrument used to detect the burst, which means that the parameters derived using the Arecibo telescope cannot be fully used when considering radiation from other telescopes. The average repetition rate and the shape parameter described above have also been derived using only a small sample of bursts, with many more burst detected since then and these values are expected to change. We cannot therefore conclude that our non-detection of any repeating bursts, even during our longest, 9-h observations, supports the clustered distribution. Even when a simple Poisson distribution is considered, which theoretically favours detections during long, continuous observations, we had more than 10% chance of not observing any bursts. This argument, combined together with other long observations taken as part of different surveys that failed to detect any repeated bursts, suggest that there were truly no detectable events, rather than anything that was missed during our processing.

4.4.2 FRB110523 and FRB130729

We have followed the same procedure as described above when examining the data from the positions in the sky where these two FRBs have previously been observed. In this case, we also found no bursts above the S/N of 7.5. To examine whether it was an unexpected result, we compare it to the current best estimate of the FRB rates. In our calculations, we use one of the most recent burst rates of $1.7^{+1.5}_{-0.9} \times 10^3\,\mathrm{FRB\,sky^{-1}\,day^{-1}}$ (Bhandari et al., 2018) at 1400 MHz. Assuming a system equivalent flux density of 28.13 Jy, 7.5σ detection, a burst with a width of 5 ms and an approximate operational bandwidth of 300 MHz, we are expected to detect 0.0132 FRBs during the ~ 25 h of observations, if we used the beams separated by FWHM. It would have therefore been unlikely to find any new FRBs during such short observing window and the lack of any detections is not surprising in this case. In practice, the number of expected bursts is even lower as our beams are overlapping, effectively reducing the instantaneous FOV on the sky. We therefore conclude that no detections during our observations have no statistical significance and does not allow us to draw any additional conclusions on the FRB burst rates.

Chapter 5

Single Pulse Simulations

5.1 Motivation for Simulating FRBs

With only a few dozen FRBs detected so far, and all but one detected at 1400 MHz, we have a limited understanding of their behaviour and bursting properties, especially at the lower end of the frequency spectrum at sub-1000 MHz observing frequencies. With only a single FRB detected at 800 MHz and none at lower frequencies despite continuing efforts, this poses a question: is the FRB radiation limited to frequencies above 800 MHz or are other contributing factors present that make it impossible to detect them with the pipelines that work so well at higher operational frequencies. One possible cause that could make FRBs undetectable is dispersion. As already discussed in Section 1.4, the delay between the signals arriving at the top and the bottom of the band is more prominent at lower frequencies. The amount of smearing within the individual channels, after the data has been channelised, is also increased. This means that bursts are smeared more and have reduced S/N, which can potentially make them more difficult to detect, especially in the presence of strong RFI. The RFI environment also changes significantly depending on the observing band used and also on geographical location of the observatory. Another factor which can possibly lead to the apparent non-detections and is common at all observing frequencies is the erroneous rejection of true-positives caused by too stringent selection criteria, such as the required S/N or the duration of the burst and also the follow-up examination by a human and/or machine learning.

Simulating our own FRBs can therefore answer a very important question in terms of non-detections: are we not detecting any viable candidates because there were truly no events or is our processing strategy simply causing us to miss all of the candidates?

We can also ask a similar question when FRBs are detected: do we see all of them or are we still missing a significant portion of them due to some incorrect assumptions in our processing techniques. Increasing the number of events that look like real FRBs as much as possible will also allow us to have a larger sample (artificial nevertheless) that can be used to better tune our selection and machine learning algorithms.

5.1.1 Simulation Framework

We therefore propose a framework for injecting simulated burst into single-pulse search pipelines. This can help to quantify and fix any possible shortcomings within the processing chain. Such a unified approach will allow for the whole pipeline to be tested in full, from the point of the data injection to the final candidates obtained from the single-pulse search software and confirmed by human and/or machine learning solutions. This has a considerable advantage over testing each stage of the processing separately as even though the correct output data can be obtained after a single processing step, independent errors present in different stages can eventually add up and corrupt the signal to the point that no detections are possible. This was observed for example during the PAFINDER pipeline tests, where individual simulated datasets were used to test each processing step independently. Such an approach is helpful during the initial development phases when potentially critical bugs can be discovered, such as incorrect memory addressing and when only a particular piece of code has to be optimised. However combining all the elements of the pipeline revealed unexpected errors as the different stages were either not communicating properly or a cascade of errors was causing the signal of interest to be corrupted and ultimately fully lost. Eventually we were able to confirm that the filterbank-generating part was working correctly with the help of the frequency sweep script. The ability to detect single pulses however was tested only during live telescope observations. This is not an optimal solution as any previously undiscovered errors mean that telescope time is lost on bug fixing and retesting the code, rather than used for scientific observations. The ability to simulate events is becoming more important every year as telescopes grow bigger and the accompanying processing systems become more complex, often making use of a combination of different programming languages and APIs and incorporating robust algorithms in places where human intervention was required even few years ago, such as RFI mitigation. With new facilities, such as the SKA, currently being under development and coming online in less that a decade, it is important to ensure that these will be able to deliver useful science from the very first day of operation.

Our proposed simulation framework would allow for thousands of candidates to be simulated, including rare, but potentially software-breaking edge cases, and occasional testing of the fully-operational system during day-to-day operation. The core part is formed by the single-pulse injection software, which will be discussed in more detail in following sections. This software is designed to simulate and inject the desired signal at the earliest possible stage, without having a significant impact on the overall processing speed. Depending on the use, this could mean creating a fake filterbank file if only the final detected data products are available, or injecting the simulated voltage itself and converting to the appropriate data format, such as CODIF when used for PAF observations or another standard. Initially, the simulated data can be generated in large volumes to better understand the limits of the telescope in use and the processing methodology. With the telescope in the operational state it can be used to randomly inject simulated bursts into the original data to test the ability of the software to pick up the rare events. This is especially important in the final stages of the processing when, independent of the techniques used to select true- and false-positives, human intervention is currently required. With sometimes hundreds to thousands of candidates that have to be examined by eye after single observation, this can lead to real candidates being lost amongst large volumes of false-positives. After making positive identification of the candidates, the user will be informed whether the selected detection has been simulated or not. The need for human intervention will most probably decrease in the future, as image recognition and classification algorithms are becoming more sophisticated, but even in such a scenario a large simulated dataset can be first used to train these and later evaluate their performance.

When used for evaluation purposes during normal telescope operation, it is important to ensure that the simulation software does not disrupt or alter the processing in any easily detectable way. It is therefore important to reduce the processing time so that the burst can be simulated and injected in the shortest possible amount of time. The injection software also has to use the smallest possible amount of resources and not save too much data to the disk. These steps are necessary to ensure that no bias is present when evaluating the ability to identify events - e.g. if the injection step introduces a significant increase in the pipeline processing time, a user might know which pointings should be examined more carefully. Even when no human intervention and accuracy is meant to be tested, the simulation time has to be as short and resource-efficient as possible to limit the use of the computational power and allow for the original signal and the one containing the fake injected burst to be processed concurrently.

5.2 Simulation Overview

Our single-pulse injection software is the first step towards a fully-operational simulation framework. The code for this has been developed fully in C++, with the performance-critical stages utilising the power of GPU computing. The simulation procedure is designed to be modular, with products after each stage stored independently, which makes it easier to modify existing stages and add additional processing stages as necessary. Having access to intermediate data products allows us to estimate the effects of each of the processing steps independently and get a better understanding of which one of them influences our probability of detecting FRBs. The current simulation implementation generates the final products in the form of SIGPROC filterbank files. As the simulation starts as early as possible in the digital processing chain, with simulated raw voltages, we have the flexibility to add support for a variety of different final output formats, including CODIF, which can later be used to simulate FRBs and check our ability to detect them with the PAFINDER pipeline. The modularity of the code means that additional steps can be added in the future, such as scattering, that will better approximate the existing FRB population as the available sample size increases and our understanding of the nature of FRBs becomes more complete.

5.2.1 Voltage Generation

The voltage is first generated in a way that approximates the white noise present in all radio-astronomical observations. As we are concerned with the white noise at this stage, an array of pseudorandom numbers is generated, drawn from a normal distribution, with zero mean and unit standard deviation. Both the real and imaginary parts are generated, as we want to simulate complex voltages which are more practical for the coherent dispersion implementation used at the later stages of the simulation. For this purpose we used a cuRAND library, which allows us to generate the pseudorandom numbers directly on the GPU, using the Mersenne Twister engine (MT19937), which has a large period, larger than any viable voltage series we should ever generate, and is currently one of the most popular and recommended engines for the purpose of pseudorandom number generation. With a total bandwidth of B, the generated complex signal has a sampling time of $1/B$s, i.e. we need to generate tens to hundreds of millions of samples for each second of signal, depending on the bandwidth used during the observations. It can therefore be seen that memory constraints will be one of the biggest challenges in running efficient simulations.

The complex voltage is then multiplied with the envelope of the burst. For the majority of simulations, bursts are approximated by a Gaussian function with the FWHM equal to the supplied burst width, described as

$$f(x) = n\exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{5.1a}$$

$$\text{FWHM} = 2\sqrt{2\ln 2}\sigma, \tag{5.1b}$$

where n is the amplitude of the Gaussian, μ is the position of the centre and σ is the standard deviation, used to quantify the width of the function. Additional shapes were considered, in particular the von Mises function, which closely resembles the Gaussian function, but has smaller tails, i.e. it goes to zero faster than the standard Gaussian function. Additionally, an option for a user-supplied profile was developed at the beginning, but was later removed for the main simulations and will be reinstated in the future. As the amplitude of the Gaussian is a configurable, user-supplied option, it is important to ensure that the tails would meet the noise at the correct level without erroneous jumps introduced. The burst envelope was therefore generated between points at what we call the FWUA - Full Width at Unit Amplitude, where the Gaussian function reaches unity, i.e. $1/n$ of the amplitude. This value can be derived using the expression for the Gaussian function as

$$\text{FWUA} = 2\sqrt{2\ln n}\sigma = \text{FWHM}\sqrt{\frac{\ln n}{\ln 2}} \approx 1.2\ \times \text{FWHM}\sqrt{\ln n}. \tag{5.2}$$

The burst points not included in this range are assigned a value of 1, which returns the original white noise voltage after the multiplication. The resulting real part of the burst voltage can be seen in Figure 5.1. This particular burst had the amplitude of the Gaussian function set to 10.0. It can be seen that when combined with the voltage, it can go as high as ~ 40. This is however not surprising as the voltage generated using Gaussian distribution with the standard deviation of 1 is expected to have $\sim 32\%$ of values to lie beyond the 1σ boundary. The burst envelope can also be seen to join the 'noise' portion of the voltage at a correct level, without any jumps at the boundary. This is more clearly visible in the zoomed-in portion of the burst, focusing on the trailing tail joining the voltage.

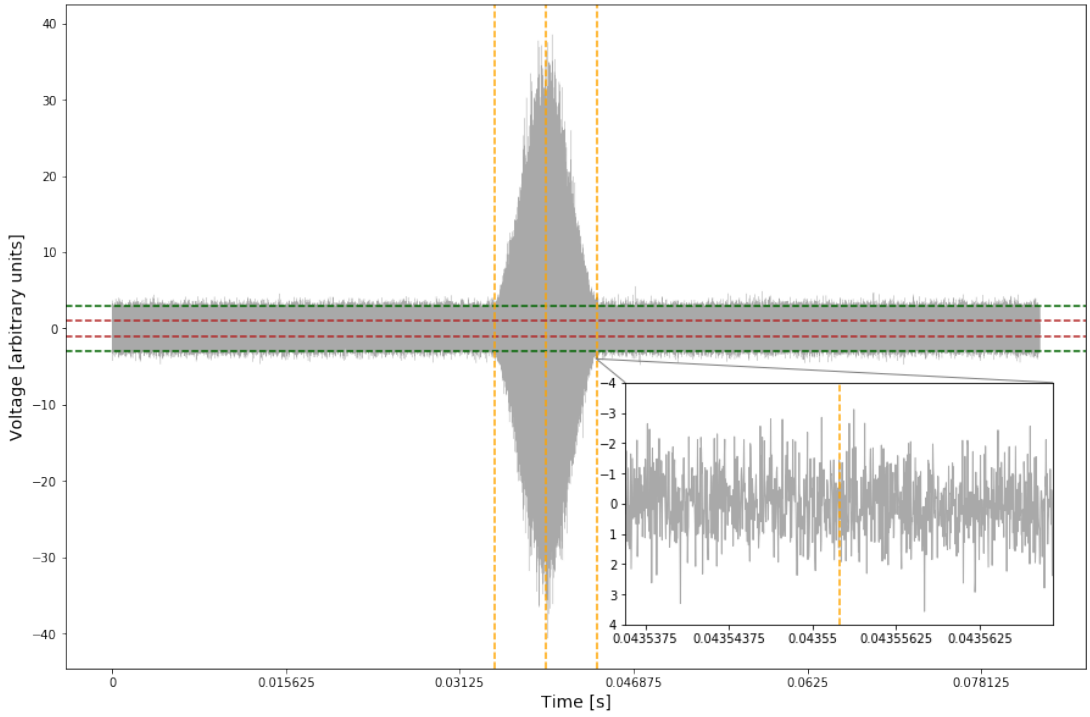


Figure 5.1: Simulated FRB voltage. Orange lines shows the burst boundaries and its centre, red lines and green lines show 1σ and 3σ voltage levels respectively. Inset shows the region where the burst envelope joins the white noise, showing that there are no jumps introduced and the transition is smooth at 1σ level.

5.2.2 Dispersion

We employ two main different dispersion methods: fully coherent and semi-coherent. The semi-coherent method in turn uses two different approaches, which solve the problem and disperse the burst in slightly different ways. The basic principle is the same for all of these methods as the generated voltage, or some part of it, is dispersed with the help of the phase-only transfer function, which simulates the effects of dispersion. The transfer function has the same form as Equation 2.2 and is multiplied with the frequency spectrum of the incoming signal, obtained using the FFT.

5.2.2.1 Fully Coherent Dispersion

The fully coherent dispersion can be considered the most straight-forward approach that is easy to implement. Here, we have to generate a voltage array long enough to hold the entire dispersed simulated burst. This ensures that after the final filterbank is generated, the burst is in one piece and not fragmented. The 'leakage' of additional bursts is caused by the periodicity of the FFT, and it is therefore important to add sufficient padding to prevent this in the properly dispersed filterbank. As the transfer function is expressed with respect to the central frequency, the amount of padding added to the front would generally be different to the amount of padding at the back of the burst. If we consider only the shorter padding at the front, resulting from the delay between the top of the band and the centre frequency, the total number of required samples is equal to

$$\Delta s = 2 \times 4.15 \times 10^3 \times \left(\left(\frac{f_{\text{centre}}}{\text{MHz}} \right)^{-2} - \left(\frac{f_{\text{top}}}{\text{MHz}} \right)^{-2} \right) \times \left(\frac{\text{DM}}{\text{pc cm}^{-3}} \right) \times \left(\frac{B}{\text{Hz}} \right), \quad (5.3)$$

where B is the total bandwidth over which the signal is being recorded and is equal to the sampling rate as we are generating complex voltage ($1/B$ is the sampling time) and the factor of 2 at the start reflects the fact we apply the same padding at the start and at the end of the signal. Assuming an approximate centre frequency of 330 MHz and a total bandwidth of 32 MHz, we need 220359 time samples per unit DM. Using two single-precision floating point numbers to express the real and imaginary parts of the simulated voltage, this corresponds to ~ 1.76 MB of memory used to store the data. This is not a problem when relatively low dispersion measure values are used, but the memory requirements increase linearly with the dispersion measure. At a DM of 1000 pc cm^{-3}, a region where a lot of known FRBs reside and therefore often used

during our simulations, the simulation already requires 1.76 GB of memory for the voltage generation stage alone, which can be a significant proportion of the total memory available on the GPU (e.g. GeForce GTX 980 has only 4 GB of on-board memory available, which makes it impossible to simulate bursts with DM $\geq 2000\,\mathrm{pc\,cm^{-3}}$ using this direct method). To ensure we have enough datapoints to create a correct output, we also have to generate a number of samples that is a multiple of the number of channels in the filterbank obtained at the end of the processing. This also increases the number of data points, but has a much smaller total contribution to the final number of the generated voltage samples, as in the worst-case scenario 1023 additional samples are generated - less than 0.5% of the padding required per unit DM of delay.

The large number of samples also increases the processing time, mainly during the FFT, where the voltage is changed from time to frequency space in order to be multiplied by the transfer function and also requires the transfer function to span the entire frequency spectrum. After the dispersion has been applied, the inverse Fourier Transform is used to bring the signal back into the time domain. A filterbank file is then created by dividing the signal into N_t, N_c-channel wide chunks and running the forward FFT for each chunk, where $N_c \times N_t = N$ - the total number of generated voltage samples. It is at this stage that our choice of the number of samples being a multiple of N_c becomes important - we require that N_t is an integer. The signal is then detected, which combines the real and imaginary parts of the voltage into the total power, and stored according to the SIGPROC filterbank file format specification. As we are using complex voltage in our simulations, the series of N_c-point FFTs results in a file with N_c channels and N_t time samples. Using fully coherent dispersion has an advantage over the other methods described below in that using a single multiplication of signal in the frequency domain with the transfer function fully disperses the burst, including the intrachannel smearing in the channelised data product as well as the interchannel dispersion delay. However the large memory requirements and long processing time make it impractical to use in a system where a large number of bursts have to be simulated in a short period of time, especially when using older graphics cards.

5.2.2.2 Semi-coherent Dispersion

The semi-coherent approach uses both the coherent and incoherent dispersion to fully disperse the signal. The coherent part is used to apply the smearing to the individual filterbank channels, so that the burst is correctly widened. The incoherent dispersion is then applied to delay the arrival time of the burst across the whole band, with respect

to the top channel. We have chosen the highest frequency channel in this approach as the dedispersion algorithm we use as part of our single-pulse detection pipeline dedisperses the signal with respect to the highest frequency channel, and therefore the burst arrival time is approximately equal to the burst start time in that highest channel. The two different methods mentioned above use different approaches to the coherent part of the dispersion process. They both however allow us to reduce the number of samples required to generate the burst as they do not require as much padding as the fully coherent approach.

The first variant of the semi-coherent method is used to create the filterbank file of the non-dispersed burst immediately after the burst voltage generation. Similar to the procedure described above, we divide the simulated, but not yet dispersed burst voltage into N_c-point long chunks and run the forward FFT on each one of them. This time however, the signal is not detected, which means we leave the data in its complex form, therefore preserving the phase information required for coherent dispersion. We then run a series of N_c, N_t-point forward Fourier transforms, which effectively provides us with a finer channelisation for each of the N_c original filterbank channels. At this stage we can run the coherent part of the dispersion algorithm. Each of the frequency spectra is multiplied by the corresponding transfer function, adjusted to the centre frequency and bandwidth of the individual filterbank channels. Dispersing each channel individually smears the pulse within the channel, but the interchannel dispersion is not performed, which means we can now use a much smaller padding. The amount of extra samples that have to be generated is now dictated by the amount of smearing in the lowest frequency channel, where the effects of dispersion are the greatest, and the increase in width is given by Equation 1.12. Using the same example as above, with a central frequency and bandwidth of 330 MHz and 32 MHz respectively, for the 1024 channel filterbank requires less than 300 additional samples per unit DM, a significant reduction from the 220359 required for the fully-coherent approach. After the burst is widened, the inverse Fourier transform is run to return to the time domain in the individual filterbank channels. The signal is also detected at this stage. The resulting filterbank containing the total power can later be incoherently dispersed by delaying the individual channels to obtain the fully dispersed burst.

An example of the application of the first stage of this semi-coherent method used to widen the bursts is shown in Figure 5.2. In this case we simulated a 5 ms duration burst at a DM of 3000 pc cm^{-3}. It can be clearly seen that this method does not require a large amount of padding in order to apply the intrachannel dispersion. By comparison,

a fully-coherent approach would result in more than 4 million output time samples of padding. The red line on the bottom plot shows the theoretical boundary of the smeared burst, as calculated using Equation 1.12. It can be seen that the burst is indeed wider than the original one and that its broadening follows the theoretical expectations, meaning that the semi-coherent method can provide us with accurate results, even though it is an approximation of the fully-coherent approach.

The second approach to the semi-coherent dispersion is a modified version of the method described above. Similar to the fully-coherent approach, we start by performing the Fourier Transform of the whole original burst voltage. This time the padding again has to accommodate only for the smearing in the lowest final frequency channel and not the whole dispersed burst, like in the fully-coherent approach. After the FFT, we divide the frequency spectrum into N_c chunks, each with N_t spectral points. Each chunks is then multiplied by the transfer function, adjusted to the central frequency of the spectral chunk being processed. After the dispersion transformation, each chunk undergoes an inverse Fourier transform, moving it back to the time domain. The result is a filterbank with N_c channels with N_t time samples each, with a properly widened burst inside. The incoherent dispersion and detection is run in the same way as with the first semi-coherent method. The comparison of the intermediate dispersion stage (before the incoherent dispersion) between this approach and the previous one, shown in Figure 5.3, indicates that both methods result in a correctly smeared pulse which follows the theoretical calculations. This is also the method that has been used in the majority of our simulations, as if offers all the advantages of the semi-coherent dispersion and uses less intermediate steps and short Fourier Transforms than the first semi-coherent method described above, which can all have a negative impact on performance and increase algorithm complexity.

The implementation of all three dispersion methods is summarised in Figure 5.4. This visual representation makes it clearer that the semi-coherent methods are more complex and involve more steps to achieve the same result of simulating the fully dispersed burst. Even though additional steps are introduced in the form of multiple forward and inverse Fourier Transforms, both semi-coherent dispersion methods perform much better and they are considerably faster than the fully-coherent approach. Initial tests have shown that at large dispersion measures, the fully-coherent approach could take up to 30 minutes to simulate the burst, while any of the semi-coherent methods can complete the same task in less than a minute due to the significantly reduced size of the input signal.

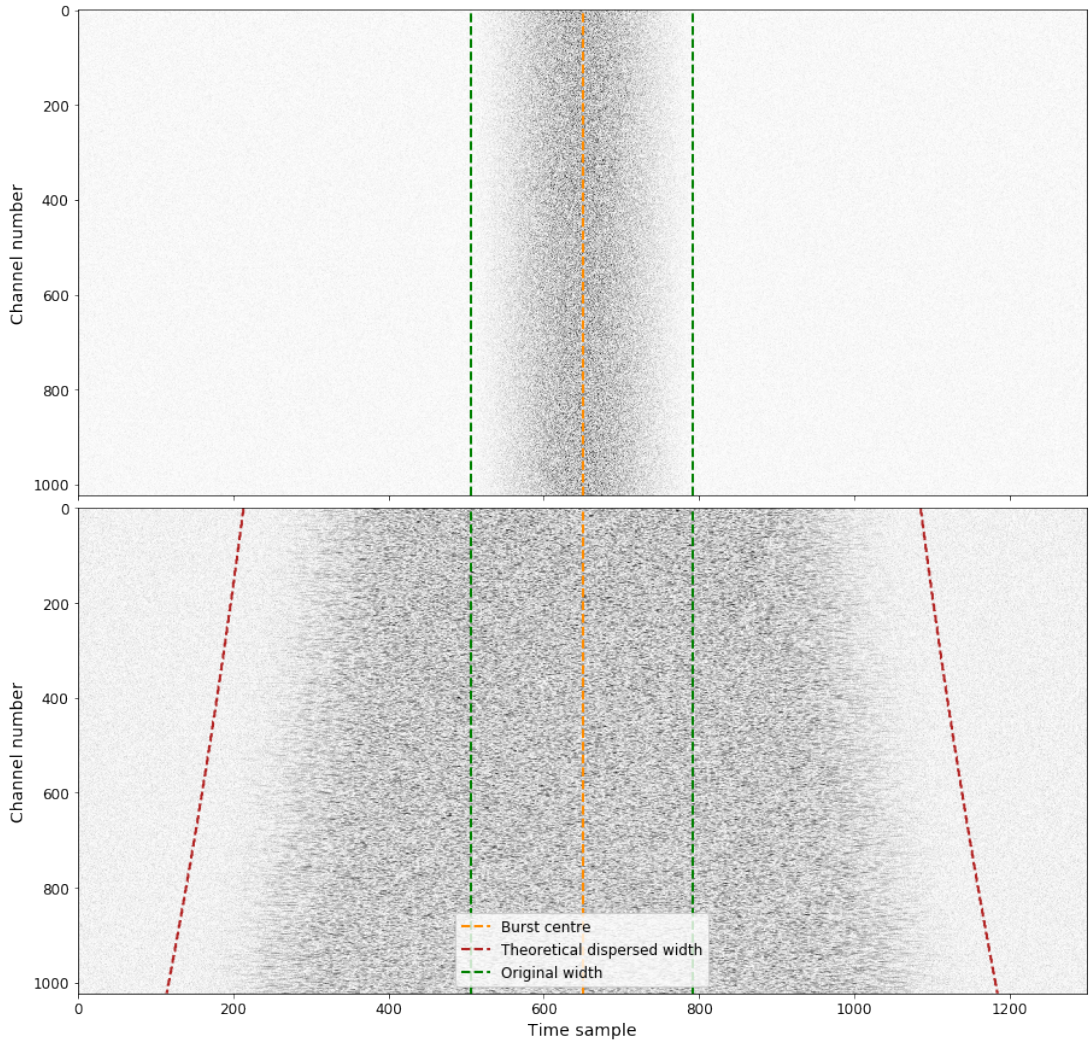


Figure 5.2: Example of the semi-coherent dispersion. The original signal in the top panel is a 5 ms FWHM Gaussian burst. The application of semi-coherent dispersion correctly smears the pulse across the filterbank channels and the increase in width closely matches the theoretical expectations.

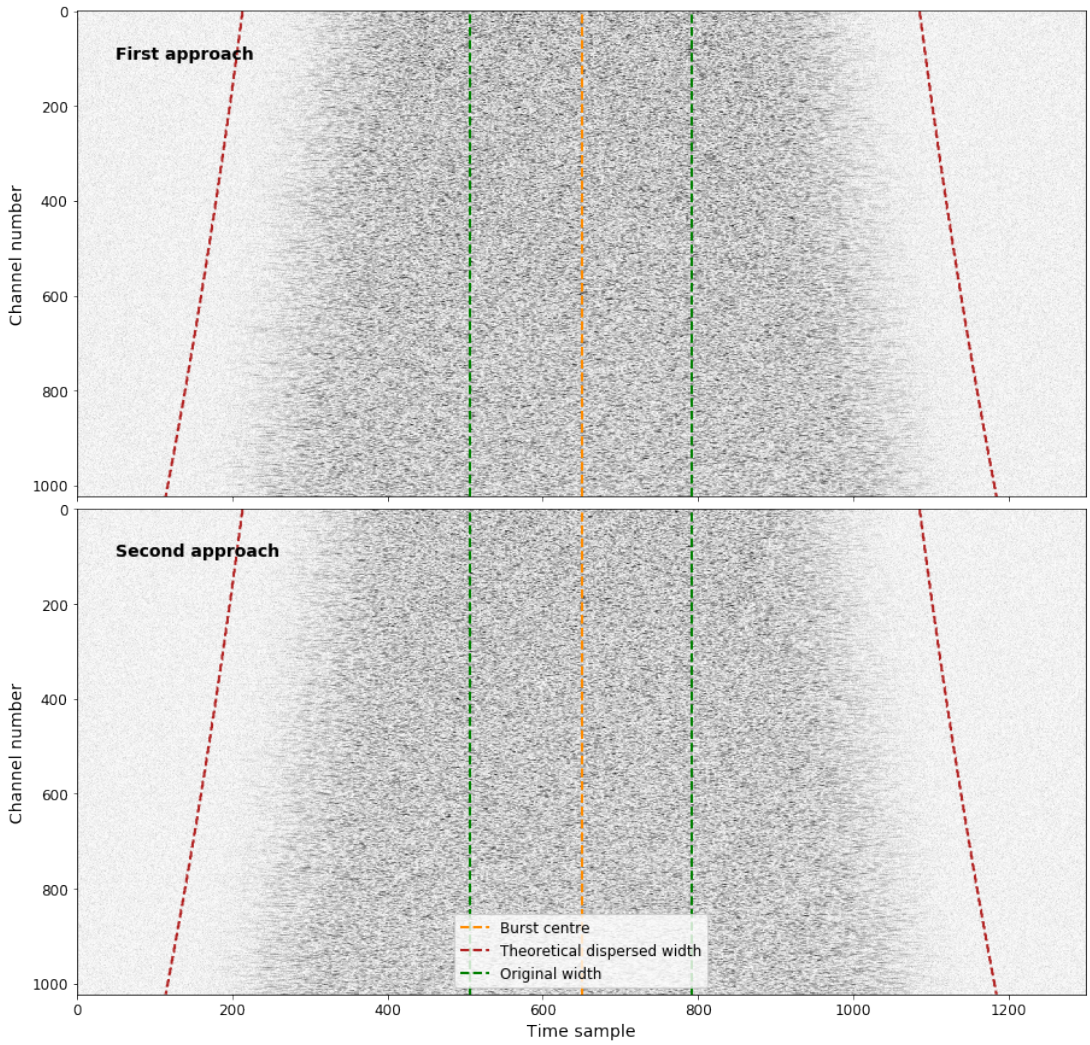


Figure 5.3: Semi-coherent dispersion burst comparison. It can be seen that for both of the methods used in our simulations, the resulting burst is properly widened.

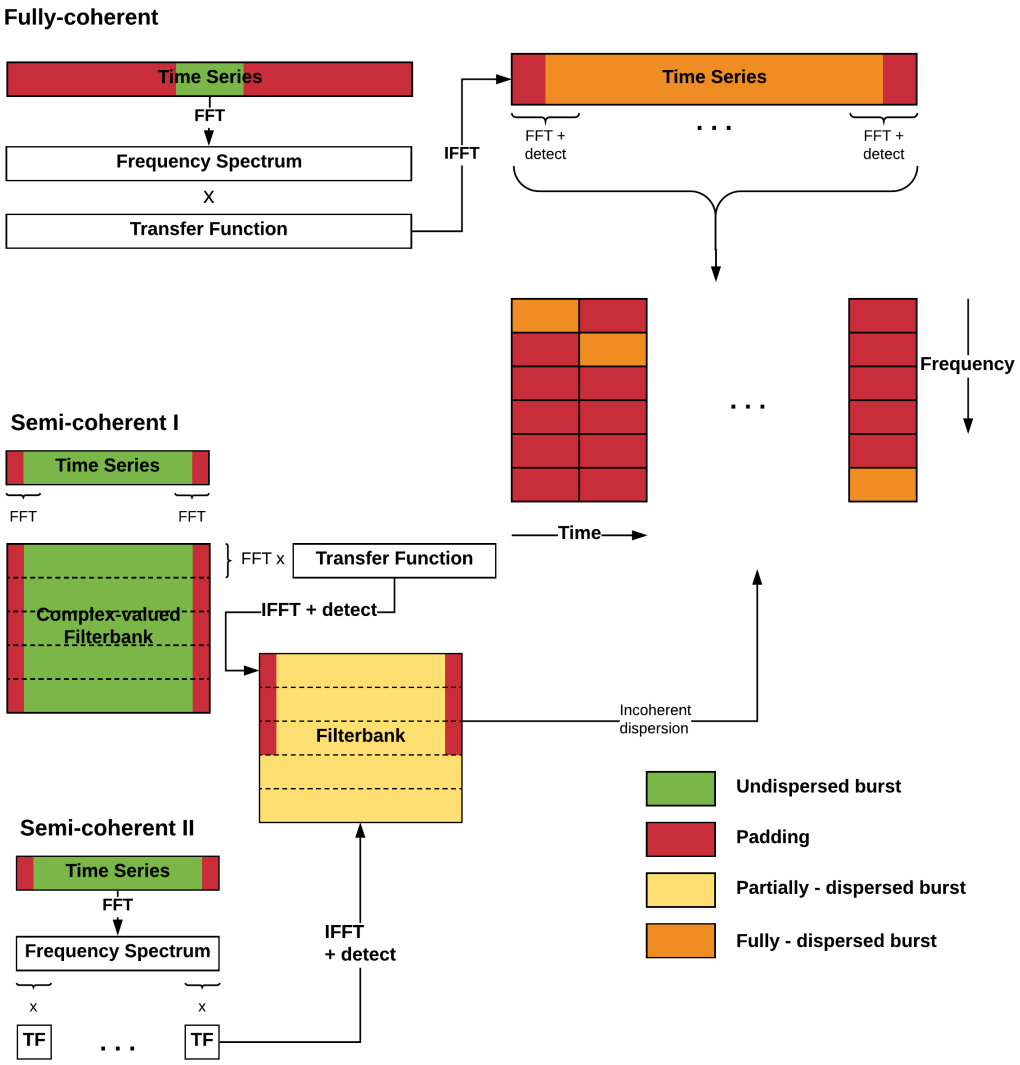


Figure 5.4: Dispersion methods workflow. The semi-coherent methods use more intermediate steps to obtain the same result as the fully-coherent approach. The increase in the processing speed of the order of 10 $\times$ is achieved despite the increased complexity, due to the lower total data volume.

5.2.3 Burst Injection

The last stage of the simulation process concerns the injection of the generated burst into the existing filterbank file. In our simulations we have chosen to save the bursts into real data recorded during observations rather than simulating all the observation data as well. This allows us to examine the effect of RFI and various other signals, such as the presence of pulsars, on our capabilities of detecting the burst. Even though this approach means we had no control over the characteristics of noise in the data, this greatly simplified the simulation process as it removed the need to generate synthetic, realistic-looking RFI.

As we are dealing with real data which has already undergone some degree of processing (e.g. raw voltage to filterbank conversion) at the time of the burst injection, we have to make sure that our generated signal covers the same range of values as the data it is meant to be injected into. We therefore have to scale the burst data so that, when injected, no significant deviations in terms of signal mean and standard deviation are introduced and the general shape of the bandpass is preserved. As the bandpass can vary in the original data file, we have to make sure our scaling takes this into account and includes fluctuations consistent with the changes in the bandpass. We achieve this using the scaling operation previously described in Section 3.2.4, this time divided into two stages. First, we individually scale each channel of the burst filterbank such that they all have $\mu = 0$ and $\sigma = 1$. We then multiply these scaled values with the standard deviation of the corresponding channel in the original data. After this final scaling stage, the scaled burst data is added on top of the input filterbank file, with the incoherent dispersion applied at this stage in the case of the semi-coherent algorithms. This is achieved by simply saving the individual burst channels with the correct shift introduced corresponding to the interchannel dispersion delay. After the burst data is added, the resulting filterbank file is saved to the disk for further processing.

Chapter 6

GHRSS survey

6.1 Survey Description

The Giant Metrewave Radio Telescope is an array of 30 parabolic dishes, each with a diameter of 45 m located in Western India (Ananthakrishnan & Pramesh Rao, 2002). They are spread in an approximate 'Y' configuration, with 12 antennas distributed semi-randomly in the central area of 1 km^2 and the rest placed uniformly in three arms. The longest possible baseline between antennas is 25 km. Observations are possible in six frequency bands in the range 50–1420 MHz, with the software backend providing support for a bandwidth up to 32 MHz.

GHRSS, the GMRT High Resolution Southern Sky survey, is making use of the GMRT capabilities to detect new radio transients, including regular and millisecond pulsars, and Fast Radio Bursts. The survey operates at a centre frequency of 322 MHz, which provides good system temperature and a large size of the primary beam from the range of available parameters (Ananthakrishnan, 2005), providing an increased instantaneous field of view. The survey makes use of both coherent and incoherent beams, with the incoherent beam used during the search phase, when the largest possible field of view is desired. Coherent beam is used for the follow-up observations of new candidates.

Two ranges of galactic latitudes are observed, with longer integration times used for regions closer to the galactic plane, as shown in Table 6.1, due to the increased sky temperature contributions in these regions of the Milky Way. The choice of the sky coverage partially fills the gap in the southern sky coverage currently not observed by other surveys operating at similar frequencies, such as the Green Bank Northern Celestial Cap survey, which can cover declination down to -40° (Stovall et al., 2014).

	Mid–galactic latitude	High galactic latitude
Galactic latitude	$5^\circ < \|b\| < 20^\circ$	$\|b\| > 20^\circ$
Declination	$-54 < \text{dec} < -40$	$-54 < \text{dec} < -40$
Integration time	20 min	15 min
Sampling time	61.44 μs	30.72 μs
Bandwidth	33.3 MHz	33.3 MHz
Number of channels	2048	1024
Number of pointings	682	911
Data per pointing	37 GB	28 GB
Total data	25 TB	25 TB

Table 6.1: GHRSS survey parameters.

The GHRSS survey also covers portions of the sky that were last time scanned at lower frequencies more than two decades ago, with the most recent one, The Parks Southern Sky Survey operating at 436 MHz, completed in the mid-1990's (Manchester et al., 1996).

Depending on the operational mode, the data can be recorded with 1024 or 2048 channels across the entire frequency band. With sampling times of 30.72 μs and 61.44 μs respectively and 8 bit sampling, the survey produces 2 GB of data per minute of observation. As can be seen in Table 6.1, the amount of data per pointing is high, approaching 40 GB at mid-galactic latitudes and in individual cases can reach up to 60 GB when longer observations of known pulsars are performed. Additionally, visibilities data are recorded every 2 s which will allow better localisation of any pulsars or FRBs discovered during the survey.

As has been highlighted before, it is especially important to process the data in real-time or near-real-time when searching for Fast Radio Bursts, as these detections are necessary for any possible follow-up observations. In the case of the GHRSS survey however, our focus was not on the real-time capabilities of the system as the processing was not done on the data as it was being recorded by the telescope, but was run weeks or even months after the initial observation, after the data has been shipped and copied to our processing facilities. Our focus was therefore shifted towards more thorough identification of possible candidates and development of new and improvement of existing algorithms that would allow us the better understand the effects of RFI and various processing and post-processing stages that could theoretically impact our ability to detect any new transient phenomena.

6.1.1 Survey Limits

Operating at a centre frequency of 322 MHz, we benefit from a lower receiver temperature and sky temperature at higher galactic latitudes, estimated to be 53 K and 40 K respectively. Combined with the contributions from the ground temperature, the total system temperature, $T_{\rm tot}$ is 106K (NCRA, 2006). We can use this to estimate the minimum detectable flux from a pulsar as

$$S_{\rm min} = \frac{\sigma T_{\rm tot}}{G\sqrt{BN_pN_at}}\sqrt{\frac{w}{P-w}}, \tag{6.1}$$

where σ is the required S/N, G is the antenna gain, 0.32 K/Jy at 322 MHz, B is the total operational bandwidth, N_p is the number of polarisations, N_a is the number of antennas combined incoherently, t is the integration time, P is the pulsar period and w is the pulse width. In our calculations we use $\sigma = 5.0$, $B = 32$ MHz, $N_p = 2$, $N_a = 30$, $t = 900$ s and a duty cycle of 10%. Using a simple constant width approximation we obtain a minimum obtainable survey sensitivity of 0.42 mJy. This value will however depend of the amount of dispersion and scattering smearing and will increase with the dispersion measure. We therefore include the dependence of the burst width on the dispersion, increasing the pulse width according to Equation 1.12. Pulse broadening due to scattering over a DM range, τ_s expressed in milliseconds, is derived from (Bhat et al., 2004)

$$\log\tau_s = a + b\log{\rm DM} + c\,(\log{\rm DM})^2 - \alpha\log(f_G), \tag{6.2a}$$

$$a = -6.46, b = 0.154, c = 1.07, \alpha = 3.86, \tag{6.2b}$$

where f_G is the centre frequency expressed in GHz.

Figure 6.1 shows the results of our calculations for pulsars with 8 different periods between 2 ms and 1 s. The duty cycle is kept the same in all cases at 10%. We can see that our approximate value of 0.42 mJy can be used reliably at low dispersion measures, up to about 25 pc cm^{-3} for all considered pulsar periods and becomes increasingly inaccurate with the increasing dispersion measure, mainly due to the broadening caused by scattering. We can now make a more informed decision on the allocation of computational resources and significantly lower the upper limit of the processed DM range when searching for millisecond pulsars and ordinary pulsars. It is clear that even for the highest considered period of 1000 ms, the pulse is sufficiently broadened so that the total smearing exceeds the pulse period close to 400 pc cm^{-3}, which is only

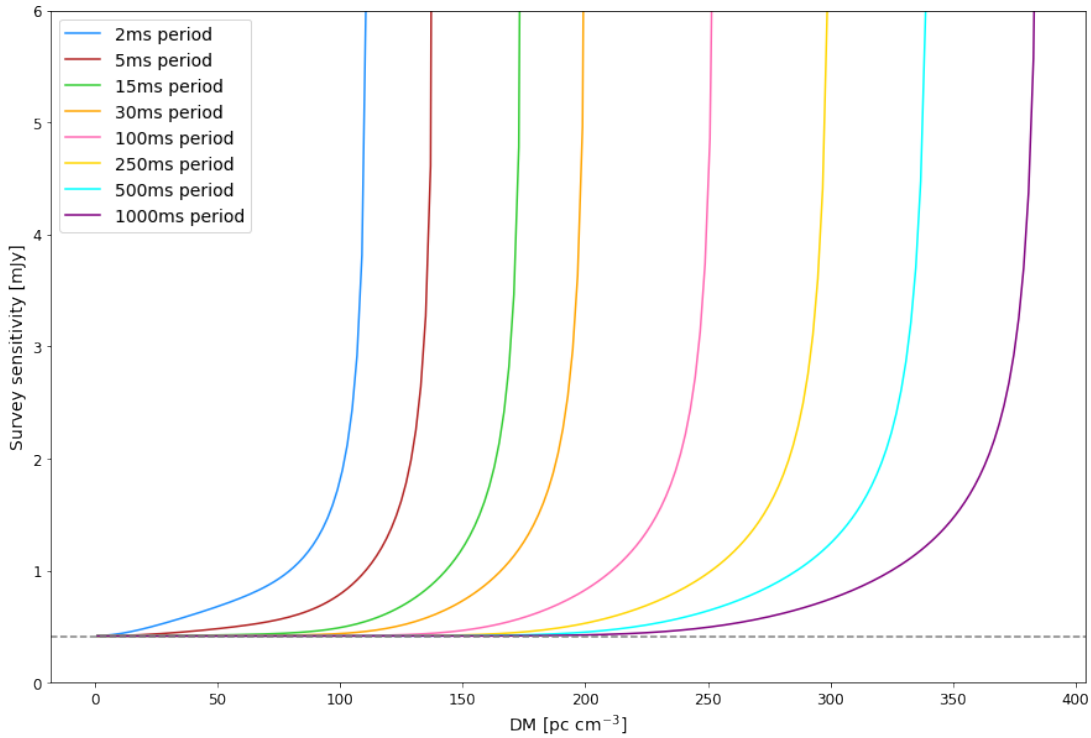


Figure 6.1: GHRSS pulsar sensitivity as a function of dispersion measure, for pulsars with different periods and a constant 10% duty cycle. Effects of dispersion and scattering on pulse broadening are included. The dashed line represents the minimum sensitivity for a burst with a constant width at 0.42 mJy.

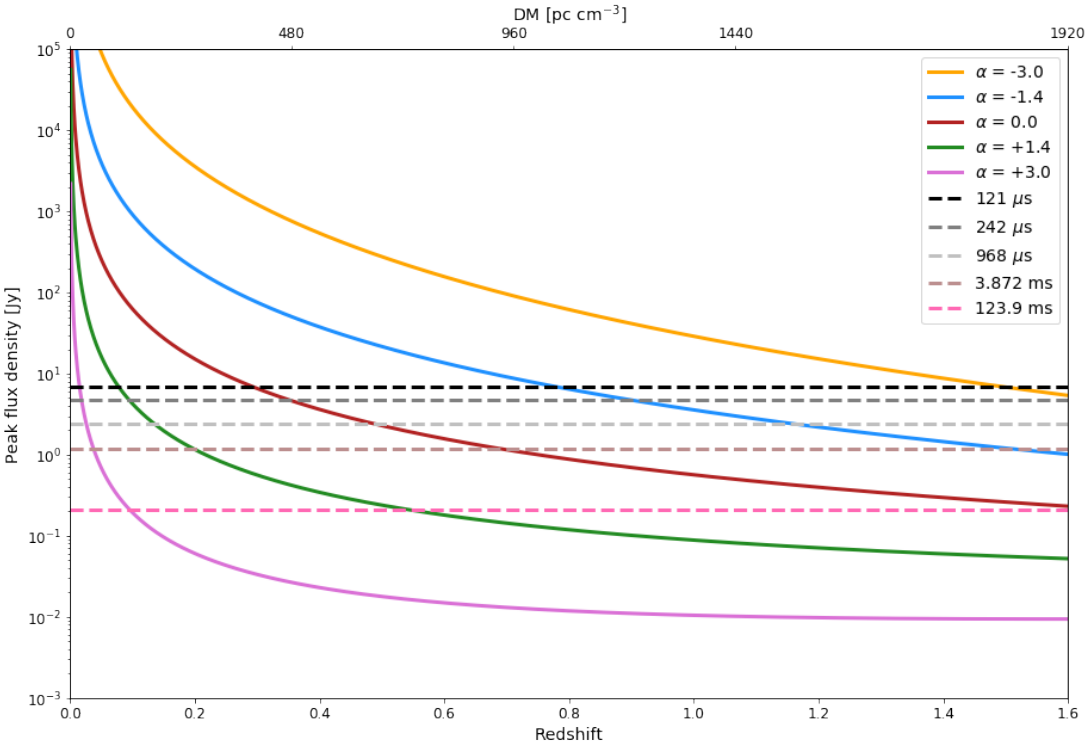


Figure 6.2: GHRSS survey sensitivity to FRBs with different widths used for matched-filter boxcar function and following power laws with different spectral index values. FRBs are assumed to be standard candles in the universe described by ΛCDM cosmology.

20% of the total covered DM range. The DM range is further reduced when searching for millisecond pulsars, where the broadening limits our useful search space down to dispersion measures lower than 200 pc cm^{-3}, which is only 10% of the total space processed.

We can perform similar analysis when considering single-pulse detections which are important from the point of view of the FRB survey. We use the following relationships to describe the dependence of peak flux density on the FRB spectral index α,

redshift z and the comoving distance $D(z)$ (Lorimer et al., 2013):

$$S = \frac{L(1+z)^{\alpha-1}}{4\pi D(z)^2 \left(f_{\text{high}}^{\alpha+1} - f_{\text{low}}^{\alpha+1}\right)} \left(\frac{f_t^{\alpha+1} - f_b^{\alpha+1}}{f_t - f_b}\right), \tag{6.3a}$$

$$D(z) = \frac{c}{H_0} \int_0^z \frac{dz'}{\sqrt{\Omega_m (1+z')^3 + \Omega_\Lambda}}, \tag{6.3b}$$

$$f_{\text{high}} = 10\,\text{GHz}, f_{\text{low}} = 10\,\text{MHz}, \tag{6.3c}$$

$$\Omega_m = 0.32, \Omega_\Lambda = 0.68, H_0 = 68\,\text{km}\,\text{s}^{-1}\,\text{Mpc}^{-1}, \tag{6.3d}$$

where the relationship between the redshift and the dispersion measure is approximated using Equation 1.3. We assume a simplified model where the contributions from the Milky Way and FRB host galaxy ISM to the measured DM values are negligible, and the dispersion is dominated by the Intergalactic Medium. To obtain the correct relationship between flux and redshift, the luminosity has to be scaled as a function of the spectral index. This assumes a peak flux density of 1 Jy at 1.4 GHz and reference redshift of 0.75 (Lorimer et al., 2013).

Figure 6.2 shows simulated FRB peak flux density assuming that they are standard candles and universe is described by the ΛCDM cosmology with the matter, Ω_m and dark energy, Ω_Λ total energy densities, as given by Equation 6.3d. The dashed horizontal lines represent the sensitivity limits of our survey calculated for pulse widths between 121μs and 123 ms, which are the practical lower and upper limits used for the data recorded using 2048 channels and sampling time of 61.44 μs. As the spectral index of FRBs is currently unknown, we consider different scenarios, with $\alpha = -1.4$ being the current best estimate for the mean pulsar spectral index (Bates et al., 2013). It can be seen that our ability to detect bursts with the assumed positive spectral index decreases the most rapidly, with all burst widths searched for during processing undetectable at redshifts greater than 0.6. If FRBs were to have a large positive spectral index, such as $\alpha = +3.0$, we are expected to observe only the closest ones, with less than 10% of the entire DM range detectable for bursts with widths less than approximately 4 ms. With the median width of all FRBs reported so far at 4 ms, we can assume that all FRBs are of a similar width. In such a case, the GHRSS survey is sensitive to bursts across most of the probed redshift range only if they were to have negative, i.e. similar to pulsars and lower spectral index, with $\alpha = -1.4$ and $\alpha = -3.0$ being the only values under consideration that would allow us the fully search the entire DM rage for bursts with widths less than 4 ms. We use conservative values of the expected

CPU	2 × Intel(R) Xeon CPU E5-2620 v2, 2.10 GHz, 6 physical cores
RAM	128 GB
GPU	4 × GTX 980, 4 GB memory

Table 6.2: Specification of GHRSS processing nodes.

spectral index, but it is important to note that for the bursts which had values of α measured (currently only FRB121102 and FRB150807 have been reported to have reliable spectral index measurements), show a large spread. FRB121102 is the most prominent example, where the measured spectral index values for repeated bursts span a range $-10.4 \leq \alpha \leq +13.6$ (Spitler et al., 2016), which can have significant impact on the predicted survey limits and expected event rates.

6.2 Processing Pipeline

The data was processed using a GPU-enabled pipeline, capable of efficiently using multiple GPUs at one. Additionally the data are stored externally on multiple hard drives connected directly to the head cluster node and have to be transferred to the compute nodes before the processing. The hardware configuration is shown in Table 6.2, with 5 identical nodes used for the processing.

6.2.1 Preprocessing and Distribution on the Cluster

A number of steps have to be taken in order to prepare the data before a single-pulse or a pulsar search can be run. These include downloading the raw voltages into the compute node, creating a filterbank file and performing the zero-DM filtering. Combined together, these steps can take up to 2 h to complete when processing the largest of the files and only after that time can the pulsar and single pulse searches begin. It is therefore important to efficiently use and share the available resources and keep various processing stages idle for the shortest periods of time possible and process the data concurrently.

Two approaches were considered for the data processing which minimise the time spent on preparing the data. They both rely on splitting the preprocessing and single-pulse/pulsar search phase and executing them concurrently. The first considered solution would allow for the full utilisation of available GPU resources with 4 files processed at the same time and another file being prepared for the processing. If this processing mode was to be used, with the largest files being 40 GB in size, we would

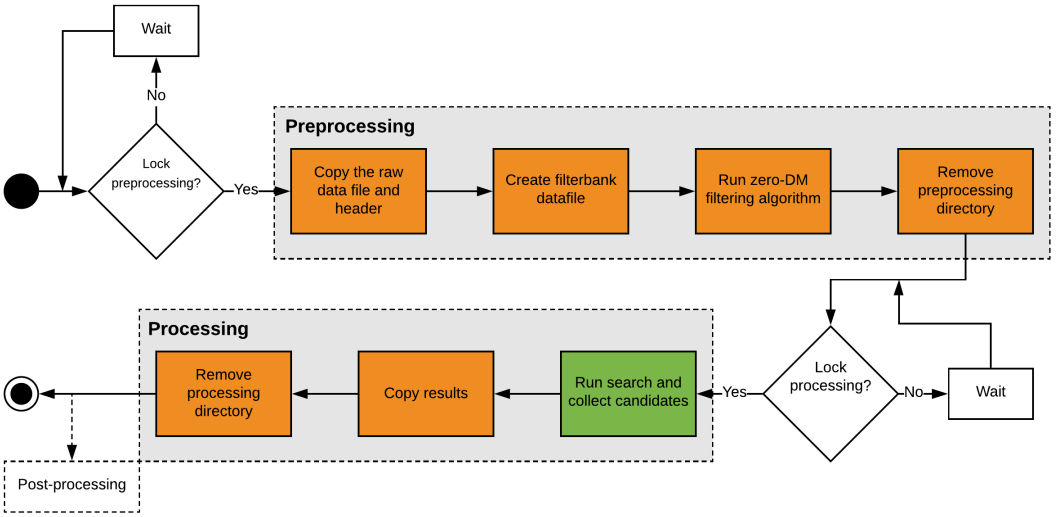


Figure 6.3: GHRSS pipeline workflow for a single job submitted to the node. In orange are the parts of the pipeline run solely or mainly on the CPU, while the green parts run primarily on the GPU. The post-processing phase where the data is combined and various plots are generated is optional.

then require at least 160 GB when processing the data and another 40–80 GB for a file in the preparation stage, meaning a machine with 256 GB of RAM would be a necessity. Having only 128 GB available per node means we are limited in the number of operations that can be run in parallel. We have therefore chosen to use a scaled-down approach and work on only two files at the same time: one being prepared for the processing and one being searched for radio transients. In the search stage, all of the GPUs do as much work as possible on a single file, which in our case means splitting the dedispersion, which is the most time consuming part, across all 4 GPUs. The single-pulse search part however still uses a single-GPU implementation, with the remaining 3 kept idle.

The pipeline structure currently in use is shown in Figure 6.3. Splitting the processing into two separate phases means we have to make sure only a single instance of each stage is being executed at the same time on every node. With one file being copied at a time, other jobs are required to wait until the file preparation phase is over. The same is true for file that is being processed - we can only run the search using the data from one pointing at a time, and therefore preprocessed files have to wait for their turn.

The main skeleton of the pipeline is written in Bash to ease the development and ensure compatibility between different Linux workstations, with jobs submitted to nodes

through a job scheduler. Unlike many programming languages, such as Python or C++, Bash lacks explicit support for mutex/semaphore behaviour in its most basic from and additional features were not available to us in our node configuration. We have therefore selected an approach that can simulate the desired locking behaviour, protecting different pipeline processing stages. As our requirements are not too stringent, i.e. we do not require the level of protection that multi-threaded applications usually do, the intended semaphore-like behaviour is achieved by creating a directory, which serves as a mutex, as shown in Listing 6.1, as the `mkdir` is an atomic operation. After the directory is created, if another job tries to create it again, the operation is unsuccessful and the job is required to wait for a given amount of time, which in this case was chosen to be 2 minutes.

When the job is in the preparation stage, the necessary raw data files are copied and converted to the SIGPROC filterbank file format and the zero-DM filtering is applied, which is used to remove low-DM RFI contamination by calculating the mean of all the frequency channels for every time sample and later subtracting this mean from every channel for that time sample (Eatough et al., 2009). When completed, all temporary files are removed, leaving only the final filterbank product, reducing the disk space usage, which is especially important when multiple files are waiting for processing (with the available disk space, we can have up to 10 prepared filterbank files maximum waiting). The lock directory is then removed, enabling another job to move to the file preparation stage when woken up.

The same approach has been used for the files waiting after the preprocessing stage. In this case, the job that is running the single-pulse search is responsible for creating the locking directory and removing it after the processing has been finished. The sleeping processes do not use any CPU resources, so it is therefore safe to have more than one. Using this simplified solution and not fully-fledged semaphore means that we cannot guarantee the order of completion, i.e. whichever job wakes up and creates the directory first can proceed further, no matter its position in the job scheduler queue. We also do not wake the sleeping processes up if the processing is finished before the 2 minute long sleep period is over. This can lead to a situation where the files which have been sent to the processing at the beginning of the queue can be processed further down the line or in the most extreme cases at the very end. This mechanism also means we have to be especially careful when the jobs are cancelled for whatever reason, be it erroneous code execution or problems with the cluster hardware and software. If the job is terminated before it manages to remove the blocking directory, it then has

to be removed manually from each node, otherwise all future jobs will not be able to progress beyond that block and will wait until cancelled.

Listing 6.1: A simple semaphore-like mechanism for locking the access to different parts of the pipeline.

```
1    while ! mkdir lock_file_prep &>> lock.log
2    do
3    sleep 1m
4    done
```

Even though we are not capable of processing 4 files at the same time and fully utilising the available GPU resources, there is still a benefit of computational time reduction as GPUs do not stay fully inactive when the file is being prepared. The only time when some GPUs do not do any work is during single pulse searches, which leads to underutilisation of GPUs for a short period of time. The amount of time required for the processing of one file lies between 30 to 60 minutes on average, depending on the file size, which, including file preparation stage, gives between 1 and 2 h total. Overlapping the preprocessing and processing stages means we can effectively hide the time 'wasted' on preparing the data and receive the final data products every hour or so, instead of every 2 h in the worst case scenario if we employed a fully-sequential execution.

6.3 Bifrost Code Overview

The Bifrost code combines the separate pulsar and single-pulse detection pipelines into single one, which helps us to streamline the search methodology. Using Heimdall for single-pulse and peasoup[7] for pulsar search, using the standard FFT approach, as described in Section 2.2.2, allows us to combine the common execution steps, as both of these pipelines make use of the same input data products and use the same libraries and algorithms for some parts of the execution, e.g. dedispersion. They where also both designed to make use of GPUs, which means we were able to reduce to number of intermediate data products and expensive device↔host copies. Both codebases have also been modified with the GHRSS survey in mind to better fit our needs and reduce the code complexity required if the pipeline had to be flexible enough to be run with a number of different surveys. Bifrost is also a stage that is marked green in Figure 6.3

[7] https://github.com/ewanbarr/peasoup

- with the most critical parts run on the GPU and responsible for producing the FRB and pulsar candidates.

6.3.1 Dedispersion

The theoretical framework for the dedispersion implementation has been discussed in Section 2.2.1. Bifrost processing uses the dedisp[8] library, which is a fast GPU implementation of the brute force dedispersion algorithm. As mentioned previously, it enables splitting of the processing and running the dedispersion concurrently across multiple GPUs and we can therefore use all our available GPUs to significantly reduce the processing time of this computationally intensive part of the pipeline. For every file we run the dedispersion over more than 3000 DM trials between 0 and 2000 pc cm^{-3}. 2000 pc cm^{-3} is also the upper limit for the pipeline dedispersion stage, enforced by the GPU memory constraints. As shown in Figure 1.3, the dispersion delay is much more prominent at the centre frequencies used in the GHRSS survey than, for example, any survey run using Parkes radio telescope, operating close to 1400 MHz. The dedispersion library used as a part of the pipeline makes certain assumptions about the length of the data chunk necessary to fully dedisperse the signal. The dedispersion is then performed with the help of the texture memory which, for the architecture used in the GPUs that were installed in the cluster, imposed a data size limit which did not allow us to dedisperse signals using higher DM values unless a significant time averaging is introduced.

During the initial testing, dedispersion was found to be the most crucial step of the processing, where all the candidate information can be lost if the final dedispersed time series is not processed properly. The input filterbank files for the GHRSS survey are stored with 8 bits per sample and the same value was also chosen for the dedispersed time series output, stored as an unsigned integer, to minimise the memory requirements and reduce the time spent on copying the data. This means that the output data had to be scaled down. We use the same approach as described in Section 3.2.4 to fit the dedispersed time series into an 8 bit number, but this time with the final standard deviation set to 32.

The original approach was to again calculate the mean and standard deviation of the first few thousand time samples only, and use the same values for the scaling of all the other time samples, the same approach used during the PAF processing. This

[8] https://github.com/ewanbarr/dedisp

approach heavily relied on the assumption that the mean and standard deviation did not deviate significantly from their initial values during the observation. This was found not to be the case for the majority of the observations as the dynamic RFI environment, changing rapidly over a period as short as few seconds, has a large effect on the quality and levels of the data and did not improve considerably with the application of the zero-DM filtering. If we measured the mean and standard deviation using a 'clean' portion of the observation only, it would generally result in overflows, truncated to the maximum value of 255, in parts with excessive RFI. If, however, the opposite took place - with strong RFI at the start of the observation, the pipeline would return the data with a large number of underflows set to 0 in the clean parts as we were subtracting mean that was too high.

An ideal solution would be to clean all the files by hand, but our main goal was to find a viable algorithmic solution that could be applied not only to the data that we have already collected, but also for future observations performed as part of the GHRSS survey which is still underway. The rapidly changing characteristics of the RFI also meant that we were not able to suppress it using standard techniques to a level allowing for smooth RFI reduction and more sophisticated algorithms were not available to us at the time of processing. A different approach was therefore designed in which the data is split into a number of chunks with the mean and standard deviation calculated for each chunk separately and the correct values used during the scaling procedure. Each chunk has the same size as the portion of the dedispersed time series being processed during the single-pulse search. This size is small enough that large fluctuations caused by bursts of RFI are not fully flattened out and large enough so that if any signal of interest was present in the data, it would not significantly alter the calculated values. Figure 6.4 shows an example for one of the observations, where the mean and standard deviation are indeed relatively constant for the first 300 seconds of observations, but both sharply increase afterwards and continue to fluctuate until the end of the data - the change that would result in corrupted output if not taken into account properly when using only the start of the observation to estimate the signal levels and its variations.

The obtained values of mean and standard deviation cannot be used directly during scaling. Rapid changes of large magnitude should generally be avoided, as they can introduce additional data corruption, especially when two neighbouring dedispersed time chunks, scaled using different parameters are examined for the presence of single pulses. Both values have to be smoothed before they can be used. Figure 6.4 also shows the comparison between the unsmoothed and smoothed standard deviation values. In

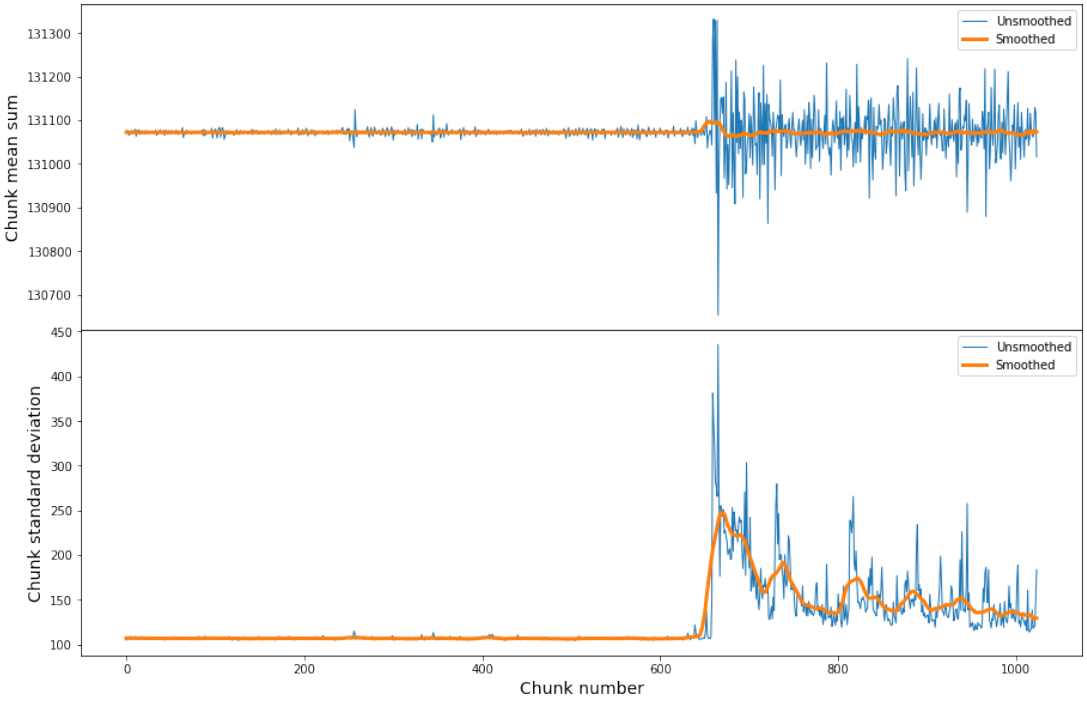


Figure 6.4: Mean sum and sum standard deviation for 1024 dedispersed time series chunks for one observation. The mean sum and standard deviation remain relatively constant at the start of the observation due to the zero-DM filtering applied before the processing.

this case, the smoothing has been done by using a simple moving average of width 32. After the smoothing, rapid fluctuations have been removed and the sharp increase of standard deviation, which originally occurred over one chunk, has been spread over the neighbouring chunks.

This process can be computationally expensive if applied to each dispersion measure trial separately. Instead, it was chosen to be run using a single representative DM value and apply the same scaling to all of the dedispersed time series for a given pointing. As the application of zero-DM algorithm changes the statistics of the signal and in most cases we are interested in finding signals at high dispersion measures, it was important to use a higher dispersion measure that would result in the best distribution of the final values across all DM trials. We have run a number of tests using a selection of observations which showed various degrees of RFI affecting the data quality. As can be seen in Figure 6.5, the dedispersed time series at a DM of $0\,\mathrm{pc\,cm^{-3}}$ experiences more rapid and greater variations, even after the application of the algorithm meant to reduce the amount of zero-DM RFI. Even though the mean has a similar level at all DM values, the standard deviation is much greater at a DM of $0\,\mathrm{pc\,cm^{-3}}$ than at all of

the other DMs. This means that if we were to use these values, we would overestimate the variations of our signal levels, which would result in an incorrect scaling and the distribution of the output values would not fit into the desired range.

This effect can be clearly seen in Figure 6.6, showing the results of the analysis for one of the pointings. Even though using μ and σ obtained at the lowest dispersion measure results in all the values being within the desired range, large signal variance and therefore standard deviation means the resulting distribution is highly concentrated around the mean value, with all of the differences in the dedispersed time series smoothed out, which can result in a fully removed signal of interest. Using the mean and standard deviation obtained from dedispersing the data to different dispersion measures results in a more spread out and correct distribution for all but the lowest DM. In this case we have a very large spread and only around 35% of all the values are found in the desired range. This behaviour however disappears relatively fast and the distribution returns to normal at a DM of $\sim 50\,\mathrm{pc\,cm^{-3}}$, leaving the range of the DM values of interest for FRB surveys properly dedispersed. Out of all the DM values and tests run over the subset of 15 pointings, showing a varying degrees of RFI contamination, $\sim 1000\,\mathrm{pc\,cm^{-3}}$ was marginally better than all other large DM values, and this is the value we use for all of our observations.

6.3.2 Pulsar Search

The Bifrost pulsar detection stage follows the same scheme as described in Section 2.2.2. The vast majority of the pipeline was kept the same as in the original codebase. Small changes were introduced, mainly to allow the testing of different GPUs in different configurations and include the option to choose which GPUs the pipeline is meant to run on, not just their number, and additional timing and logging code. The file reading code has been changed to include time averaging, in order to decrease the total number of time samples and reduce the amount of the GPU memory used during the processing. This halves the amount of RAM used to store the original data and also allows us to reduce the computational time. The pulsar search was initially run for DMs and accelerations in the range 0–$2000\,\mathrm{pc\,cm^{-3}}$ and -250 to $+250\,\mathrm{m\,s^{-2}}$ respectively, for candidates with S/N grater than 5.0.

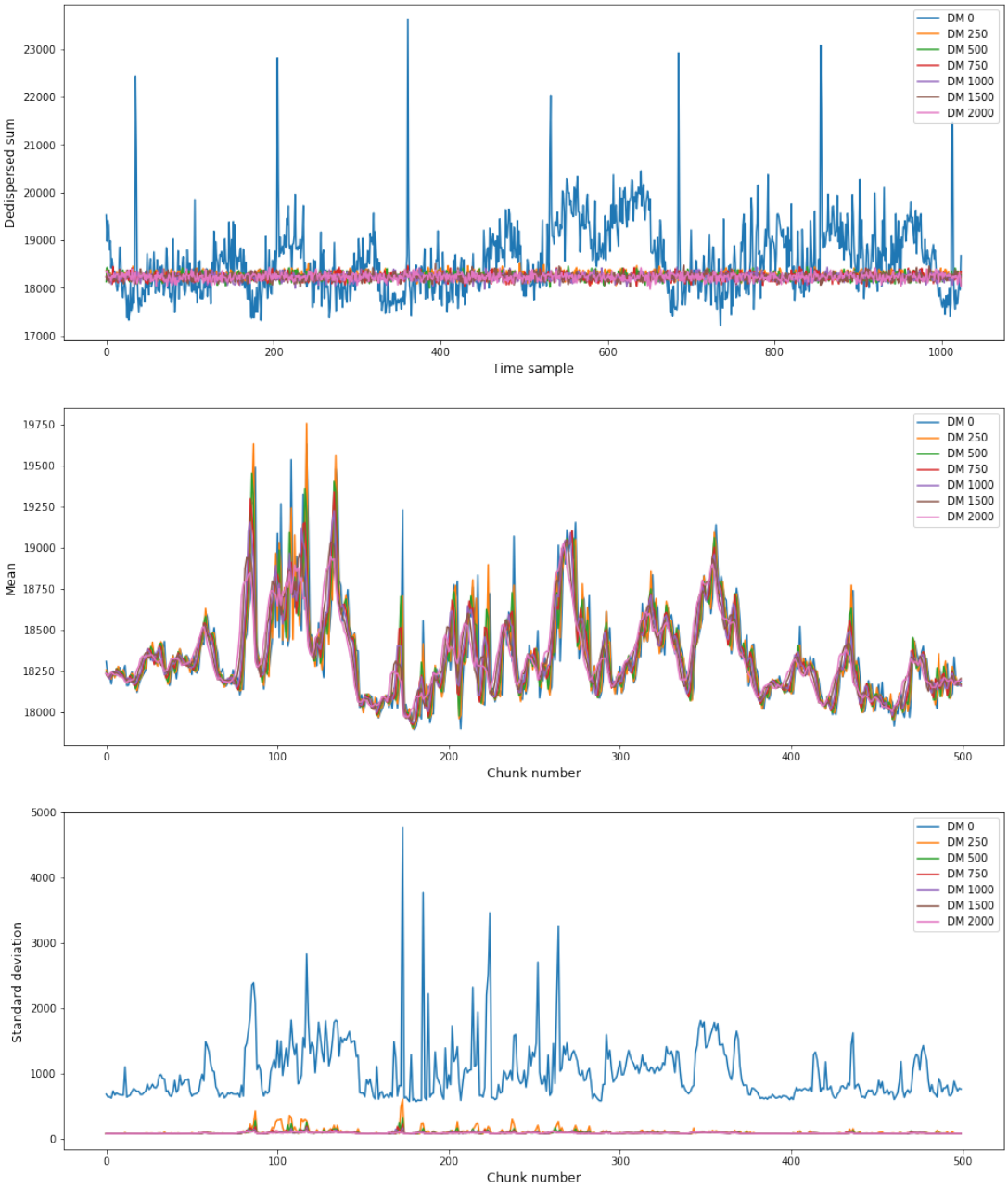


Figure 6.5: Comparison of properties of dedispersed time series using different DM values. Top: the sum of first 1024 fully-dedispersed time samples. Middle: mean dedispersed sum within a time chunk with 16384 time samples. Bottom: standard deviation of the sum within the time chunk. The signal dedispersed to $\sim 0\,\mathrm{pc\,cm^{-3}}$ does not have the same properties as when using other DM values, which is most evident in the differences in the dedispersed time series, which in turn results in a much larger standard deviation.

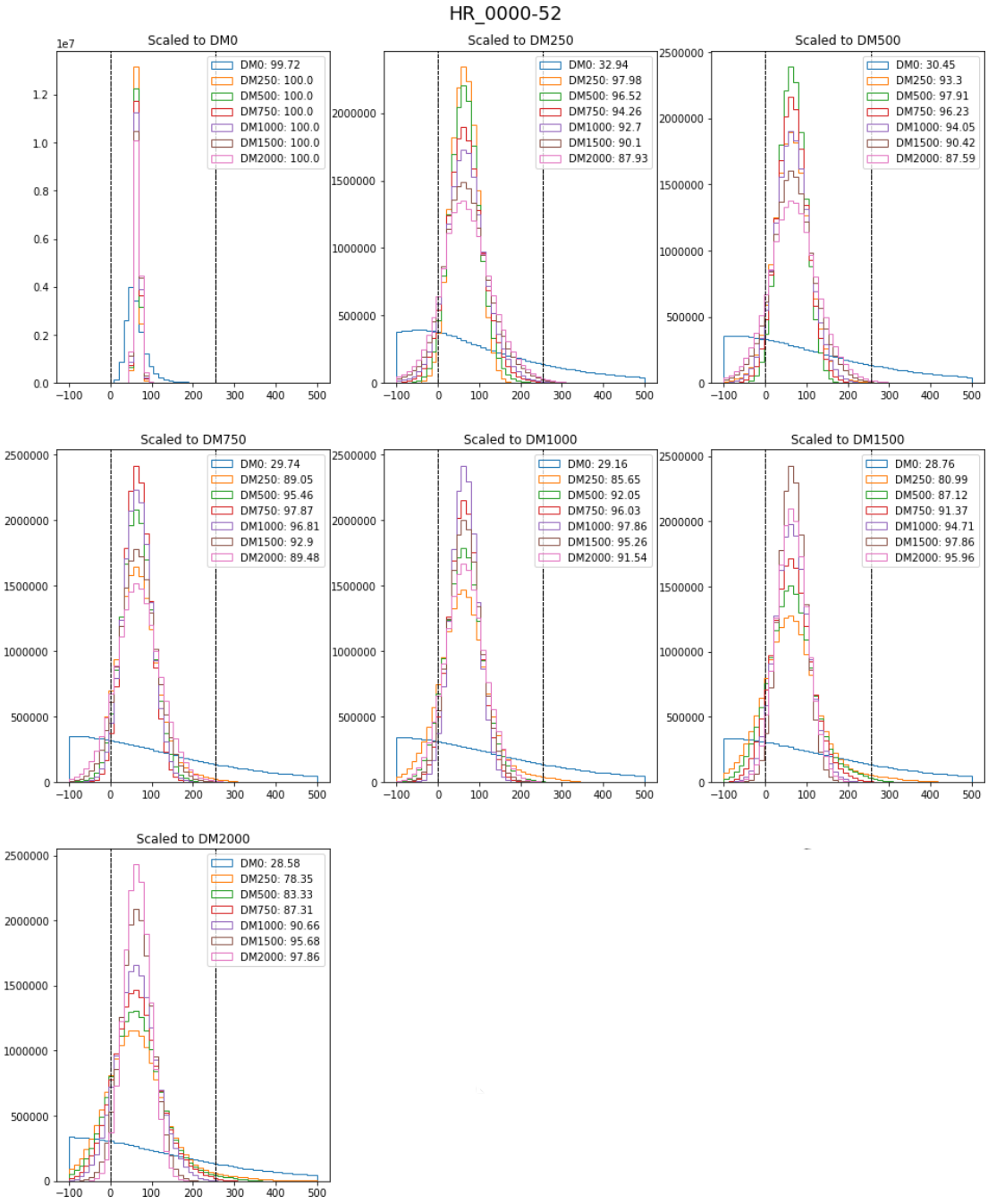


Figure 6.6: Scaled dedispersed time series values distribution with the mean and standard deviation obtained at different DM values. Each subplot corresponds to a different DM value at which the μ and σ have been obtained. Different histograms within the subplot show the resulting distribution after the scaling has been applied to data dedispersed to a given DM. The values next to each legend entry show the percentage of datapoints lying within the 0–255 range.

6.3.2.1 Post-processing

A vast number of candidates had to be examined to reject the false-positives and further process any promising candidates. To achieve this in a reasonable amount of time, an automated post-processing stage was introduced. First, a script is used to compile a table of necessary information, such as candidate ID, period, DM, acceleration, for each candidate detected during a given observation. It is then used to create candidate files, one file per candidate, which are later fed into the software responsible for folding the data using the provided candidate parameters. A number of candidates can be folded at the same time, with resulting output files saved in separate directories. These output files contain 10-second long sub-integrations which have to be combined to cover the entire period of observations before further analysis. The resulting archives are scrunched in the frequency dimension into groups of 64 channels. The main advantage is the considerable reduction of the data size, which results in a shorter computational time and lower total disk space usage. The obvious disadvantage is the loss of finer frequency information and any possible insight into pulse profile changes with frequency. However, as the main focus at this stage is to confirm any possible candidates, this resolution loss can be accepted and the non-scrunched data can be used during further examination, after the pulsar has been confirmed. The post-processing PostScript files used for quick visual inspection of the most promising candidates with the highest SNR are generated at the end from the frequency scrunched archives.

Figure 6.7 shows two folded profiles obtained using the post-processing pipeline, each reported as a high-S/N Bifrost pulsar candidate. It can be clearly seen that only one set of parameters results in a clearly visible pulse. Surprisingly, the candidate that cannot be seen has the higher S/N out of the two as indicated by the pulsar search code. This shows that signal-to-noise ratio cannot be used as the only deciding factor in the candidate selection and the additional analysis described above is indeed required. The candidate with a clearly visible pulse has a period of 0.4548 s and DM of $49.3\,\mathrm{pc\,cm^{-3}}$, which is consistent with the reported values for $\mathrm{B1510}-48$.

6.3.3 Single-pulse Search

The single-pulse search is executed after the pulsar detection, if both of them are requested. The original Heimdall code has therefore been modified to make use of the already dedispersed time series, instead of unnecessarily reading the original filterbank file and dedispersing it again. Other small modifications included optimising the code

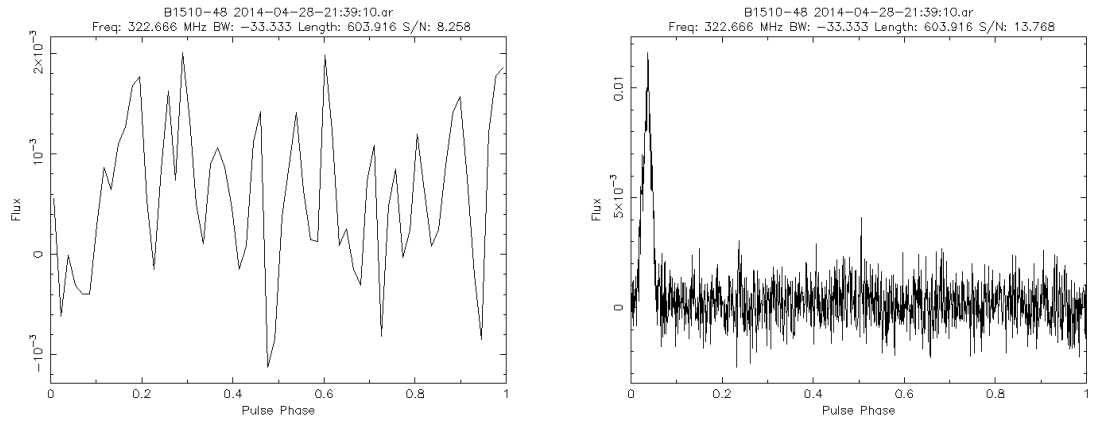


Figure 6.7: Comparison of folded profiles for two pulsar candidates: one with a S/N of 48 (left) and the other with a S/N of 38 (right). Even though the pulsar present in the field of view has been rediscovered with a relatively high signal-to-noise ratio, this shows that some candidates with a higher S/N can clearly be false positives. Different pulse resolutions are the result of different pulse periods, with the plot spanning the entire pulsar rotations, and the false-positive having much shorter period that the true detected pulsar radiation.

for the files with 8 bits per sample, as used in the GHRSS data files, resulting in a shorter and less complex code, making it easier to track and fix bugs.

The dedispersed time series is processed in chunks corresponding to roughly 24 s of observation. Some files contain a large number of false candidates, which can be caused by RFI, problems with dedispersed time series scaling or problems with the telescope hardware. The candidate detection stage is much shorter than the candidate grouping one and with a reasonable number of candidates, they both take less than a minute per chunk to complete. However, this time can increase to up to 10 minutes per chunk when the number of candidates exceeds tens of thousands. A threshold of 250000 on the acceptable number of candidates per minute is therefore used. If it is exceeded, the processing is halted and no candidates are printed out for a given data chunk.

6.4 Results

During the initial commissioning, we used both the pulsar and single-pulse search paths of the pipeline. It was later decided to use a different method for pulsar searches and the Bifrost pipeline was moved to fully single-pulse operations. Here we present the results of this processing, with the main focus on FRB searches and the results from the single-pulse simulations described in Section 5.

6.4.1 Known Sources Detections

The pulsar search pipeline has been used to process the first round of observations, which included around 50 pointings. We were able to confirm the ability to discover pulsars by detecting signals from known objects. Figure 6.8 shows two redetections from the dataset of known pulsars processed with the Bifrost pipeline. The top panel shows the redetection of J1514 – 4834, an ordinary pulsar with a period of 0.4584 s and a DM of 51.5 pc cm^{-3} (Newton et al., 1981). It was detected by the pipeline with a S/N of 32.2, a period of 0.4548 s and a DM of 49.70 pc cm^{-3}. The bottom panel shows the redetection of one of the 10 GHRSS survey pulsars discovered so far (Bhattacharyya et al., 2016), first detected using an alternative method, J1559 – 44. It is located next to the previously-know pulsar J1559 – 4438. In fact, most of the candidates were coming from this well known pulsar, with multiple redetections at different periods, indicating a relatively bright pulsar, which was further confirmed by clear single-pulse detections seen in Figure 6.9. The new pulsar was detected after a careful examination of weaker candidates and was found at a S/N of 8.1 (for comparison, the strongest J1559 – 4438 candidate had a S/N of 346.8). Detected by the Bifrost pipeline at a period of 1169.80 ms and a DM of 117.39 pc cm^{-3}, this is consistent with the officially confirmed detection at 1169.80 ms and 122.0 pc cm^{-3} (Bhattacharyya et al., 2016) and has been confirmed to indeed be a different pulsar and not another signal coming from the previously discovered J1559 – 4438.

We have also used known pulsars as a test dataset for the single-pulse detection part of the pipeline. Initially, none of them were found or had large chunks of the candidate output data missing. This was found to be caused by problems with the final dedispersion scaling, as described above. The data was later reprocessed after the improved scaling solution had been implemented. As can be seen in Figure 6.9, we were able to redetect them with a varying degree of success. Some of the pulsars were clearly visible, like J1559 – 4438 in panel c). In the case of J1143 – 5158, only a small number of single pulses was detected, but it can be seen that there are vertical bands of missing data, e.g. a large gap between the 500 and 575 s marks, caused by a large number of candidates and the pipeline stopping the chunk processing (a small gap can also be seen in the J1559 – 4438 detection plot). For other pulsars, such as J1514 – 4834 and J1822 – 4209, we were not able to identify any detections that could be classified as coming from these objects. This was an indication of a problem with our ability to detect single pulses, which could have serious consequences when trying to find FRBs, possibly causing us miss events that were buried in the noisier parts of

the data and therefore not fully processed or simply missed due to their low S/Ns. To better understand our detection capabilities, we injected the simulated single pulses as described in Section 5, which was expected to provide more useful information than the periodic pulsar signals.

6.4.2 Simulation Results

As has been already described, the number of candidates was expected to increase as the RFI gets worse, but after a certain threshold for the number of detections per unit time is reached, the search for a given data chunk will stop and no candidates will be saved. We therefore wanted to better understand the influence this would have on the fraction of the data processed properly and therefore the predicted FRB rates. For this series of tests we have chosen this threshold to again be 250000 candidates per minute, the same as for all of the original processing. Strong RFI was present in most observations. However, its exact nature and time span varied between individual pointings. The negative effect of RFI was sometimes sufficient that the candidate threshold was reached for every chunk and no candidates were returned for the whole observation. Using our simulation software also helped us to better quantify the limits of our pipeline, including the sensitivity to bursts of different widths and at various dispersion measures.

In total, we have injected simulated data into 216 pointings, more than 50% of all the processed data, with 76 pointings coming from earlier observations run between May–September 2014, and the remaining 140 taken between January 2015 and May 2016. The more recent data has been recorded mainly using 2048 instead of 1024 channels, although some 1024 channel data is still present. As the same total bandwidth has been used in both cases, that means the 2048–channel data has an improved frequency resolution at the cost of the time resolution, which is two times lower than for the observations using 1024 channels. All of the datasets have been prepared and processed in the same way, so that all differences can be attributed to the data in the files themselves rather than different processing methodology.

The earlier dataset has been observed to experience substantial RFI contamination, which is reflected in the simulation results. Out of 76 simulated events, we were able to successfully detect only 31, meaning we lost the information on around 59% of possible detections. Out of the non-detected 45 events, 42 had processing halted early due to a high candidate rate and the remaining 3 had no candidate detected, even though the corresponding time chunk was fully processed. Even though the final percentage

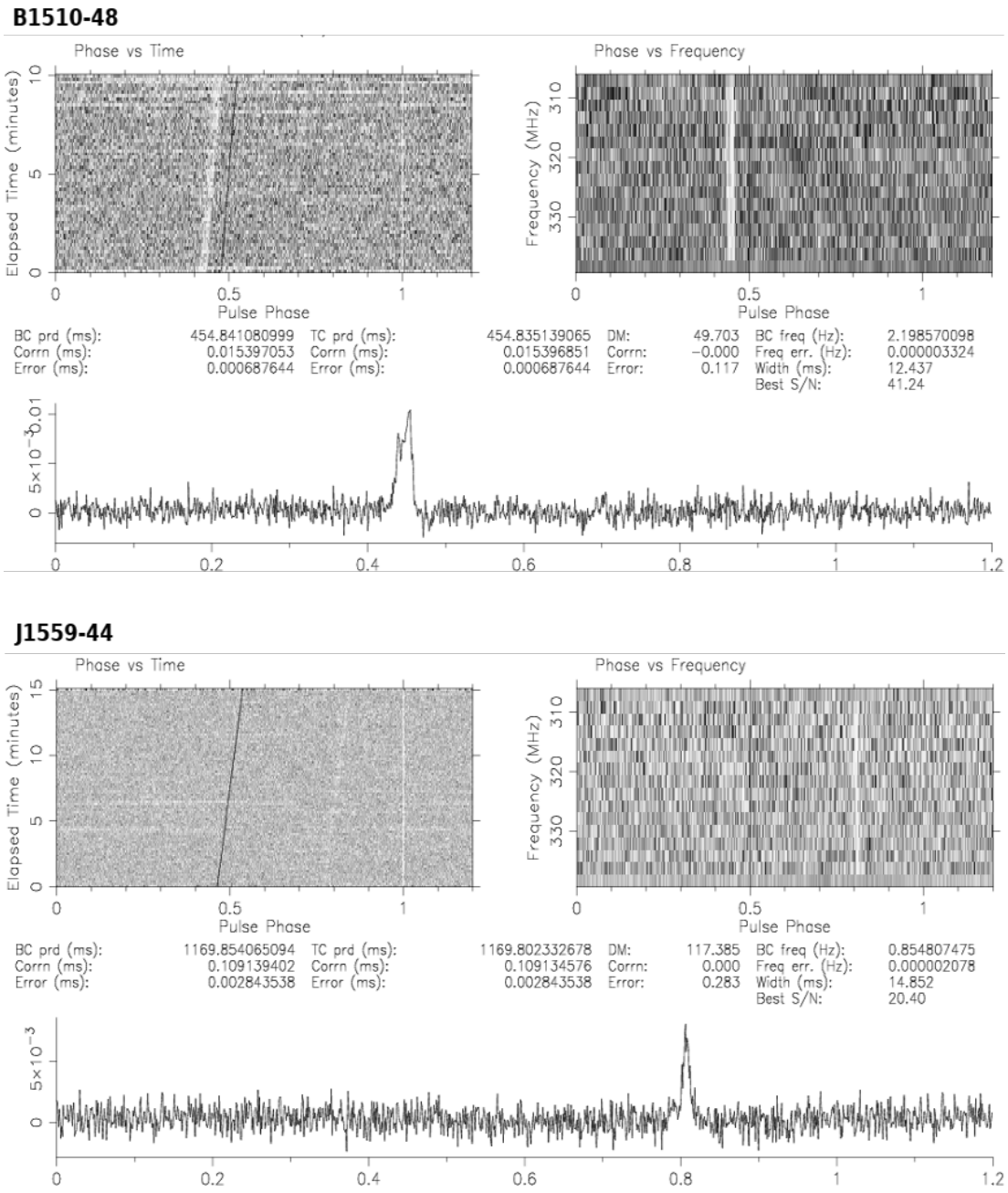


Figure 6.8: J1514 – 4834 and J1559 – 44 redetections obtained using the pulsar search part of the pipeline.

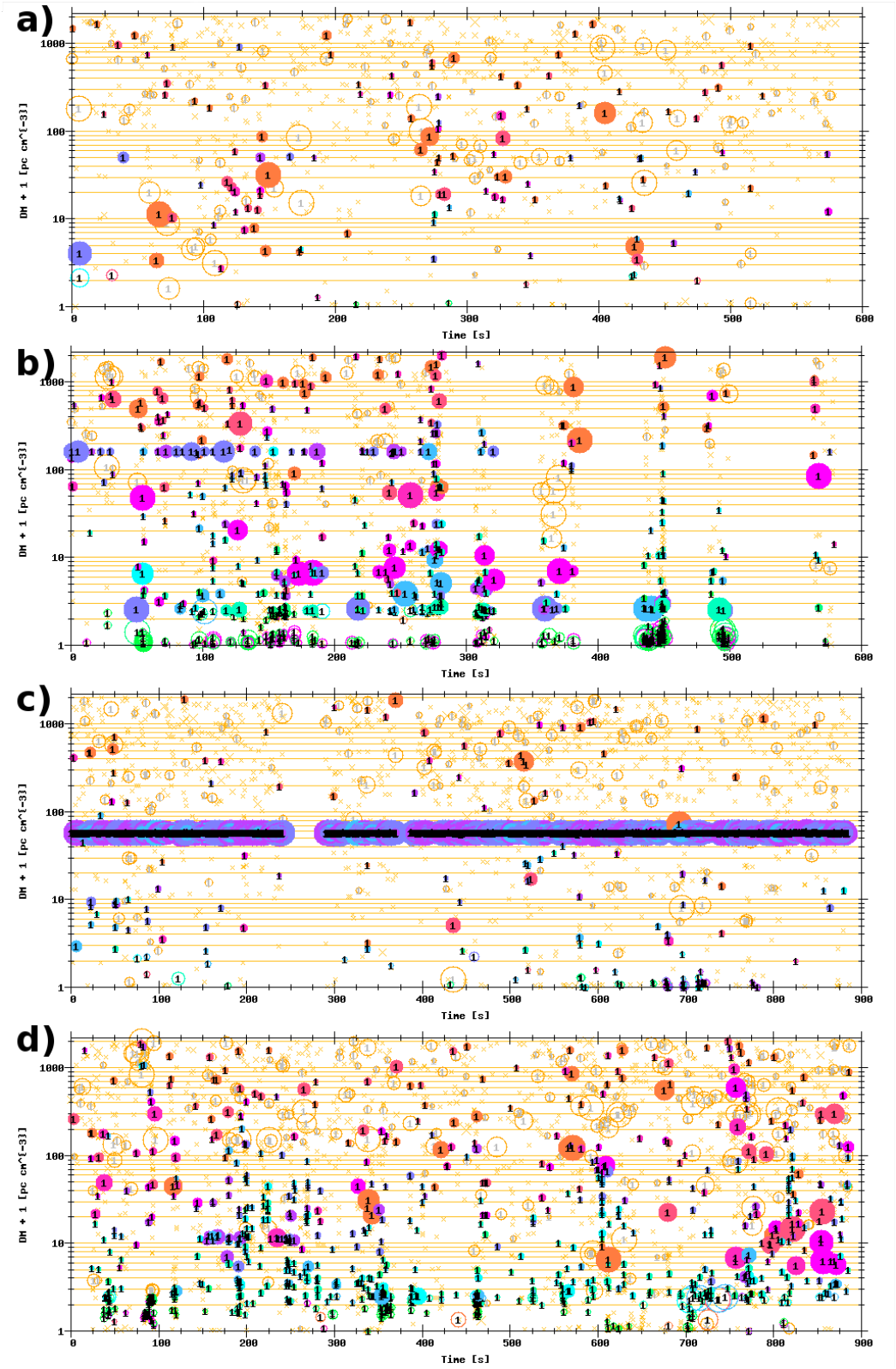


Figure 6.9: Single pulse detection outputs for known pulsars present in the available dataset. From the top: a) J1514 – 4834, b) J1143 – 5158, c) J1559 – 4438, d) J1822 – 4209. J1559 – 4438 is clearly visible in the plot, whereas J1143 – 5158 has only some pulses detected, but a large number of data chunks are missing from the output. Pulses for the two other pulsars were not detected, most probably due to their low S/Ns.

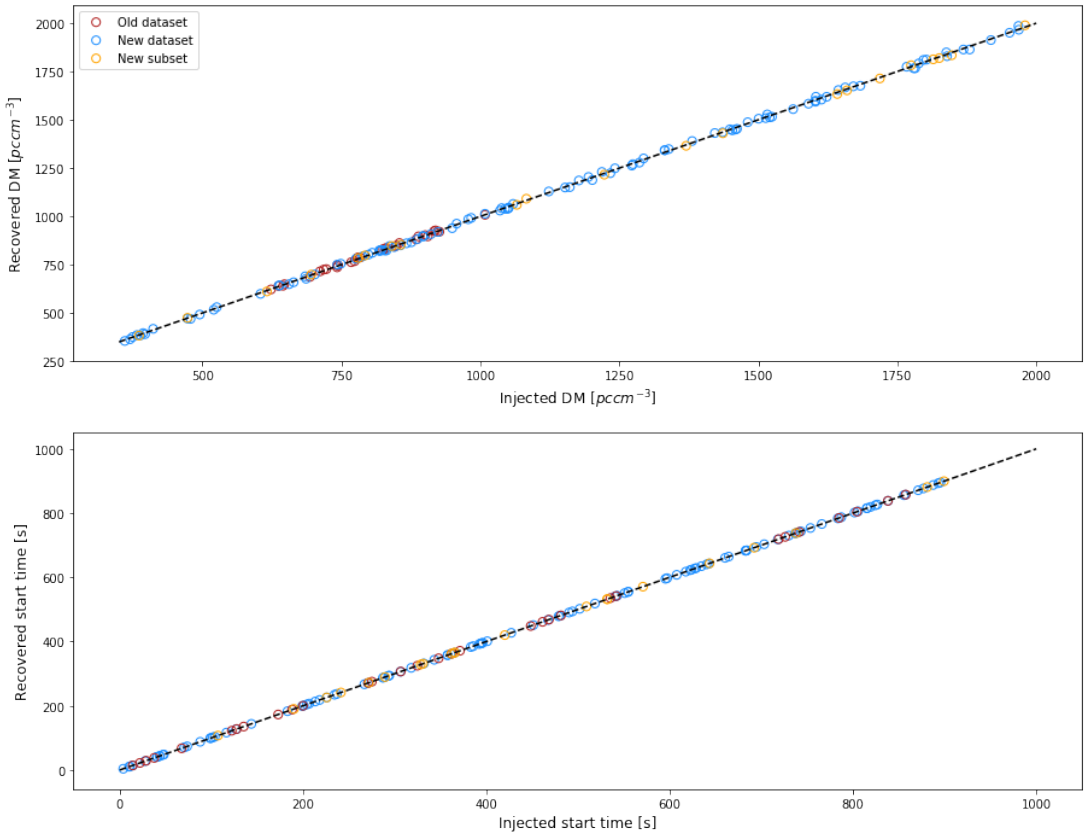


Figure 6.10: Simulated FRBs detected by the pipeline. We generally recover the correct DM and start time of the pulse. Small variations of the recovered DM from the injected values are present, but the 'true' DM is usually obtained during additional post-processing stages.

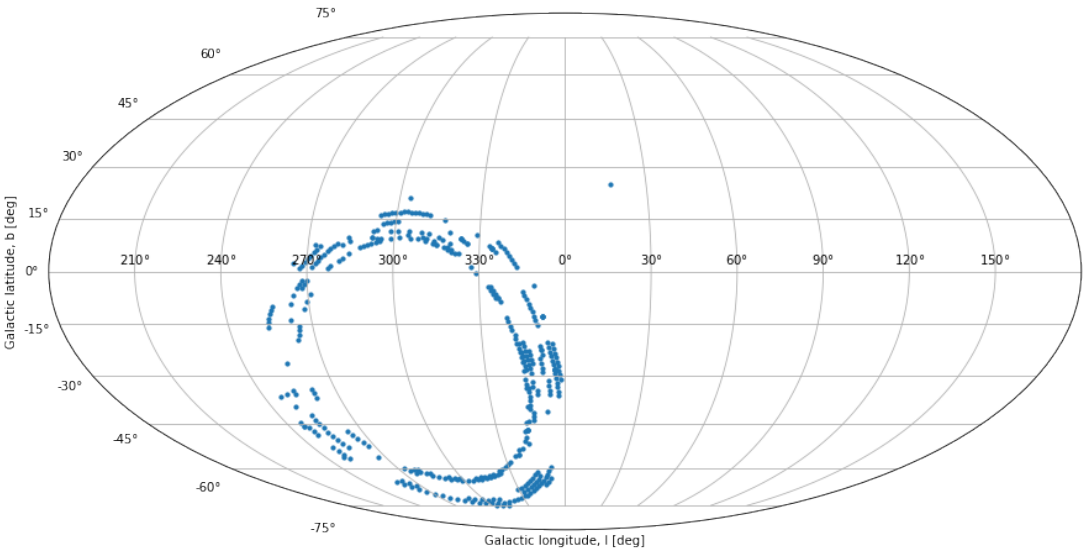


Figure 6.11: GHRSS pointings in Galactic coordinates.

of recovered bursts was low, it represented an improvement over earlier simulations, without proper post-processing applied to the dedispersed time series, where up to 90% of injected candidates were lost. As show in Figure 6.10, single pulses were generated over a small DM region, between 600 and 1000 $\mathrm{pc\,cm^{-3}}$ in this particular dataset. This was chosen to represent the population of known FRBs at the time when this particular simulation was run. It was also restricted due to the fully-coherent mode used at that time which, as described in Section 5.2.2.1, takes a lot of time and uses a lot of resources to disperse bursts at higher DM values. Nevertheless it allowed us to assess our ability to recover bursts and improve it.

The situation improved when the newer dataset was used. In this case, we were able to detect 115 out of 141 simulated single pulses - close to an 82% recovery rate. The new dataset has been found to experience much lower levels of RFI causing dropped processing. This simulation run used a more uniform DM distribution, starting at 350 $\mathrm{pc\,cm^{-3}}$ up to the practical maximum achievable limit with the current pipeline configuration of 2000 $\mathrm{pc\,cm^{-3}}$. As we have detected all of the pulses that have been fully processed, we can conclude that out pipeline is sensitive enough to detect bursts over a wide range of dispersion measures where the majority of known FRBs have been found before and that the increased smearing at large DM values does not negatively impact its performance.

6.4.3 FRB Search Results

In total, we have processed data from 396 pointings, spanning 127 hours, for pointings ranging between 5 and 20 minutes in duration. Figure 6.11 shows their positions on the sky, as projected in the Galactic coordinate system. In this time period we have not detected any events that, after closer examination, could be classified as Fast Radio Bursts. As discussed previously, estimating the expected rates is complicated by the fact that out ability to process the data and therefore detect candidates is severely influenced and hindered by the presence of strong RFI. Our theoretical calculations and simulations have shown that if processed correctly we should be able to detect, even though bright, a large portion of events. This is also confirmed by the single-pulse detections from known pulsars, with some examples shown in Figure 6.9.

The number of expected events is highly dependent on the FRBs' spectral index, which is currently unknown. This causes the estimates of the expected number of events at low frequencies be highly dependent on this value, as the most accurate and up-to-date rates are obtained with surveys operating at 1400 MHz. Using our peak flux density model, as described in Section 6.1.1, we can obtain an estimate of the expected number of events, depending on the spectral index of the FRB population. We consider the limiting peak flux density of 1.2 Jy for bursts with width of 3.9 ms. With each pointing covering an area of 1.8° on the sky, we calculate the survey metric to be 228.6 deg^2 hr. Assuming a flat spectrum with $\alpha = 0.0$, we expect to see a single FRB in our dataset. This however assumes that all of the 127 h have been fully processed. As has been shown in Section 6.4.2, this is not true, as in certain cases we can miss more than 50% of the data. Taking a moderate total data loss across all of the pointings of 25% into account, we do not expect to see any FRBs in the processed dataset and the percentage of the data that has been processed properly is most likely less than 75%. We can therefore conclude that no detections in 127 h of data were expected and more good quality data obtained in the future as part of the GHRSS survey will have to be processed, which will hopefully allow us to obtain better constraints on the FRB rates at 322 MHz.

As the burst rates are highly dependent on the value of the spectral index, negative spectral index values are expected to result in a higher number of FRBs detectable at lower operational frequencies. The positive values considered in our calculation would however result in no detections for bursts above redshifts of 0.2, corresponding to a DM of 240 $\mathrm{pc\,cm^{-3}}$, which is lower than any of the currently reported dispersion measure values for all of the FRBs discovered so far. If FRBs were to follow a power

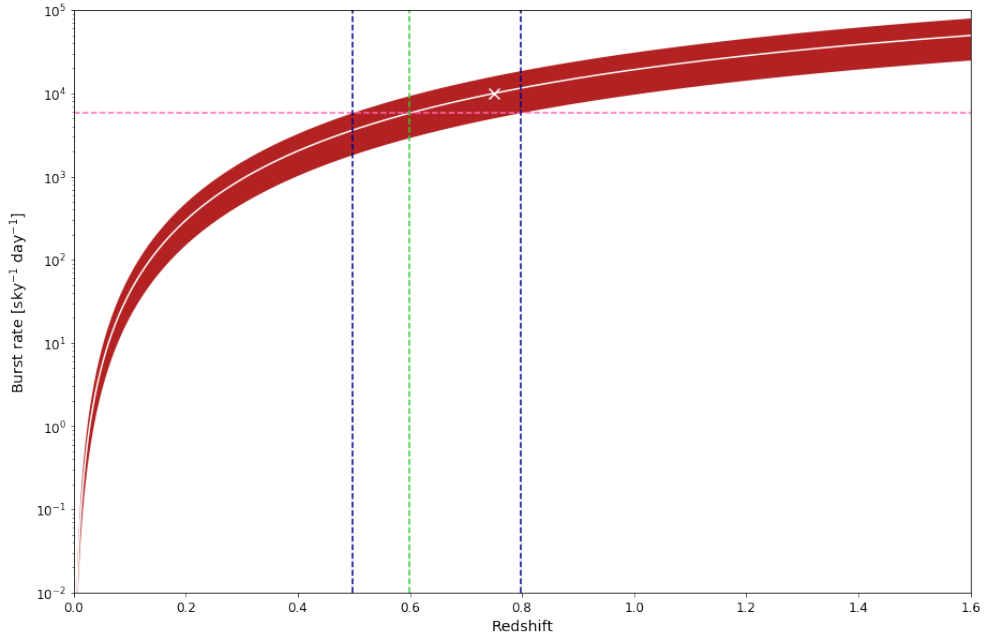


Figure 6.12: Modelled FRB burst rate. Pink line represents the estimated upper limit on the burst rate from the GHRSS survey, the green line is the redshift where our burst rate crosses the modelled burst rate and the blue lines indicate redshift limits when considering errors on the Thornton burst rate. The white cross marks the Thornton rate of 10000 sky^{-1} day^{-1} at the redshift of 0.75.

law with spectral index similar to that of pulsars, at $\alpha = -1.4$, we would expect to observe at least 7 FRBs, with that number increasing to 16 if the spectral index was to have the lowest considered value of -3.0.

To obtain a better estimate on the spectral index, we follow an analysis similar to one performed for the ARTEMIS project (Karastergiou et al., 2015), operating at a frequency of 145 MHz. We can estimate an upper limit on the FRB rate at 322 MHz as 5775 sky^{-1} day^{-1}, assuming 75% of the available data was processed correctly. Using the burst rate scaling with the expected probed maximum redshift (Lorimer et al., 2013), as shown in Figure 6.12, we can estimate the maximum redshift our survey was sensitive to. With the maximum redshift limit we are expected to search to set to 0.59 from the burst rates estimates, we can now proceed to get a better estimate on the FRB spectral index. Using a smaller range of the spectral index values, which cause the peak flux density for go below the limit of 1.2 Jy for bursts with width of 3.9 ms around the limiting redshift, as shown in Figure 6.13 allows us the estimate the minimum spectral index $\alpha = 0.18^{+0.12}_{-0.39}$, which is consistent with the results obtained previously at a frequency of 145 MHz (Karastergiou et al., 2015). With that in mind, we can conclude

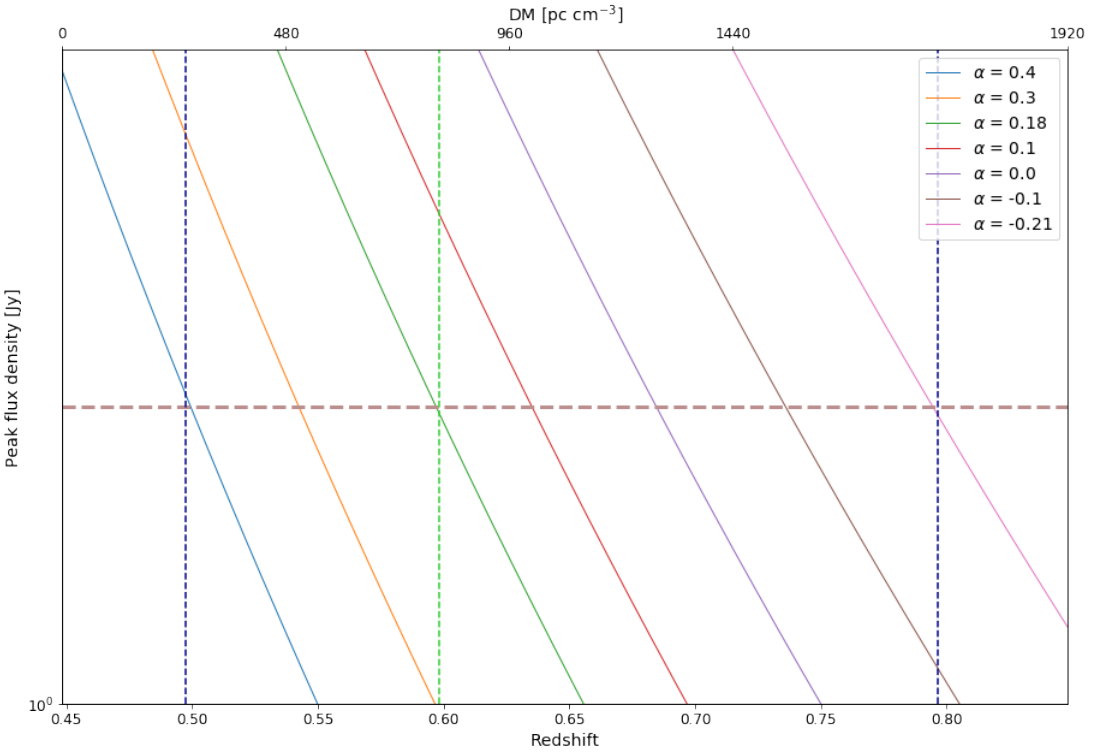


Figure 6.13: Predicted spectral index values assuming the limiting redshifts derived from the expected burst rates. Using the different peak flux curves crossing our limit of 1.2 Jy for bursts with width of 3.9 ms, we can estimate the spectral index of $\alpha = 0.18$.

that in the light of no discoveries in our data, the FRBs are most likely to follow a flat power spectrum or close to it. This result however still depends on the expected burst rates and is expected to change as they are updated thanks to new surveys and that the rate of $10000\,\mathrm{sky}^{-1}\,\mathrm{day}^{-1}$ can currently be considered one of the larger burst rate estimates.

Chapter 7

Conclusions

7.1 New GPU Processing Pipeline for Effelsberg PAF

The PAFINDER single-pulse search pipeline has been successfully commissioned and is now being used for scientific observations with the Effelsberg radio telescope phased array feed. This new software has been developed from scratch and the optimised GPU code allows us to process large volumes of data and create filterbank data products in real time. Initial tests and scientific observations showed our ability to record and detect single pulses from various sources, including multiple detections from RRAT J1819 – 1458 using offline processing. We have also successfully proved the capability to send the output data products to single-pulse search software. However, this part is not currently run in real time when a large number of candidates is present, for example, with a pulsar present in one of the beams. Real-time performance is however met when processing data from an empty patch of sky, which is usually the case when searching for new radio transients. With the use of a DADA buffer as the final product, we have expanded the functionality of the pipeline beyond the initially planned single-pulse search and enabled different processing steps, such as a pulsar search, using software supporting DADA buffers.

Our observations of J1819 – 1458 revealed that this RRAT was undergoing a period of increased bursting, with a burst rate of $54\,\mathrm{h}^{-1}$ measured over a 2 hour long observation period, which is greater than the often reported bursting rate of around $20\,\mathrm{h}^{-1}$. Previously, the increased burst rate has been linked to the presence of glitch in the timing data on a single occasion. We did not find any evidence of a recent glitch after fitting the post-glitch ephemeris. Fitting for timing residuals, however, revealed a complex emission environment, with J1819 – 1458 emitting primarily in three separate

bands with the upper and lower bands separated from the centre emission component by around 0.04 and 0.05 s respectively. Closer examination of all of the 108 single-pulse detections revealed a complex and changing structure of individual bursts. We were able to identify pulses with multiple components, with 39 having two and 5 having three separate emission peaks. We have also found a single grouping of multi-component bursts, with two triple-component and three double-component detections. This groups has been found to be isolated from other detections by more than 68 s and 310 s to the previous and next burst respectively, which is the largest separation for any of the groups identified in our dataset. This could be an evidence for the existence of a period of more violent outbursts that are later followed by less energetic or no emission at all. No clear correlation between the size of groups and their separation from other detections has been found though and we therefore recommend repeating this study with a larger dataset spanning more than 2 hours to identify any possible correlations.

We have also re-observed positions of three FRBs discovered in the past, with one field containing the repeating FRB121102. We were not able to detect any new bursts. With no detections in a continuous 9-hour long observation of FRB121102, this is one of the longest periods in which the repeater has been observed and no bursts have been detected. We use this to estimate the probability of no detections and find compelling evidence that the behaviour of this particular FRB can be better described by the Weibull distribution, which unlike the Poisson distribution, allows for burst clustering and makes the probability of detecting repeating burst dependent on bursts observed previously. When estimated using the Weibull distribution, the probability of not seeing any repeated bursts is almost 5 times greater than if FRB121102 was to follow Poissonian statistics. Even though we were not able to detect any new burst, we have developed new processing techniques to aid future observations and significantly reduce the total number of candidates that have to be examined by hand by removing up to 75% of false-positives caused by RFI.

7.2 Fast Radio Bursts Simulations

We have designed a framework for simulating and injecting Fast Radio Bursts into observational data. This approach can be used with current surveys to better understand the detection limits. It can also be useful when designing and developing new processing pipelines. Designed to be injected as early as possible into the processing stream, it offers the possibility to test the entire detection chain, which at the later stages of

the development provides more useful information than separate tests of individual elements, often using different datasets.

New software has been developed which currently simulates a burst approximated by a Gaussian curve and disperses it either fully-coherently or semi-coherently, depending on the project needs and the available computational resources. It currently supports a small number of input parameters and the propagation effects are limited to dispersion only, and the SIGPROC filterbank is the only file format supported which can have bursts injected into. The modular design, however, means that adding new processing steps is relatively easy and will provide the required level of flexibility to test different emission models and processing setups.

7.3 GHRSS Survey

The Bifrost pipeline has been developed which combines the ability to run single-pulse and pulsar searches in a single codebase, simplifying and speeding up the processing by reusing common steps such as initial time averaging and dedispersion. It has been used as part of the GHRSS survey, with the main focus of discovering new FRBs. We have developed a number of processing techniques that try to mitigate the negative effects of RFI. Our single-pulse simulation software was used to inject the data into more than half of all the pointings processed so far. We have found that in the worst case scenario, we can lose information on more than 50% of candidates due to the detection thresholds being exceeded, caused by the presence of strong RFI. Even though we were able to recover less than 50% of candidate information using one dataset, it represented a significant improvement over more than 90% of the data not being processed correctly when our RFI mitigation techniques were not in place. Using a second, newer dataset, we are able to fully process more than 80% of the available data, indicating a much more stable and significantly improved RFI environment.

After processing 127 hours and a survey metric of $228.6\,\mathrm{deg}^2$ hr with our single-pulse search pipeline, did not detect any new FRBs. This is not an unexpected result, as we expect to observe a single FRB and only when the entire 127 h of data is fully processed, which was shown to not be the case. By treating FRBs as standard candles and considering power laws with different spectral indices, we were able to find evidence for FRBs following a different spectrum to pulsars. If they were to follow a spectrum with α close to -1.4 or less, we would expect to observe around 10 FRBs, and the lack of any detections disfavours values in this range. By considering the upper

limit on the FRB burst rate of $5775\,\mathrm{sky}^{-1}\,\mathrm{day}^{-1}$ obtained for our survey, we are able to obtain an independent estimate of the spectral index of $\alpha = 0.18^{+0.12}_{-0.39}$.

7.4 Future Work

The main challenge for the PAFINDER pipeline is the ability to run single-pulse search software in real time. The current execution procedure does not allow for efficient resource sharing between different processes running on the GPU and alternative methods will have to be investigated. The processing workload can also be reduced by changing the processing parameters, such as the dispersion range, the allowed smearing and the detection threshold. These will have be tested and tuned carefully to ensure the best possible compromise between the computational speed and the achieved sensitivity. An important development will have to involve the investigation of the feasibility of implementing a raw data buffer which will be saved to disk only in the case of a probable single-pulse detection. This can give us access to the polarisation information and will allow us to study any possible new FRBs in greater detail than possible with the current implementation, using a final detected output only. A more robust candidate classification approach can also be implemented, using machine learning that will allow for more effective sifting and identification of true-positives and rejection of false-positives.

The current single-pulse simulation and injection software can be considered a single module in what will eventually become the universal framework for testing the detection capabilities of various telescopes and pipelines. Additional modules are currently being developed that will introduce different propagation and emission effects besides dispersion, such as scattering and spectral index. Support for other file formats, besides SIGPROC filterbank is currently being developed as well, with the CODIF format as the main target, which will make it easier to develop, test and fully benchmark new processing steps for the PAFINDER pipeline without the need to use on-sky telescope time. A new digitisation and injection procedure will also be developed, which will eliminate problems with final bursts having similar reported S/Ns, independently of the amplitude of the original Gaussian envelope.

Bibliography

Abbott B. P., et al., 2017, Phys. Rev. Lett., 119, 161101

Ananthakrishnan S., 2005, International Cosmic Ray Conference, 10, 125

Ananthakrishnan S., Pramesh Rao A., 2002, in Manchanda R., Paul B., eds, Multi-colour Universe.

Applebaum S., 1976, IEEE Transactions on Antennas and Propagation, 24, 585

Baade W., Zwicky F., 1934, Phys. Rev., 46, 76

Baars J., Swenson Jr G., 2007, The Paraboloidal Reflector Antenna in Radio Astronomy and Communication. Springer-Verlag New York

Backer D., Kulkarni S., Heiles C., et al., 1982, Nature, 300, 615

Bailes M., et al., 2017, PASA, 34, e045

Bannister K., Madsen G., 2014, MNRAS, 440, 11

Bannister K. W., et al., 2017, ApJ, 841, L12

Barrau A., Rovelli C., Vidotto F., 2014, Phys. Rev. D, 90

Barsdell B. R., Bailes M., Barnes D. G., Fluke C. J., 2012, MNRAS, 422, 379

Bassa C. G., et al., 2017a, ApJ, 843, L8

Bassa C. G., et al., 2017b, The Astrophysical Journal Letters, 846, L20

Bates S. D., Lorimer D. R., Verbiest J. P. W., 2013, MNRAS, 431, 1352

Berkhuijsen, E. M. Müller, P. 2008, A&A, 490, 179

Bhandari S., et al., 2018, MNRAS, 475, 1427

Bhat N. D. R., Cordes J. M., Camilo F., Nice D. J., Lorimer D. R., 2004, ApJ, 605, 759

Bhattacharyya B., et al., 2016, ApJ, 817, 130

Bhattacharyya B., et al., 2018, MNRAS,

Bingyi Cui M. M., 2016, The RRATalog, http://http://astro.phys.wvu.edu/rratalog/

Brown A. J., et al., 2014, in 2014 International Conference on Electromagnetics in Advanced Applications (ICEAA). pp 268–271, doi:10.1109/ICEAA.2014.6903860

Burke-Spolaor S., Bailes M., Ekers M., et al., 2011, ApJ, 727

Callister T., Kanner J., Weinstein A., 2016, ApJ, 825, L12

Camelio G., Gualtieri L., Pons J. A., Ferrari V., 2016, Phys. Rev. D, 94, 024008

Cameron A. D., Barr E. D., Champion D. J., Kramer M., Zhu W. W., 2017, MNRAS, 468, 1994

Cerdá-Durán P., Elias-Rosa N., 2018, preprint, (arXiv:1806.07267)

Champion D. J., et al., 2016, MNRAS, 460, L30

Chan T. F., Golub G. H., LeVeque R. J., 1982, in Caussinus H., Ettinger P., Tomassone R., eds, COMPSTAT 1982 5th Symposium held at Toulouse 1982. Physica-Verlag HD, Heidelberg, pp 30–41

Chatterjee S., et al., 2017, Nature, 541, 58

Chippendale A. P., et al., 2015, in 2015 International Conference on Electromagnetics in Advanced Applications (ICEAA), p. 541-544. pp 541–544 (arXiv:1509.00544), doi:10.1109/ICEAA.2015.7297174

Chippendale A. P., Beresford R. J., Deng X., Leach M., Reynolds J. E., Kramer M., Tzioumis T., 2016, in 2016 International Conference on Electromagnetics in Advanced Applications (ICEAA), Cairns, QLD, p. 909-912. pp 909–912 (arXiv:1606.03533), doi:10.1109/ICEAA.2016.7731550

Cooley J. W., Tukey J. W., 1965, Mathematics of Computation, 19, 297

Cordes J., Goldshith P., 2001

Cordes J. M., Lazio T. J. W., 2002, ArXiv Astrophysics e-prints,

Cordes J. M., et al., 2006, ApJ, 637, 446

Dennison B., 2014, MNRAS, 443, 11

Eatough R., Keane E., Lyne A., 2009, MNRAS, 395, 410

Falcke H., Rezzolla L., 2014, A&A, 562

Fisher J. R., et al., 2009, in astro2010: The Astronomy and Astrophysics Decadal Survey.

Gaensler B. M., Madsen G. J., Chatterjee S., Mao S. A., 2008, Publications of the Astronomical Society of Australia, 25, 184–200

Geng J. J., Huang Y. F., 2015, The Astrophysical Journal, 809, 24

Gold T., 1968, Nature, 218, 731

Gómez G. C., Benjamin R. A., Cox D. P., 2001, The Astronomical Journal, 122, 908

Hankins T. H., Rickett B. J., 1975, in Alder B., Fernbach S., Rotenberg M., eds, Vol. 14, Methods in Computational Physics. Volume 14 - Radio astronomy. pp 55–129

Hankins T. H., Kern J. S., Weatherall J. C., Eilek J. A., 2003, Nature, 422, 141

Hansen B., Lyutikov M., 2001, MNRAS, 322, 695

Hessels J. W. T., Ransom S. M., Stairs I. H., Freire P. C. C., Kaspi V. M., Camilo F., 2006, Science, 311, 1901

Hu H. D., Esamdin A., Yuan J. P., Liu Z. Y., Xu R. X., Li J., Tao G. C., Wang N., 2011, A&A, 530, A67

Hulse R. A., Taylor J. H., 1975, ApJ, 195, L51

Jeffs B. D., Warnick K. F., Landon J., Waldron J., Jones D., Fisher J. R., Norrod R. D., 2008, IEEE Journal of Selected Topics in Signal Processing, 2, 635

Johnston S., et al., 2007, PASA, 24, 174

Karastergiou A., et al., 2015, MNRAS, 452, 1254

Keane E. F., 2010, PhD thesis, University of Manchester, doi:10.1007/978-3-642-19627-0

Keane E. F., 2016, Monthly Notices of the Royal Astronomical Society, 459, 1360

Keane E. F., Ludovici D. A., Eatough R. P., Kramer M., Lyne A. G., McLaughlin M. A., Stappers B. W., 2010, Monthly Notices of the Royal Astronomical Society, 401, 1057

Keane E. F., Kramer M., Lyne A. G., Stappers B. W., McLaughlin M. A., 2011, Monthly Notices of the Royal Astronomical Society, 415, 3065

Keane E., Stappers B., Kramer M., Lyne A., 2012, MNRAS, 425, 71

Keane E. F., et al., 2018, MNRAS, 473, 116

Keith M. J., et al., 2010, MNRAS, 409, 619

Kirk D., mei Hwu W., 2010, Programming Massively Parallel Processors. A Hands-on Approach. Morgan Kaufmann, Burlington, USA

Landon J., et al., 2010, AJ, 139, 1154

Law C. J., et al., 2017, ApJ, 850, 76

Liu L., Grainge K., 2017, in 2017 XXXIInd General Assembly and Scientific Symposium of the International Union of Radio Science (URSI GASS). pp 1–4, doi:10.23919/URSIGASS.2017.8105014

Lorimer D. R., Kramer M., 2004, Handbook of Pulsar Astronomy

Lorimer D. R., Yates J. A., Lyne A. G., Gould D. M., 1995, MNRAS, 273, 411

Lorimer D. R., Bailes M., McLaughlin M. A., Narkevic D. J., Crawford F., 2007, Science, 318, 777

Lorimer D. R., Karastergiou A., McLaughlin M. A., Johnston S., 2013, Monthly Notices of the Royal Astronomical Society: Letters, 436, L5

Lyne A. G., McLaughlin M. A., Keane E. F., Kramer M., Espinoza C. M., Stappers B. W., Palliyaguru N. T., Miller J., 2009, MNRAS, 400, 1439

Macquart J.-P., et al., 2010, PASA, 27, 272

Manchester R. N., Taylor J. H., 1972, Astrophys. Lett., 10, 67

Manchester R. N., et al., 1996, Monthly Notices of the Royal Astronomical Society, 279, 1235

Manchester R. N., et al., 2001, MNRAS, 328, 17

Marcote B., et al., 2017, ApJ, 834, L8

Masui K., et al., 2015, Nature, 528, 523

McConnell D., et al., 2016, PASA, 33, e042

McLaughlin M. A., et al., 2006, Nature, 439, 817

NCRA 2006, The GMRT: System Parameters and Current Status, `http://www.gmrt.ncra.tifr.res.in/gmrt_hpage/Users/doc/GMRT-specs.pdf`

NVIDIA Corporation 2006, Technical Brief: NVIDIA GeForce 8800 GPU Architecture Overview, `http://www.nvidia.com/object/IO_37100.html`

Newton L. M., Manchester R. N., Cooke D. J., 1981, MNRAS, 194, 841

Oppermann N., Yu H.-R., Pen U.-L., 2018, MNRAS, 475, 5109

Padman R., 1995, in Emerson D. T., Payne J. M., eds, Astronomical Society of the Pacific Conference Series Vol. 75, Multi-Feed Systems for Radio Telescopes. pp 3–26

Perley R. A., Butler B. J., 2017, ApJS, 230, 7

Petroff E., et al., 2015, MNRAS, 451, 3933

Petroff E., et al., 2016, Publications of the Astronomical Society of Australia, 33, e045

Popov S., Postnov K., 2015

Price D. C., et al., 2018a, Research Notes of the American Astronomical Society, 2, 30

Price D. C., et al., 2018b, Research Notes of the AAS, 2, 30

Pshirkov M., Postnov K., 2010, Ap&SS, 330, 13

Radhakrishnan V., Manchester R. N., 1969, Nature, 222, 228

Radhakrishnan V., Srinivasan G., 1982, Current Science, 51, 1096

Ransom S. M., Demorest P., Ford J., McCullough R., Ray J., DuPlain R., Brandt P., 2009, in American Astronomical Society Meeting Abstracts #214. p. 605.08

Ravi V., Lasky P., 2014, MNRAS, 441, 2433

Reifenstein E. C., Brundage W. D., Staelin D. H., 1969, Phys. Rev. Lett., 22, 311

Roshi D. A., et al., 2015, in 2015 IEEE International Symposium on Antennas and Propagation USNC/URSI National Radio Science Meeting. pp 1376–1377, doi:10.1109/APS.2015.7305077

Scheuer P., 1968, Nature, 218, 920

Schnitzeler D., 2012, MNRAS, 427, 664

Spitler L. G., et al., 2016, Nature, 531, 202

Staveley-Smith L., et al., 1996, PASA, 13, 243

Stovall K., et al., 2014, ApJ, 791, 67

Taylor J. H., 1974, A&AS, 15, 367

Taylor J. H., Weisberg J. M., 1982, ApJ, 253, 908

Tendulkar S. P., et al., 2017, ApJ, 834, L7

Teoh A., 2015, The ATNF Pulsar Database, `http://www.atnf.csiro.au/people/pulsar/psrcat`

The Nobel Prize Committee 1993, The Nobel Prize in Physics 1993, `http://www.nobelprize.org/nobel_prizes/physics/laureates/1993/`

Thornton D., Stappers B., Bailes M., et al., 2013, Science, 341, 53

Totani T., 2013, PASJ, 65, 12

Tuthill J., Hampson G., Bunton J., Brown A., Neuhold S., Bateman T., de Souza L., Joseph J., 2012, in 2012 International Conference on Electromagnetics in Advanced Applications. pp 1067–1070, doi:10.1109/ICEAA.2012.6328788

Wang W., Luo R., Yue H., Chen X., Lee K., Xu R., 2018, The Astrophysical Journal, 852, 140

Welford B. P., 1962, Technometrics, 4, 419

Yamasaki S., Totani T., Kiuchi K., 2017, preprint, (arXiv:1710.02302)

Yu Y., Cheng K., Shiu G., Tye H., 2014, JCAP, 11, 1

Zhang B., 2014, ApJL, 780