The Astrophysical Journal, 835:11 (13pp), 2017 January 20
doi:10.3847/1538-4357/835/1/11

AN ACCURATE AND EFFICIENT ALGORITHM FOR DETECTION OF RADIO BURSTS WITH AN UNKNOWN DISPERSION MEASURE, FOR SINGLE-DISH TELESCOPES AND INTERFEROMETERS

Barak Zackay and Eran O. Ofek
Benoziyo Center for Astrophysics, Weizmann Institute of Science, 76100 Rehovot, Israel; bzackay@gmail.com, Eran.ofek@weizmann.ac.il

Received 2014 November 4; revised 2016 November 23; accepted 2016 November 28; published 2017 January 16

ABSTRACT

Astronomical radio signals are subjected to phase dispersion while traveling through the interstellar medium. To optimally detect a short-duration signal within a frequency band, we have to precisely compensate for the unknown pulse dispersion, which is a computationally demanding task. We present the "fast dispersion measure transform" algorithm for optimal detection of such signals. Our algorithm has a low theoretical complexity of $2N_f N_t + N_t N_\Delta \log_2(N_f)$, where N_f, N_t, and N_Δ are the numbers of frequency bins, time bins, and dispersion measure bins, respectively. Unlike previously suggested fast algorithms, our algorithm conserves the sensitivity of brute-force dedispersion. Our tests indicate that this algorithm, running on a standard desktop computer and implemented in a high-level programming language, is already faster than the state-of-the-art dedispersion codes running on graphical processing units (GPUs). We also present a variant of the algorithm that can be efficiently implemented on GPUs. The latter algorithm's computation and data-transport requirements are similar to those of a two-dimensional fast Fourier transform, indicating that incoherent dedispersion can now be considered a nonissue while planning future surveys. We further present a fast algorithm for sensitive detection of pulses shorter than the dispersive smearing limits of incoherent dedispersion. In typical cases, this algorithm is orders of magnitude faster than enumerating dispersion measures and coherently dedispersing by convolution. We analyze the computational complexity of pulsed signal searches by radio interferometers. We conclude that, using our suggested algorithms, maximally sensitive blind searches for dispersed pulses are feasible using existing facilities. We provide an implementation of these algorithms in Python and MATLAB.

Key words: methods: data analysis – methods: statistical

1. INTRODUCTION

When a radio pulse propagates through the interstellar and intergalactic plasma, different frequencies travel at different speeds. This phenomenon, known as dispersion, hinders the detection of radio pulses. This is because integrating over a wide bandwidth during a given time frame dilutes the signal with noise, as only a narrow bandwidth contains the signal's energy at any given interval within the integration frame. The solution to this problem is to dedisperse the signal (i.e., to apply frequency-dependent time delays to the signal prior to integration). Since in most cases the dispersion is a priori unknown, we need to test a large number of dispersion values. The best dispersion is the one that maximizes the signal-to-noise ratio (S/N) of the pulse. A different way to look at this problem is that we need to integrate the flux along many dispersion curves in the frequency–time domain.

The difference in pulse arrival time between two frequencies is given by

$$\Delta t = t_2 - t_1 = 4.15\mathrm{DM}(f_1^{-2} - f_2^{-2})\ \mathrm{ms}, \qquad (1)$$

where DM is called the dispersion measure of the signal, given by the column density of free electrons along the line of sight to the source (measured in units of pc cm^{-3}), f_i are frequencies measured in GHz, and t_i is the arrival time of the signal at frequency f_i. For brevity, throughout the paper, we will use d to denote the dispersion measure in which all the dimensional constants are absorbed, and the relation is given by

$$\Delta t = t_2 - t_1 = d(f_1^{-2} - f_2^{-2}). \qquad (2)$$

The raw input from a radio receiver is a time series of voltage measurements sampled typically at high frequency (e.g., ~100 MHz). We denote the sampling interval by τ. In order to generate a spectrum ($I[t, f]$) as a function of time (t) and frequency (f), the time series is divided into blocks of size N_f samples, and each block is then Fourier transformed (a process known as short-time Fourier transform, or STFT), and the absolute value squared at each frequency is saved after summing the power from all polarizations N_{pol}.[1]

There are two distinct processes that we can apply to dedisperse a signal: incoherent dedispersion and coherent dedispersion. The term incoherent dedispersion refers to applying frequency-dependent time delays to the $I(t, f)$ matrix, while coherent dedispersion involves applying a frequency-dependent phase shift to the Fourier transform of the raw voltage signal. This subtle difference is important—incoherent dedispersion is only an approximation that is valid under certain conditions that we are going to review shortly (Section 2). A typical input matrix ($I[t, f]$) to incoherent dedispersion is presented in the top panel of Figure 1, while on the bottom we show a zoom-in on the output of the transform.

The exact mathematical description of signal dispersion is the multiplication of the Fourier transform of the raw signal with the phase-only filter:

$$\hat{H}(f) = \exp\left(\frac{2\pi i d}{f + f_0}\right). \qquad (3)$$

(Lorimer & Kramer 2012), where f_0 is the baseband frequency. In order to coherently dedisperse the signal, we can apply the inverse of this shift.

[1] Sometimes, an additional stage of binning is then applied to reduce the time resolution.

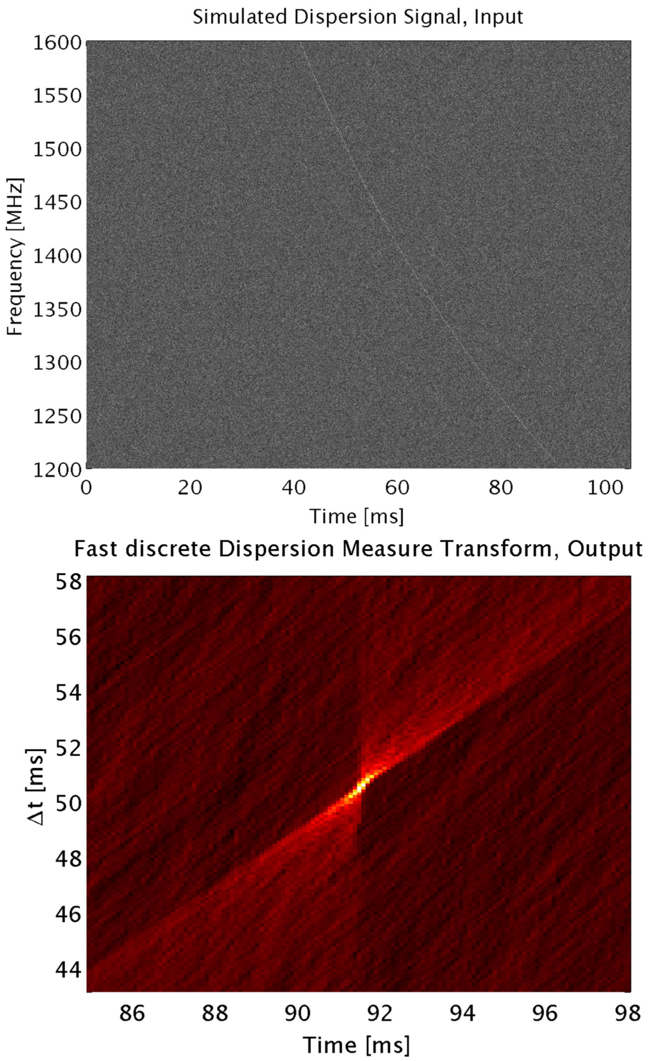

Figure 1. A dispersed signal and its dispersion transform (FDMT), based on simulated data. The top panel shows a 0.1 ms wide dispersed pulse with $D = 40$ pc cm^{-3}. The bottom panel shows a zoom-in on the significant part of the dedispersion transform. Notice that the Y axis has units of time because the dispersion measure is parameterized by the total time delay between the entrance and exit of the pulse into and out of the observed band. Note that because the pulse is so thin, any slight error on the dispersion path will immediately result in a substantial loss of the pulse power. Note also that the characteristic shape visible near the correct solution is due to the fact that curves with close $t0$, Δt can substantially intersect the pulse's curve. This may cause significant detections in these $t0$, Δt combinations. This is not to suggest that match-filtering this shape on the FDMT output is required, as the most informative detection statistic is the value returned in the FDMT output itself.

The computational requirements of incoherent dedispersion are more tractable than those of coherent dedispersion, and therefore whenever a blind search for astrophysical pulses is done, incoherent dedispersion is usually the method of choice (e.g., van Leeuwen & Stappers 2010).

The main practical motivation for improving dedispersion algorithms is to allow efficient analysis of data from modern radio interferometers. When a blind search for new sources is conducted using a multicomponent radio interferometer, coherent dedispersion of a large number of synthesized beams is the most sensitive detection method. But, in reality, coherent dedispersion, even on one beam, was considered (until this work) unfeasible. In addition, applying incoherent dedispersion to all independent sky positions is also generally unfeasible. The standard solutions for this problem were one of the following:

1. Dedisperse incoherently only a small fraction of the field of view.
2. Combine the power from all antennas incoherently, and incoherently dedisperse.
3. Use only a small compact core of the interferometer for blind searches, and incoherently dedisperse.

The above solutions result, each with a different trade-off, in orders-of-magnitude losses in time resolution, sensitivity, field of view, and angular resolution. These losses can degrade the scientific yield of transient surveys by orders of magnitude.

It is therefore important to improve upon these methods, especially when searching for nonrepeating radio transients such as fast radio bursts (FRBs; Lorimer et al. 2007; Thornton et al. 2013). Efficient detection of such objects requires both high sensitivity and a good spatial localization that is calculated in real time. This is crucial for the multiwavelength follow-up of these elusive transients (e.g., Petroff et al. 2014).

In this paper, we present the fast discrete dispersion measure transform algorithm (henceforth FDMT) to solve the problem of incoherent dedispersion. FDMT is a transform algorithm, having (generally) equal sizes for both input and output and a complexity that is only logarithmically larger than the input itself.

In addition, we present a hybrid algorithm that achieves both the sensitivity of coherent dedispersion and the computational efficiency[2] of incoherent dedispersion. Finally, we show that using this algorithm, it is feasible to perform blind searches with modern radio interferometers, and consequently to open new frontiers in the search for pulsars and radio transients.

The structure of the paper is as follows. In Section 2 we analyze the sensitivity of incoherent dedispersion. In Section 3 we review the existing approaches for incoherent dedispersion. In Section 4 we describe the proposed algorithm, along with its complexity analysis. In Section 5 we present a variant of the algorithm that utilizes the fast Fourier transform (FFT) to make the algorithm more efficient on multiprocessor technologies. In Section 6 we compare the runtime of the implementation we provide with existing implementations of brute-force dedispersion. In Section 7 we propose a new hybrid algorithm for detection of pulses shorter than the dispersive smearing limits of incoherent dedispersion. In Section 8 we discuss the application of the proposed algorithms for interferometers, and we show that sensitive detection of short pulses, with maximal resolution, using all the elements of the interferometer, is feasible with current facilities. We conclude in Section 9.

2. SENSITIVITY ANALYSIS OF INCOHERENT DEDISPERSION

In this section, we develop the conditions on the pulse length, the sampling interval, and the dispersion delay that allow sensitive detection with incoherent dedispersion. This will be of importance in Section 7.

We denote by x the raw voltage signal and by N_s the total number of samples. We further denote the pulse duty time

[2] We define efficiency as the ratio between computational operations performed and the amount of calculated scores, each score being the optimal S/N statistic for detecting a pulse with the specific combination of starting time and dispersion measure.

(length) by $t_p = N_p \tau$, where N_p is the number of samples within the pulse. The optimal score for pulse detection is the sum of the squared voltage measurements within the pulse start time (t_0) and end time:

$$S_{\rm opt} = \sum_{j=0}^{N_p} x(t_0 + j\tau)^2. \tag{4}$$

We define $d_{\max}$ to be the largest dispersion measure up to which we want to dedisperse. We further define the maximal dispersion time t_d to be the total delay of the pulse within the band, and we define N_d to be the dispersion time in units of samples:

$$t_d = N_d \tau = d_{\max}(f_{\min}^{-2} - f_{\max}^{-2}). \tag{5}$$

An important property of the dispersion kernel $H(f)$ is that it is power-preserving ($|\hat{H}(f)|^2 = 1$). Another important property is that the majority of pulse power will lie within the dispersion curve in $I(t, f)$. However, by summing over the dispersion curve in I, we also sum the power of the noise outside the pulse, the total variance of which is proportional to the number of added I bins, that is, to the area of the dispersion curve. Assuming the dispersion curve is close to linear, the total area of the dispersion curve can be approximated by

$$\sim N_f \left(\max\left(1, \frac{N_p}{N_f}\right) + \frac{N_d}{N_f^2}\right). \tag{6}$$

Therefore, the ratio of the noise power summed by incoherent dedispersion and the noise power within the pulse is

$$\sim \frac{N_f}{N_p} \left(\max\left(1, \frac{N_p}{N_f}\right) + \frac{N_d}{N_f^2}\right). \tag{7}$$

Optimizing the sensitivity with respect to N_f we get the following:

1. If $N_p < N_f$, the choice of N_f that maximizes sensitivity is
$$N_f = \sqrt{N_d}, \tag{8}$$
regardless of pulse length N_p.
2. If $N_f \leqslant N_p$, the choice of N_f that maximizes sensitivity is
$$N_f = N_p. \tag{9}$$

In order for the sensitivity loss to be less than a factor of $\sqrt{2}$, we equate Equations (7)–(2) and get the following:

1. If $N_f > N_p$, then $N_d < N_f(2N_p - N_f)$. Substituting Equation (8) we get the relation
$$N_d < N_p^2 \tag{10}$$
2. If $N_f < N_p$, then $N_d < N_f N_p$, which together with the assumption yields again the relation
$$N_d < N_p^2. \tag{11}$$

Equation (10) transforms into an important relation between the minimal pulse duration t_p, the maximal dispersion time t_d, and the sampling time τ:

$$\frac{t_d}{\tau} < \frac{t_p^2}{\tau^2} \tag{12}$$

or, simplified,

$$t_p > \sqrt{t_d \tau} \equiv t_{\rm inc}. \tag{13}$$

This means that incoherent dedispersion cannot avoid a minimal dispersion smearing of $t_{\rm inc}$. The sensitivity loss factor from incoherent dedispersion is therefore (substituting Equations (8) and (13) into Equation (7) and simplifying)

$$\frac{({\rm S/N})_{\rm inc}}{({\rm S/N})_{\rm opt}} = \sqrt{\frac{t_p}{\max(t_{\rm inc}, t_p) + t_{\rm inc}}}. \tag{14}$$

Therefore, incoherent dedispersion is insensitive[3] when $t_p \ll t_{\rm inc}$ and approaches maximal sensitivity when $t_p \gg t_{\rm inc}$. This pulse duration depends (weakly) on frequency and dispersion measure and is given by

$$t_{\rm inc} = 3.13 \times 10^{-5} \left(\frac{DM}{100\ {\rm cm^{-3}\,pc}}\right)^{1/2} \times \left(\frac{f_{\min}}{\rm GHz}\right)^{-1} \left(\frac{f_{\max}}{\rm GHz}\right)^{-1} \left(\frac{f_{\min} + f_{\max}}{2\ {\rm GHz}}\right)^{1/2}\ {\rm s}. \tag{15}$$

See Cordes & McLaughlin (2003) for a similar derivation of $t_{\rm inc}$.

Among the astrophysical sources with short pulse durations reside pulses from millisecond pulsars, with durations of $\sim 10^{-3}$ s–$10^{-4.5}$ s (Kramer et al. 1998), FRBs with durations of $\sim 10^{-3}$ s (Thornton et al. 2013; Champion et al. 2015), and pulsar giant pulses, for which the duration can be shorter than 2 ns (Hankins et al. 2003).

As a result of the insensitivity of incoherent dedispersion to very short pulses, the phase space in which $t_p \ll t_{\rm inc}$ is effectively unexplored. For example, past search campaigns for giant pulse-emitting pulsars could not probe this phase space because of the limitations of incoherent dedispersion (see McLaughlin & Cordes 2003; Bhat et al. 2011).

One phenomenon that prevents astrophysical pulses from being infinitesimally narrow is the phenomenon of scattering. Pulse scattering is a widely documented phenomenon in which astrophysical radio pulses broaden as a result of multipath propagation of the pulse through the interstellar medium (ISM). In general, we expect scattering to increase with the dispersion measure, as both depend on the amount of interstellar medium between the source and the observer. An empirical correlation between the observed dispersion measure of pulsars and the observed scattering can be found in Bhat et al. (2004). However, this correlation has a large scatter, which spans orders of magnitude of scattering times per constant dispersion measure. Scattering measures depend on the $\frac{D_l D_s}{D_{ls}}$ ratio, where D_l is the distance to the screen, D_s is the distance to the source, and D_{ls} is the distance between the screen and the source. This ratio may differ substantially for different sources. Possible examples for sources in which this ratio is substantially different than that of pulsars is the speculated population of radio transients from cosmological sources (Macquart & Koay

[3] We call a statistical score or algorithm insensitive if it retains substantially less than $\sim$70% of the optimal S/N.

2013). Another example is the recently found pulsar in the vicinity of the galactic center (Spitler et al. 2014). In both examples, the observed pulse broadening is orders of magnitude shorter than the predicted scattering based on the observed relation between dispersion measure and pulse broadening. In addition, the scaling of scattering with frequency ($\sim\nu^{-4}$) is much steeper than the scaling of $t_{\rm inc}$ with frequency ($\nu^{-3/2}$). Therefore, the need for coherent treatment of dedispersion also depends on the observing band.

As a result of all these complications, we find it more robust to keep the discussion in the rest of the paper to using observed pulse durations, rather than scattering measures, distances, or spectral indices, which are often found in the literature.

3. EXISTING ALGORITHMS FOR INCOHERENT DEDISPERSION

Algorithmically, there are two leading approaches for incoherent dedispersion of single-dish data streams. These are the tree dedispersion (Taylor 1974) and brute-force dedispersion (e.g., Magro et al. 2011; Barsdell et al. 2012; Clarke et al. 2013).

The tree dedispersion algorithm efficiently calculates the integrals of all the straight-line paths with slopes between 45° and 90° through the input time versus frequency matrix (this is similar to the discrete Radon transform, Gotz & Druckmuller 1996; Brady 1998). The computational complexity of this algorithm is $N_t N_f \log_2 N_f$, where we use the notation $N_t = N_s / N_f$ (note that N_t and N_f are the dimensions of $I(t, f)$). However, since the dispersion curve is not linear, the use of this method results in a substantial loss of sensitivity. This can be somewhat mitigated by applying the algorithm to many small subbands[4] of the data, and then combining the results with a dedicated algorithm. This approach is not exact, and it increases the computational complexity of the algorithm. More details on the sensitivity analysis and computational complexity of this algorithm are given in Barsdell et al. (2012).

The brute-force dedispersion algorithm simply scans all of the trial dispersion measures one at a time, integrating its path on the input map and finding curves with excess power. This method is exact but has the high complexity of $N_\Delta N_f N_t$, where N_Δ is the number of trial dispersion measures scanned. In order to expedite the search speed, the algorithm was implemented on graphical processing units (GPUs), and this method is now capable of analyzing single-beam data in real time (Barsdell et al. 2012). Furthermore, as shown by Cordes & McLaughlin (2003), one can reduce the number of dispersion trials by a small factor by optimizing the distances between adjacent dispersion trials. The maximal sensitivity, along with the possibility of real-time analysis using GPUs, makes this method likely to be the most popular algorithm for dedispersion.

Here we present an algorithm that combines both maximal sensitivity and low computational complexity. A comparison of all these mentioned algorithms is summarized in Table 1.

4. THE FDMT ALGORITHM

The input to the FDMT algorithm is a two-dimensional array of intensities, denoted by $I(t, f)$. Denote the width of a frequency bin by δ_f and the width of a time bin by δ_t, satisfying

$$\delta_f = \frac{f_{\rm max} - f_{\rm min}}{N_f}, \quad \delta_t \geqslant N_f \tau. \tag{16}$$

The FDMT algorithm calculates the integral over all curves defined by Equation (2). A dispersion curve can be uniquely defined by the arrival time of the signal at the lowest frequency (t_0) and the time delay between the arrival times of the lowest and highest frequencies (Δt).

Therefore, the FDMT result can be expressed as a two-dimensional array that contains the integrals along dispersion curves as a function of t_0 and Δt:

$$A_{f_{\rm min}}^{f_{\rm max}}(t_0, \Delta t) = \sum_{f=f_{\rm min}}^{f_{\rm max}} I\left(t_0 - d\left[\frac{1}{f_{\rm min}^2} - \frac{1}{f^2}\right], f\right), \tag{17}$$

where $f_{\rm min}$ and $f_{\rm max}$ are the minimum and maximum frequencies in the observed band.

To compute the FDMT transform of the input, the algorithm works in $\log_2(N_f)$ iterations. The inputs of the ith iteration are the FDMT transforms of a partition of the original band into $N_f / 2^{(i-1)}$ subbands of size $2^{(i-1)}$ frequencies. The outputs of the ith iteration are the FDMT transforms of a partition of the original input into $N_f / 2^i$ subbands of size 2^i frequencies. Every two successive subbands are combined using the addition rule described below. After $\log_2(N_f)$ iterations, we have the FDMT over the whole band.

The FDMT process of combining two successive subbands into $A_{f_0}^{f_2}(t_0, \Delta t)$ is given by the following addition rule:

$$A_{f_0}^{f_2}(t_0, \Delta t) = A_{f_0}^{f_1}(t_0, t_0 - t_1) + A_{f_1}^{f_2}(t_1, t_1 - t_2). \tag{18}$$

Here, $A_{f_0}^{f_1}$ and $A_{f_1}^{f_2}$ are part of the output of the previous iteration, and t_1 is the intersection time of the dispersion curve at the central frequency $f_1 = \frac{f_2 + f_0}{2}$. And t_1 is uniquely determined by the formula

$$t_1 \equiv t_0 - \Delta t \frac{f_1^{-2} - f_0^{-2}}{f_2^{-2} - f_0^{-2}} \equiv t_0 - C_{f_2, f_0} \Delta t, \tag{19}$$

where

$$C_{f_2, f_0} \equiv \frac{f_1^{-2} - f_0^{-2}}{f_2^{-2} - f_0^{-2}}. \tag{20}$$

By definition, $A_{f_0}^{f_1}(t_0, t_0 - t_1)$ is the calculated sum over the unique dispersion curve between the coordinates (t_0, f_0) and (t_1, f_1), and $A_{f_1}^{f_2}(t_1, t_1 - t_2)$ is the same for (t_1, f_1) and (t_2, f_2). After an FDMT iteration, the only dispersion curve passing through (t_0, f_0), (t_2, f_2) will be given by $A_{f_0}^{f_2}(t_0, t_0 - t_2)$.

For sufficiently early t_0, the time t_1 will be smaller than zero. In that case we just copy, that is, use the alternative addition rule:

$$A_{f_0}^{f_2}(t_0, \Delta t) = A_{f_0}^{f_1}(t_0, t_0 - t_1). \tag{21}$$

The operation of one iteration of the algorithm is graphically illustrated in Figure 2.

The only thing left to deal with is the data initialization.

This is done prior to the first iteration, generating $A_{f_0}^{f_1}$ for every two consecutive frequencies. If the maximum dispersion delay between two consecutive frequencies is smaller than the

[4] A subband of the data is a part of the data that is both limited and continuous in frequency.

Table 1
Algorithm Comparison

	FDMT	Brute Force	Tree
Computational complexity	$\max\{N_t N_\Delta \log_2(N_f),\ 2N_f N_t\}$	$N_f N_t N_\Delta$	$N_f N_t \log_2(N_f)$
Information efficient	Yes	Yes	No
Memory-access friendly	Yes	Yes	Yes
Parallelization friendly	Yes	Yes	No

Note. Comparison of the FDMT algorithm with two other approaches to incoherent dedispersion, brute force (e.g., Barsdell et al. 2012) and tree dedispersion (Taylor 1974). The computational complexity of FDMT is at least two orders of magnitude smaller than that of brute-force dedispersion for typical $N_f \sim 2^{10}$, while retaining maximal sensitivity.

width of a time bin, then we can use the simple initialization:

$$A_f^{f+\delta_f}(t, 0) = I(t, f). \tag{22}$$

Otherwise, the energy of the signal at some frequencies is not located at a single bin. Note that this implies $N_\Delta \gg N_f$, which we show in Section 2 to be suboptimal in terms of sensitivity. This can be compensated for in two ways:

1. By computing partial averages over the time axis:

$$A_f^{f+\delta_f}(t, \Delta t) = \frac{\delta_t}{\Delta t}\sum_{j=0}^{\frac{\Delta t}{\delta_t}} I(t + j\delta_t, f). \tag{23}$$

   Note that if the maximal dispersion measure is large, multiple Δt's are computed, and this step, unlike the FDMT iterations, inflates the data.
2. If for a certain d the time delay within each single frequency bin is larger than one time bin, we can simply reduce the time resolution (i.e., bin). This process is analogous to rebinning before scanning dispersion measures above the "diagonal DM" using tree dedispersion.

Note that option 1 is more sensitive than option 2 in general as there might be a mismatch between the pulse phase and the boxcar phase. For further discussion, see Keane & Petroff (2015). In the MATLAB and Python codes we provide, we use option 1. The maximal time delay within each frequency bin is uniquely determined by $d_{\max}$, the maximal d we want to scan, and is given by

$$\Delta t_{\max}(f_0) = d_{\max}(f_0^{-2} - (f_0 + \delta_f)^{-2}). \tag{24}$$

Therefore, this decision can be made prior to the computation.

A pseudocode of FDMT is given in Algorithm 1. In addition, we provide implementation for the algorithm in Python and MATLAB.[5] Note that so far we have not treated rounding and binning issues; these are discussed in Section 4.2 and are implemented in the codes we provide.

Algorithm 1. The FDMT Algorithm

```
Input: $I(f, t)$ input matrix (possibly packed); $t$ axis is continuous in memory.
Output: Time series of integrated flux density as a function of trial dispersion
    measures. Output is arranged in a (possibly packed) two-dimensional table
    $A_{f_{\min}}^{f_{\max}}(t, \Delta t)$ where $\Delta t$ represents the dispersion measure axis.
Initiate the table by $A_f^{f+\delta_f}(t, \Delta t) = \frac{\delta_t}{\Delta t}\sum_{j=0}^{\Delta t/\delta_t} I(f, t + j\delta_t)$
for iteration $j = 1$ to $j = \log_2 N_f$ do
  for $f_0$ in the range $[f_{\min}, f_{\max}]$ with steps $2^j\delta_f$ do
    $f_2 = f_0 + 2^j\delta_f$
    $f_1 = \frac{f_2 + f_0}{2}$
    $C_{f_2,f_0} = \frac{f_1^{-2} - f_0^{-2}}{f_2^{-2} - f_0^{-2}}$
    $\Delta t_{\max}(j, f_0) = d_{\max}(f_0^{-2} - f_2^{-2})$
    for $\Delta t$ in the range $[0, \Delta t_{\max}(j, f_0)]$ with steps $\delta_t$ do
      for $t_0$ in range $[C_{f_2,f_0}\Delta t, N_t]$ with steps $\delta_t$ for
        $t_1 = t_0 - C_{f_2,f_0}\Delta t$
        $A_{f_0}^{f_2}(t_0, \Delta t) = A_{f_0}^{f_1}(t_0, t_0 - t_1) + A_{f_1}^{f_2}(t_1, t_1 - t_2)$
      end for
      for $t_0$ in the range $[0, C_{f_2,f_0}\Delta t]$ with steps $\delta_t$ do
        $A_{f_0}^{f_2}(t_0, \Delta t) = A_{f_0}^{f_1}(t_0, t_0 - t_1)$
      end for
    end for
  end for
end for
```

4.1. Computational Complexity

To calculate the computational complexity, we need to trace the number of operations done throughout the algorithm. The amount of additions in iteration j is bounded from above by $N_b N_t N_\Delta(j)$, where N_b is the number of subbands processed at the current iteration, and $N_\Delta(j)$ is the number of distinct dispersion measures needed to calculate for a subband at iteration j:

$$N_\Delta(j, f_0) = \frac{d_{\max}(f_0^{-2} - (f_0 + 2^j\delta_f)^{-2})}{\delta_t}, \tag{25}$$

$$N_\Delta(j) = \max_{f_0}\{N_\Delta(j, f_0)\}$$
$$= \frac{d_{\max}(f_{\min}^{-2} - (f_{\min}+2^j\delta_f)^{-2})}{\delta_t}. \tag{26}$$

As a first-order approximation, one can assume that the dispersion curve is almost linear, meaning that the number of unique dispersion trials needed in iteration $j + 1$ is roughly twice the number needed in iteration j. In the last iteration, $N_\Delta(j_{\max}) = N_\Delta$, and therefore, in iteration j,

$$N_\Delta(j) \approx \frac{N_\Delta 2^j}{N_f}. \tag{27}$$

The number of bands (N_b) in each iteration is $N_b = \frac{N_f}{2^j}$. Therefore, under the approximation of almost linear dispersion (or narrow band), the following approximation is correct:

$$N_b N_\Delta(j) \approx \max\{N_\Delta, N_b\}. \tag{28}$$

[5] The Python code is available from https://sites.google.com/site/barakzackayhomepage/home, and the MATLAB code is available from http://webhome.weizmann.ac.il/home/eofek/matlab/ (Ofek 2014).

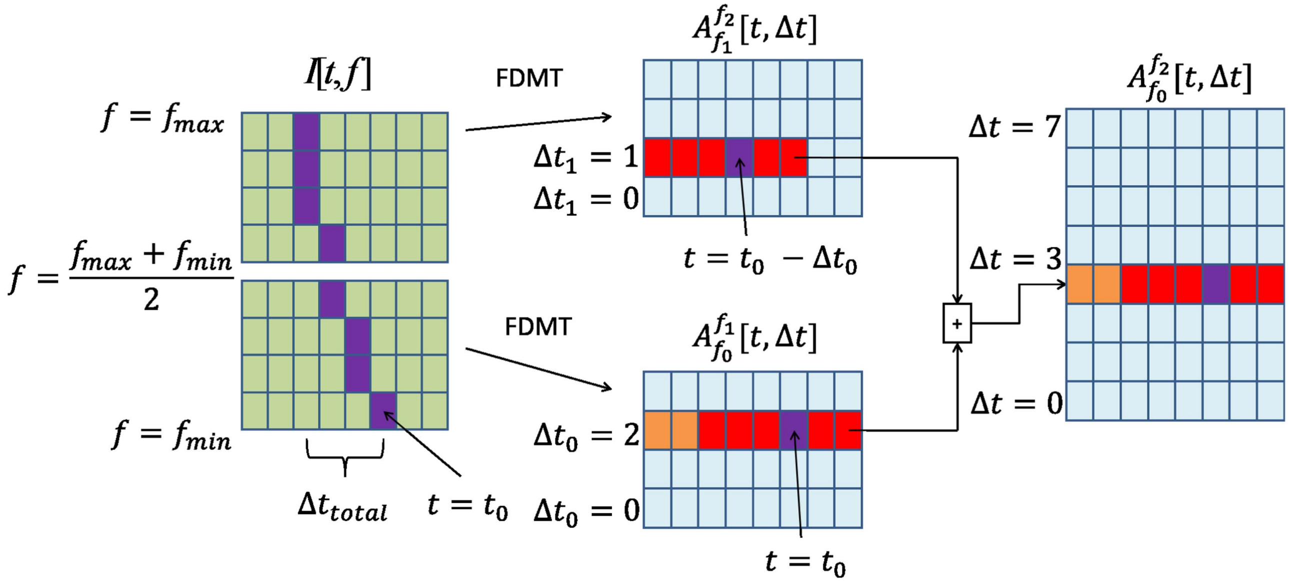

Figure 2. Illustration of a single iteration of the FDMT algorithm. On the left side, the input frequency vs. time input table is drawn. An example dispersion curve is highlighted in purple. The input table is methodically divided into two subbands. The right-hand table shows the final dedispersion transform of the input (left), where the integral over the purple dispersion curve (left) is marked by the purple cell on the right. In the middle, the dispersion measure transform of the two subbands (which are assumed to be calculated in the previous iteration) are drawn, where each of the two subbands contains in each cell the sum of the unique dispersion trail with exit point t and total delay Δt through the corresponding subbands. The dispersion measure dimension is parameterized using $\Delta t_0 = t_0 - t_1$ and $\Delta t_1 = t_1 - t_2$. The cells that contain the partial sums of the two halves of the purple dispersion curve on the left are highlighted in purple. Highlighted in red is a row in the dispersion tables that contributes to the calculation of the red cells on the right. Notice that we can add the lines highlighted in red as vectors, in order to implement the algorithm in a vectorized form. Highlighted in orange are the cells that use the alternative addition rule, in the case when the dispersion trail exits the boundaries of the input table.

Summing this for all iterations, and assuming N_Δ is dominant in all iterations, and taking into account the number of entries in each added row (N_t), we get the complexity

$$C_{\rm FDMT} = N_t N_\Delta \log_2(N_f). \quad (29)$$

If we assume N_b is dominant, we get

$$C_{\rm FDMT} = N_t N_f + \frac{N_t N_f}{2} + \ldots = 2N_t N_f. \quad (30)$$

Therefore, the total complexity of the algorithm is bounded from above by

$$C_{\rm FDMT} \leqslant 2N_t N_f + N_t N_\Delta \log_2(N_f). \quad (31)$$

Casting the complexity analysis of the FDMT algorithm with the more naturally defined N_s, N_d, using $N_s = N_f N_t$ and $N_\Delta = \frac{t_d}{\delta_t} \leqslant \frac{N_d}{N_f}$ we get

$$C_{\rm FDMT} \leqslant 2N_s + \frac{N_s N_d}{{N_f}^2} \log_2(N_f). \quad (32)$$

Adding to the above the complexity of data preparation by STFT, $N_s N_{\rm pol} \log_2(N_f)$, and the fact that if we chose $N_f < N_p$ we can effectively bin the input matrix $I(f, t)$ in both time and frequency domains by a factor $\frac{N_p}{N_f}$ (or low pass, see Section 5.1) to a final size of N_p, we get

$$C_{\rm FDMT} \leqslant N_s N_{\rm pol} \log_2 N_f + \frac{2N_f N_s}{N_p} + \frac{N_d N_s}{{N_p}^2} \log_2(N_f). \quad (33)$$

Here we can see that the data preparation complexity dominates the operation count of the algorithm whenever incoherent dedispersion is maximally sensitive (i.e., $N_d < {N_p}^2$).

4.2. Implementation Details

Here, we consider implementation issues, such as rounding and binning, pulse profile convolution, and dividing the band into an arbitrary number of frequency channels. In addition, it is important to implement the tricks of the trade, in order to transform the theoretical complexity reduction into a real speedup.

Rounding and Binning: The exact formulas written above need to take into account the discreteness of both frequency and time axes. To keep the formulas readable, we did not include these considerations in the algorithm description and pseudocode. However, we include them in the implementation we provide, and we advise readers who want to implement the FDMT to pay attention to the discretization process. That is because an incorrect choice might lead to a significant reduction in accuracy.

An example of the most important discretization issue is that, when combining two subbands, the point t_1 where the dispersion curve travels from one subband to the next might not be well defined. This can happen because the dispersion curve might travel one bin between the end frequency of the first band $f_0 + (2^i - 1)\delta_f$ and the start frequency of the second band $f_0 + 2^i \delta_f$. The implemented solution for this problem is to calculate two versions of C_{f_0,f_2}, one with the end frequency of the lower subband, and the other with the start frequency of the upper subband. Using the different versions of C_{f_0,f_2} in the two different uses of t_1, we can account for a time shift between the added bands, approximating the dispersion curve better.

Machine Word Utilization: One can utilize the machine word width (or the Advanced Vector Extensions (AVX)) to pack a few instances of the dedispersion procedure into one computation (since modern computers operate on machine words of 64 bits, this will result in a speedup factor of 4–8 depending on the number of bits per frequency and the pulse maximum allowed strength).

Memory Access: An important issue in runtime reduction is the continuity of memory access. The FDMT algorithm never performs any reordering action along the time axis. Therefore, it is recommended to store the time axis continuously in order to speed up the memory-access operations. We note that this choice is not standard, and that most telescope data streaming systems store the data in time-major ordering. Therefore, a data transpose operation may be required in order to apply FDMT efficiently.

Applying Window Functions and Filter Banks: The process of transforming from the time domain to the time–frequency matrix was described in this paper as a simple short-time Fourier transform, though in practice it is generally recommended to use a polyphase filter bank instead. While neither the computation complexity nor the sensitivity of the transient search is affected by this, the resilience to radio frequency interference (RFI) and spectral leakage is greatly improved. As a result, this is not a main concern for this paper.

Mitigating the Sensitivity Loss or Squeezing Additional Speedup: It is possible to increase the sensitivity of a pulse search (or alternatively, viewing it as an additional speedup) by choosing a slightly lower time-dm resolution for the FDMT, marking all of the pulse candidates above some (low) threshold. Then, for each candidate we can scan the finer resolution (t_0, d) in order to make the final decision using a score with maximal sensitivity.

An example scheme for saving computation time is as follows. In the first step, aim at half the maximal sensitivity of a full-blown search by using four times the optimal bin size for both the frequency and time dimensions. If a significant pulse is to be detected in the final stage, it should have significance larger than $\sim 8\sigma$. This means that at half the sensitivity, this pulse will have a mean significance of 4σ, and thus, assuming the dedispersed scores have a Gaussian distribution, with high probability ($>85\%$) the correct (t_0, d) combination will pass the 3σ threshold. Each random (t_0, d) combination has a probability of roughly $1/750$ to pass this threshold. Assuming the number of dispersion measures (in the coarse resolution) enumerated was $N_\Delta < 10^{3.5}$, then the computational work for dedispersing all of the candidates with maximal sensitivity is smaller than or equal to the complexity of the FDMT algorithm itself. Using this scheme, it is possible to save roughly an additional order of magnitude in the computation time of the dedispersion step without sensitivity losses or a significant false-negative probability. Using a similar scheme (with 4σ as threshold), it is possible to eliminate most of the built-in $\sqrt{2}$ sensitivity loss of the finite (t_0, d) grid without significant additional work.

Different Range of Dispersion Measures: Sometimes we have a prior knowledge of the range of dispersion measures we need to scan. In that case, one can still employ the FDMT algorithm after an additional preparation of applying either a frequency-dependent shift to the input (according to some $d_{\min}$) or a coherent dedispersion of the signal (using $d_{\min}$).

Pulse Profile: Sometimes we have prior knowledge on the pulse width or profile (there might be a different profile for each frequency, like in pulse scattering). By applying a matched filter approach, one can convolve each frequency time series with the predicted profile for that frequency and employ the FDMT at the end. For wide-enough pulse shapes, one might consider binning the time resolution.

We note that convolution of the time axis with a uniform pulse profile (for all frequencies) commutes with the entire FDMT operation. Therefore, we can test a few pulse profiles per FDMT without repeating the dedispersion process. An important note is that such an operation changes the effective variance of the detection score in a nontrivial way (because the variance change in each DM trial will be different due to different path lengths in the time–frequency matrix). We therefore recommend either keeping careful accounting of the variance through the algorithm, or simply measuring it empirically for each combination of DM trial and pulse profile separately.

Dealing with the Case of $N_f \neq 2^k$: The algorithm presented above assumes that the number of frequency channels is strictly a power of two. This assumption can be abandoned by slightly adjusting the addition rule to allow a merger of nonequal-size subbands. The only change needed is to switch f_1 in Equation (20) from being the middle frequency between f_0 and f_2 to being the border frequency between the subbands.

Applying FDMT for Other Functional Forms: The dispersion equation (Equation (2)) is used only in the preparation of C_{f_2,f_0}. One can easily extend the FDMT algorithm to search for other functional forms, for example,

$$\Delta t = f_1^\gamma - f_2^\gamma. \tag{34}$$

The only required modification is to change the power of the frequency in Equation (20) from -2 to γ. Furthermore, it could be extended to any family of curves that satisfies the condition that there is only one curve passing between any two points in the input data. Using this, one can calculate the required C_{f_2,f_0} by finding the only curve passing through both (t_0, f_0) and (t_2, f_2), and defining t_1 to be the intersection time of the curve with the frequency f_1. While the complexity of the algorithm may change with the functional form, for a sufficiently regular functional form, the complexity will be close to $N_d N_t \log N_f$.

5. ELIMINATING SHIFTS BY FAST FOURIER TRANSFORMING THE TIME AXIS

In modern computers and GPUs, memory access is frequently the bottleneck of many algorithms, especially when programming transforms, where the computational complexity is only slightly larger than the data size. Efficient implementation of transform algorithms is nontrivial and requires architecture-dependent changes in order to avoid cache misses[6] (in a general CPU setup) or to avoid communication when using distributed computing. While it is probably possible to control the behavior of the algorithm as presented above, it is nontrivial to distribute the data between different processing units while avoiding duplication and communication issues.

Here, we present a variant of the algorithm that is easily parallelized on all architectures and where the memory-access pattern is as parallelization-friendly as possible.

The algorithm, as it is described in Section 4, has only one core operation: adding a complete shifted "time" row. It is the

[6] In modern computers, the fastest memory buffer is the L1 cache. An access to a value that is not stored in the L1 cache causes a memory read from slower storage media such as the L2 cache or the RAM memory, and this is sometimes called a "cache miss."

shift operation that makes the data transfer and memory management of the algorithm challenging, and therefore we wish to eliminate shifts from the algorithm. In order to do that, we can Fourier transform the time axis. This makes the shift operation become a multiplication with a "shift vector" that is the Fourier transform of a shifted delta function. In this version, all additions are of numbers from the same (Fourier transformed) time coordinate.[7] Therefore, we can assign different parts of the (Fourier transformed) time axis to different processing units and consequently reduce the need for shared memory or data transport. At the end, we need to Fourier transform back the time axis. We call this algorithm FFT–FDMT, and it is summarized in Algorithm 2. Tracking the data in this algorithm, we can see that there are only two "global" steps and that they are both transpose operations of the data. To perform all other steps of the algorithm, we need only to access memory that is not larger than one row or one column of the input. Since present L1 cache architectures can contain more than a typical row or column of data, the algorithm can be computing-power limited. The runtime of this algorithm on any machine is comparable to the runtime of two-dimensional convolution, because of the similar number of operations and data-access patterns. We note that in the basic preparation of radio data, one often applies Fourier transforms (for example, when applying filters or screening for radio frequency interference). Therefore, if we have the computational ability to prepare the input table from the raw data, FFT–FDMT is also feasible.

Algorithm 2. The FFT–FDMT Algorithm

```
Input: $I(f, t)$ input matrix (possibly packed); $t$ axis is continuous in memory.
Output: Time series of integrated flux density as a function of trial dispersion
  measures. Output is arranged in a (possibly packed) two-dimensional table
  $A_{f_{\min}}^{f_{\max}}(t, \Delta t)$ where $\Delta t$ represents the dispersion measure axis.
Initiate the table by
  $$A_f^{f+\delta f}(t, \Delta t) = \frac{\delta_t}{\Delta t}\sum_{j=0}^{\Delta t/\delta_t} I(f, t + j\delta_t)$$
Initiate the "shift vector" $V(\tilde{t}_0, \Delta t) = \mathcal{F}(\delta(\Delta t))(\tilde{t}_0)$ where $\delta(x)$ is a vector
  containing 1 at position $x$ and zeros everywhere else, $\mathcal{F}$ is the FFT operator,
  and $\tilde{t}_0$ is the index of the Fourier transformed time axis.
Fourier transform the time axis
  $$B_f^{f+\delta_f}(\tilde{t}, \Delta t) = \mathcal{F}_t(A_f^{f+\delta_f}(t, \Delta t))$$
Transpose the data. After this action, the frequency and $\Delta t$ axes should be
  continuous in memory, and the time axis should be distributed across all
  computing units.
for $\tilde{t}_0$ in the range $[0, N_t]$ do
  for $j$ in the range $[1, \log_2 N_f]$ do
    for $f_0$ in the range $[f_{\min}, f_{\max}]$ with steps $2^j\delta_f$ do
      $f_2 = f_0 + 2^j\delta_f$
      $f_1 = \frac{f_2 + f_0}{2}$
      $C_{f_2,f_0} = \frac{f_1^{-2} - f_0^{-2}}{f_2^{-2} - f_0^{-2}}$
      $\Delta t_{\max}(j, f_0) = N_\Delta \frac{f_0^{-2} - (f_0 + 2^j\delta_f)^{-2}}{f_{\min}^{-2} - f_{\max}^{-2}}$
      for $\Delta t$ in the range $[0, \Delta t_{\max}(j, f_0)]$ do
        $\Delta t_1 = C_{f_2,f_0}\Delta t$
```

(Continued)

```
        $$B_{f_0}^{f_2}(\tilde{t}_0, \Delta t) = B_{f_0}^{f_1}(\tilde{t}_0, \Delta t_1) + B_{f_1}^{f_2}(\tilde{t}_0, \Delta t - \Delta t_1)V(\tilde{t}_0, \Delta t_1)$$
      end for
    end for
  end for
end for
Transpose the data back. Now, time is again continuous in memory.
Perform inverse Fourier transform on the time axis
  $$A_{f_{\min}}^{f_{\max}}(t, \Delta t) = \mathcal{F}_t^{-1}(B_{f_{\min}}^{f_{\max}}(\tilde{t}, \Delta t)).$$
```

5.1. Comments on the Implementation of the FFT–FDMT Algorithm

The FFT–FDMT algorithm is designed to increase the amount of computation per cache replacement. To completely optimize the algorithm for this property, we have to consider special implementation details like cache size and processing unit communication geometry. Though important to an efficient implementation of the algorithm, these details are outside the scope of this paper as they are dependent on architecture. We note that all the details discussed in Section 4.2 are valid also for the FFT–FDMT version, except for the changes listed below.

Machine Word Utilization: The long integer data type is the optimal choice for the regular FDMT algorithm in order to fully utilize the machine word capability. In the Fourier-transformed version of the algorithm, we have to use the complex floating-point data type. Using the floating-point data type, we have to leave unused the bits of the exponent field and leave some more bits unused to retain the floating-point precision needed to perform the Fourier transform operations. Furthermore, some architectures such as GPUs have a clear optimization preference for the 32 bit floating-point data type. However, it is possible to pack another algorithm instance into the complex field of the input vector. Since the result of the FDMT algorithm is real (as a sum of real numbers), packing another input into the imaginary part of the input is possible. The imaginary part of the result will be the second algorithm instance.

Pulse Profile: In addition to the ability to test several pulse profiles per FDMT operation, as explained in Section 4.2, we can further exploit the use of the Fourier-transformed time domain. If the pulse width is slightly larger than one bin, reducing the computational load by binning loses information. Instead, we can effectively apply a low-pass filter on the time axis by either keeping fewer (time-domain) frequencies or multiplying with a filter. This can be both more sensitive than binning the time axis and more efficient than having a high sampling rate.

Handling Large Dispersion Measures: If the maximum dispersion broadens the pulse to more than one time bin per frequency bin, the initialization phase of the algorithm inflates the data from size $N_t N_f$ to size $N_\Delta N_f$ (note that the use of $N_\Delta \gg N_f$ is losing sensitivity, and therefore this part is not considered a crucial part of the algorithm). The partial sum

[7] The Fourier transform of the time axis yields what is known in the literature as the fluctuation power spectrum, α.

Table 2
Runtime Comparison

	This Work	Magro et al. (2011)	Barsdell et al. (2012)
Machine used	Intel Core i5 4690	Tesla C1060 GPU	Tesla C1060 GPU
Programming language	Python (anaconda + accelerate)	C	C
Number of instances packed	5	1	1
Runtime	3.5 s	4.8 s	2.1 s
N_f, N_t (total), N_Δ	2^{10}, 5×2^{16}, 2^{10}	2^8, 2^{19}, 500	2^8, 2^{19}, 500
$N_f N_t N_\Delta$	5×2^{36}	2^{36}	2^{36}
Algorithm used	FDMT (non-FFT version)	Brute force	Brute force
Algorithm theoretical complexity	$N_t N_f + N_t N_\Delta \log_2(N_f)$	$N_f N_t N_\Delta$	$N_f N_t N_\Delta$

Note. The FDMT algorithm has a different computational complexity scaling than the brute-force dedispersion it is compared to. The implementation we provide was intended to serve as a high-level reference implementation for future highly efficient, platform-specialized implementations. But, even with a standard desktop computer, our high-level FDMT implementation is faster than the existing GPU implementations of brute-force dedispersion. Our MATLAB code has similar performance.

operation of the initialization phase is equivalent to an application of a low-pass filter on the time axis. This allows a natural reduction of computation and memory by saving a differential amount of Fouriered time bins for different Δt's. This can be used in the case of large dispersion measures to reduce the algorithm's complexity from $N_\Delta N_t \log_2(N_f)$ to $2N_t N_f \log_2\left(\frac{N_\Delta}{N_f}\right)$.

Zero Padding: Since convolution is a cyclical operation, all the shifts done in this algorithm are cyclical shifts. Therefore, we have to pad the time axis with N_Δ zeros prior to the Fourier transform. This operation can increase by a factor of two the complexity of the algorithm if $N_\Delta = N_t$. To avoid this we can choose $N_t \gg N_\Delta$. This is usually possible if the size of the input table is not too close to the maximum memory (or cache) capacity of the machine used.

6. RUNTIME AND BENCHMARKING

Accurate benchmarking of algorithms should use a mature code and contain architecture-dependent adaptations. However, it is important to demonstrate that the code we present, running on a single standard CPU, is competitive with the brute-force dedispersion implementations on GPUs. Therefore, we provide a simple benchmark for the provided code.

The benchmark we use is the runtime of performing FDMT on data with the following properties: $N_f = 2^{10}$, $N_t = 5 \times 2^{16}$, and $N_\Delta = 2^{10}$. This volume of input is similar to the one used in the "toy observation" defined in Barsdell et al. (2012) and Magro et al. (2011), $N_f = 2^8$, $N_t = 2^{19}$, and $N_\Delta = 500$. However, we modified the partition between N_f, N_t and increased N_d by a factor of two.[8] The runtime we achieve on this data is 3.5 s on a standard Intel Core i5 4690 processor. For example, these numbers can represent a real-time dedispersion of 8 s of input data with 40 MHz bandwidth and 1024 dispersion trials. To get this benchmark, we pack five instances of the algorithm into the 64 bit machine word, allocating 12 bits to each instance. The resulting packed data set has dimensions $N_t = 2^{16}$, $N_f = 2^{10}$ and serves as input to the FDMT implementation. Using this scheme, we find that our runtime is already shorter than that of the state-of-the-art brute-force implementations on GPUs reported in Barsdell et al. (2012) and Magro et al. (2011). A comparison between the run times is shown in Table 2.

7. BRIDGING THE GAP BETWEEN COHERENT AND INCOHERENT DEDISPERSION

Since some interesting transient sources such as pulsar giant pulses are in the regime $1 \ll N_p < \sqrt{N_d}$, it is important to find a feasible and sensitive algorithm for their exact dedispersion. Coherent dedispersion was, until now, the only sensitive alternative. The noise power summed when searching for a pulse that is dispersed with a dispersion measure d_0 is

$$N_p + N_{d_0}. \tag{35}$$

The noise power summed when searching for a nondispersed pulse is N_p, and therefore the largest dispersion tolerable (denoted by δ_d) for sensitive pulse detection satisfies

$$N_{\delta_d} = N_p. \tag{36}$$

Therefore, for sensitive detection, the number of dispersion measure trials we need to process is

$$N_\Delta = \frac{N_d}{N_{\delta_d}} = \frac{N_d}{N_p}. \tag{37}$$

The convolution operation performed for coherent dedispersion can be efficiently calculated with Fourier transforms of size N_p, and therefore the complexity of coherent dedispersion is

$$C_{\text{coherent}} = N_{\text{pol}} \frac{N_d}{N_p} N_s \log_2(N_p). \tag{38}$$

Noting that the computational complexity of coherent dedispersion scales with N_d/N_p and that of incoherent dedispersion scales with N_d/N_p^2, we see that using coherent dedispersion is not computationally efficient for resolved pulses (i.e., $N_p > 1$).

7.1. Hybrid Algorithm for Dedispersion

In order to have both the detection sensitivity of coherent dedispersion and the computational complexity of FDMT, we propose the following solution: coherently dedisperse the raw signal with coarse-trial dispersion values (with steps δd), and then apply STFT and absolute value squared, followed by FDMT with the maximal dispersion being the next coarse-trial coherent dedispersion. This process ensures that the FDMT will not lose sensitivity, relative to coherent dedispersion.

[8] This is a more realistic choice, since using large $N_\Delta > N_f$ usually loses sensitivity (see Section 2), and the number of frequencies is usually larger than 2^{10}.

We denote by $N_{\delta d}$ the number of bins of length τ that a delta function pulse will spread over when dispersed by δd:

$$N_{\delta d} = \frac{4.15\delta d(f_{\min}^{-2} - f_{\max}^{-2})\text{ ms}}{\tau} \tag{39}$$

As shown in Section 2, in order to retain sensitivity, the maximal dispersion residual to be processed by the following FDMT must be bounded from above by

$$N_{\delta d} = 2N_p^2. \tag{40}$$

(The factor of 2 in the above equation is due to the fact that FDMT can be applied to also find negative dispersions). Therefore, the number of dispersion trials we need to coherently dedisperse is

$$N_{\text{coherent}} = \frac{N_d}{2N_p^2}. \tag{41}$$

This process is approaching maximum sensitivity, and its complexity is

$$\begin{aligned} C_{\text{hybrid}} = &\frac{N_d}{2N_p^2}N_s N_{\text{pol}}(\log_2(N_d) + \log_2(N_f)) \\ &+ \frac{2N_d N_s}{{N_p}^2} + \frac{N_d N_s}{{N_p}^2}\log_2(N_f), \end{aligned} \tag{42}$$

where the right-hand part was computed simply by observing that the complexity of FDMT is exactly twice the input dimensions plus $\log(N_f)$ times the output dimensions. Simplifying, we get the computational complexity for detection of a pulse of length N_p:

$$C_{\text{hybrid}} = \frac{N_d N_s}{2{N_p}^2}(4 + N_{\text{pol}}\log_2(N_d) + (2 + N_{\text{pol}})\log_2(N_p)). \tag{43}$$

7.2. Using Convolving Filter Banks

The coarse enumeration using coherent dedispersion may contain very large FFTs, which may be prohibitive because of the need to access large volumes of data simultaneously. This could be mitigated in two ways. The first is by using the minimal dedispersion kernel H_{d_0} to dedisperse the signal from the last iteration, thereby linearly shifting the dispersion measure phase space scanned in each iteration by the correct amount with FFTs of size N_p^2 instead of N_d^2. We can save even further by performing small convolutions on the already channelized data, $I_c(t, f) = \text{STFT}(x(t))$, and dedisperse each channel separately using coherent dedispersion. This process is known as "convolving filter bank" and is detailed in van Straten & Bailes (2011). By combining both methods, we can reduce the complexity of the coarse coherent dedispersion stage to linear time. Essentially, using this process we can use time-domain complex convolutions on the already STFTed data with convolution kernels of very small sizes (which can be as little as convolving with a ~4 tap filter (in addition to a large shift)). This reduces further the computational complexity of hybrid dedispersion to be

$$C_{\text{hybrid}} = \frac{N_d N_s}{{N_p}^2}(cN_{\text{pol}} + \log_2(N_p)), \tag{44}$$

where we have aggregated the complexity of all the short $O\left(\frac{N_d N_s N_{\text{pol}}}{{N_p}^2}\right)$ operations (convolving filter bank, absolute value squared, and FDMT initialization into $c \sim 7$).

This complexity is near optimal because the number of uncorrelated scores is $\frac{N_d N_s}{{N_p}^2}$, which is only a logarithmic factor smaller than the computational complexity. Therefore, there is not much room for further reduction of computational complexity. The algorithm is summarized in Algorithm 3.

Algorithm 3. Coherent Hybrid FDMT Dedispersion Algorithm

```
Input: Antenna voltage series $x(t)$.
Output: Time series of integrated flux density as a function of trial dispersion
    measures. $d < d_{\max}$ with steps $\frac{N_p}{N_d}d_{\max}$ and all exit times $t_0 < N_s\tau$ with
    steps $N_p\tau$.
Apply STFT with block size $N_p$ on $x(t)$ to obtain $I_c(t, f)$.
for dispersion $d_0$ in the range $[0, d_{\max}]$ in steps of $2\frac{{N_p}^2}{N_d}d_{\max}$ do
    Apply the filter $\hat{H}_{d_0}(f) = \exp\left(\frac{2\pi i d_0}{f + f_0}\right)$ to $I_c(t, f)$ via convolving filter bank
        (i.e., apply in separate channels), perform absolute value squared, and sum
        over all polarizations to get $I(t, f)$.
    Apply FDMT to $I(t, f)$, with $-N_p^2\tau \leqslant \Delta t \leqslant N_p^2\tau$, and output the partial
        result $A_{f_{\min}}^{f_{\max}}(d_0 + d, t_0)$.
end for
```

7.3. Implications

Using this algorithm, it is possible to perform blind searches for pulses with duration in the 1 μs to 0.1 ms regime (which implies $N_p = 10^2$–10^4 for standard searches). Searching for pulses with this short duration opens a parameter space that had never been scanned before in a blind survey, and allows us to feasibly search for pulsars via their giant pulse phenomenon.

8. BLIND TRANSIENT SEARCHES WITH RADIO INTERFEROMETERS

There are two existing approaches to blind search interferometry. The first is to add antennas incoherently and then dedisperse. This is considered to be computationally feasible, and it is sensitive to the entire field of view of the interferometer. However, this process loses the angular resolution and reduces the sensitivity by a factor of $\sqrt{N_{\text{a}}}$, where N_{a} is the number of antennas.

The second approach is to beam-form and dedisperse: for every searched location (p_x, p_y), shift all the signals from all antennas with the correct shift for position (p_x, p_y), add them up, and perform dedispersion. To mitigate the computational load of this process, it is customary to use only a small subset of all sky locations at a time, considerably reducing the overall survey speed of the instrument.

Another possibility is to use a combination of both approaches by dividing the interferometers into closely packed stations, beam-forming all stations to a subset of all possible directions, and then incoherently adding the stations. All methods trade the computational unfeasibility with a significant sensitivity reduction. These approaches are further discussed in van Leeuwen & Stappers (2010), and a review on the existing algorithms for blind transient searches using interferometers can also be found in Bannister & Cornwell (2011).

8.1. Incorporating the FDMT Algorithm into Transient Searches of Radio Interferometers

In this section, we analyze the complexity of blind searches of short astrophysical signals with radio interferometers using FDMT. We first calculate the computational and communication complexity of applying the FDMT algorithm after the imaging operation. Moreover, we offer a way to reduce the communication complexity by applying the FDMT algorithm after the correlator operation and before the imaging operation. We show that in principle, using our scheme, it is possible to use modern radio interferometers to detect and locate short astrophysical pulses in real time without knowledge of the dispersion measure.

We start by introducing some additional notation. In the scenario of a blind dispersed pulse search with a radio interferometer, we have signals from several telescopes. We denote the raw voltage signal from the jth telescope by x_j. We further denote by $N_{\rm a}$ the number of antennas, and by N_l the number of distinct locations on the sky, or pixels, in the optimal image resolution of the interferometer. The desired statistic that we need to calculate for efficient detection is given by

$$S(t_0, p_x, p_y) = \sum_{t=t_0}^{t=t_0+N_p} \left| \sum_{j=0}^{N_a} (x_j * H)(t + u_j(p_x, p_y)) \right|^2, \quad (45)$$

where $u_i(p_x, p_y)$ represents the time delay of the signal at antenna i, H is the dedispersion filter needed to be convolved with to correct for dispersion, and $*$ represents convolution. We wish to calculate this score for all combinations of sky positions, dispersion trials $\frac{N_d}{N_p}$, and start times $\frac{N_s}{N_p}$. Therefore, the number of calculated scores is $\frac{N_l N_s N_d}{N_p{}^2}$.

To illustrate that the search for transients using interferometers is feasible, we estimate the number of computations required for searching FRBs with the core of MeerKAT: $f_{\rm min} = 1$ GHz, $f_{\rm max} = 1.75$ GHz, $\nu = 750$ MHz is the baseband sampling frequency, $t_d = 3.6$ s (corresponds to a target DM of 1000), $t_p = 1$ ms. Using $N_{\rm a} = 48$ antennas of diameter $D_a = 13.5$ m, spread out to a maximal baseline of $D_b = 1$ km, $N_l = \frac{\pi}{4}\left(2\frac{D_b}{D_a}\right)^2 \approx 17,300$ (covering twice the primary beam size in imaging), $N_s = 0.75 \times 10^9$, $N_p = 0.75 \times 10^6$, and $N_d = 2.7 \times 10^9$, we get 6×10^{10} scores per second, which means that the computing power required for performing FDMT (estimating the logarithmic factor in Equation (32) to be 16) is ~1 TFlop/s to process. For comparison, this computational complexity is an order of magnitude less than the required complexity for correlating the signals (~9 TFlop/s). To show the importance of performing dedispersion via FDMT instead of by brute force in the case of interferometers, we calculate the computational complexity of the dedispersion step in the same setup using brute-force dedispersion to be 220 TFlop/s.

8.2. The Proposed Solution

First, we quickly review the standard imaging process of interferometry, using the approximations of a flat sky and short observation. Assuming there is no dispersion, the desired score is

$$\begin{aligned} S(t_0, p_x, p_y) &= \sum_{t=t_0}^{t=t_0+N_p} \left| \sum_{j=0}^{N_a} x_j(t + u_j(p_x, p_y)) \right|^2 \\ &= \sum_{f=f_{\rm min}}^{f_{\rm max}} \left| \sum_{j=0}^{N_a} \hat{x}_j(t_0, f) \exp(-2\pi i f u_j(p_x, p_y)) \right|^2 \\ &= \sum_{f=f_{\rm min}}^{f_{\rm max}} \sum_{j,k=0}^{N_a} \hat{x}_j(t_0, f) \hat{x}_k^*(t_0, f) \\ &\quad \times \exp(-2\pi i f(u_j(p_x, p_y) - u_k(p_x, p_y))), \end{aligned} \quad (46)$$

denoting by $\hat{x}$ the Fourier transform of x. To efficiently calculate this score for every pixel p_x, p_y, we can use the relation

$$u_j(p_x, p_y) - u_k(p_x, p_y) \propto \frac{(L_j - L_k)}{c} \cdot (p_x, p_y), \quad (47)$$

denoting by L_j the two-dimensional location of antenna j on the plane (under the approximation of having all antennas on the same plane). Now we can calculate the score at all positions (p_x, p_y) at the same time by a two-dimensional Fourier transform of the array:

$$\begin{aligned} &\hat{S}(t_0, p_u, p_v) \\ &= \sum_{j,k,f} \hat{x}_j(t_0, f) \hat{x}_k^*(t_0, f) \delta^2\left(\frac{(L_j - L_k)f}{c} - (p_u, p_v)\right) \end{aligned} \quad (48)$$

$$S(t_0, p_x, p_y) = \mathcal{F}^{-1}(\hat{S}(t_0, p_u, p_v)), \quad (49)$$

where $\delta^2((a, b))$ is equal to one if $(a, b) = (0, 0)$ (to the desired approximation), and zero otherwise.

The summation in Equation (48) is a sum of squares. This means that coherent dedispersion operations must be performed before correlating[9] because the imaging process calculates the sum of the squared absolute values of the voltages.

Incorporating dedispersion into this, we can see that the block size N_f we used earlier is transformed in this framework to the size of the Fourier transform done by the correlator. Because of the unknown dedispersion, at each image location we need to get a time–frequency matrix. This means that we need to perform a different two-dimensional Fourier transform operation for each narrow frequency band. After that, we can apply FDMT for every pixel's time–frequency matrix. Denoting the complexity of the ith step of the algorithm by C_i, the complexity of the coherent dedispersion + STFT of all individual antennas is

$$C_1 = \max\left(1, \frac{N_d}{2N_p{}^2}\right) N_a N_s (\log_2(N_d) + \log_2(N_f)). \quad (50)$$

The complexity of correlating all pairs of antennas is

$$C_2 = \max\left(1, 2\frac{N_d}{N_p{}^2}\right) \frac{N_a(N_a - 1)}{2} N_s. \quad (51)$$

[9] The process of calculating $\hat{x}_j(t_0, f)\hat{x}_k^*(t_0, f)$ is referred to as "correlating" in the literature and is calculated by a computing infrastructure usually called "the correlator."

The complexity of the imaging process is

$$C_3 = N_l \log_2(N_l) \frac{N_s N_d}{{N_p}^2}. \tag{52}$$

The complexity of the FDMT algorithm (without the STFT part, which was already done in this context) is

$$C_4 = N_l \frac{N_s N_d}{{N_p}^2} \log_2(N_f). \tag{53}$$

So, the total complexity of this process is

$$C = C_1 + C_2 + C_3 + C_4. \tag{54}$$

While the complexity of this process is indeed "optimal," in the sense that it is only a logarithmic factor larger than the number of independent results, implementing this will result in a reduced computational efficiency. This is due to data transport between the imaging stage and the dedispersion stage. Between these stages, $\frac{N_l N_s N_d}{{N_p}^2}$ complex numbers are being transported.

This could be mitigated by the fact that dedispersion can be done before the imaging operation.

In order to move the FDMT operation before the imaging operation, we need to take some care for chromatic aberrations. If the frequency band is wide enough, then different frequencies might represent different locations in the (u, v) plane. The location on the (u, v) plane where we need to put a correlator output of frequency f and a pair of antennas j, k is

$$(p_u, p_v) = \frac{f(L_j - L_k)}{c}. \tag{55}$$

Therefore, requiring that the visibilities of all frequencies in the frequency band will be on the same bin in the (u, v) plane yields

$$\frac{\max_{j,k}|L_j - L_k|}{c}(f_{\max} - f_{\min}) < \delta p_{uv}, \tag{56}$$

where δp_{uv} is the angular frequency difference between adjacent grid points in the (u, v) plane. If the band is wide or the field of view is large, this condition will not hold for pairs of far-apart antennas. In this case, it is necessary to split the frequencies into subbands that are narrow enough to maintain Equation (56), and after that, perform FDMT on each subband separately. Then, for each final dispersion measure (at the final dispersion resolution, not in the subband's resolution), we can shift the frequency bands in time accordingly, grid the visibilities in the optimal way, and perform the imaging operation via a two-dimensional FFT to get the optimal detection scores in the image plane.

Another issue is that if t_p is short and the required field of view is large enough, it may happen that the time delay for the pulse's arrival between different antennas will be larger than t_p. In such a case, it is important to facet the field of view to areas that are small enough that the time delay of the pulse's arrival time (with respect to the central direction) to different antennas across the narrowed field of view is smaller than t_p. Then, perform this search algorithm to detect bursts within each facet.

Since the FDMT's input and output dimensions have the same size, the communication complexity of the proposed solution is $\frac{N_a(N_a - 1)N_s N_d}{2{N_p}^2}$, which should (if $N_a^2 \ll N_l$) make the algorithm's runtime be computation limited and thus feasible.

This summation-before-imaging scheme was already used by Law et al. (2015) without the preceding step of performing FDMT to reduce the computation time of the dedispersion step. It is important to note that, using our scheme, the computation required to perform the imaging part is always larger than that of the dedispersion part. This is not true when using brute-force dedispersion, which is the most computationally demanding stage when using a combination of a dense antenna placement, large bandwidth, and fine time resolution and when searching for pulses with high dispersion measures. For example, see the computational complexity stated above for searching FRBs with MeerKAT. This process is summarized in Algorithm 4.

Algorithm 4. Finding Short Pulses with Interferometers

```
Input: Antenna voltage series.
Output: $S(d, t_0, p_x, p_y)$ for every time, dispersion, and sky location.
for dispersion $d_0$ in the range $[0, d_{\max}]$ in steps of
        $$d_{\text{step}} = \frac{d_{\max} t_p^2}{t_d \tau}$$
    do
    Initialize $\hat{S}(t_0, f, p_u, p_v) = 0$
    for antenna index $j$. do
        Create the signal $x_j$ by convolving the $j$th antenna signal with the dedispersion filter with index $d_0$.
        Apply STFT with block size of $N_f$ on the signal $x_j$ to obtain $\hat{x}_j$.
    end for
    for every pair of antennas $j$, $k$ do
        for each $t, f$ do
            $(p_u, p_v) = \frac{(L_j - L_k)f}{c}$
            $\hat{S}(t, f, p_u, p_v) = \hat{S}(t, f, p_u, p_v) + \hat{x}_j(t, f)\hat{x}_k(t, f)$
        end for
    end for
    for Each populated $(p_u, p_v)$ do
        $I_{p_u,p_v}(t, f) = \hat{S}(t, f, p_u, p_v)$
        Apply FDMT with $d_{\max} = d_{\text{step}}$.
            $$\hat{S}(t, d, p_u, p_v) = \text{FDMT}(I_{p_u,p_v}(t, f))$$
    end for
    Data transpose operation. Each processing unit now holds tables of the form $\hat{S}_{t,d}(p_u, p_v) = \hat{S}(t, d, p_u, p_v)$
    for each dispersion $d$ and time $t$ do
        Perform two-dimensional inverse Fourier transform to calculate $S(t, d + d_0, p_x, p_y) = \mathcal{F}^{-1}(\hat{S}_{t,d}(p_u, p_v))$.
    end for
end for
```

9. CONCLUSION

We present the FDMT algorithm, which performs an exact incoherent dedispersion transform with a complexity of $2N_t N_f + N_f N_\Delta \log_2(N_f)$. We show that regular implementation tricks of the trade can be combined with the FDMT algorithm to achieve significant computation speedup. We also present a variant of the FDMT algorithm that is slightly more computationally intensive but concentrates all memory-access operations in two global transpose operations, and might present further speedup on massively parallel architectures such as GPUs. We show that the FDMT algorithm dominates all other known algorithms for incoherent dedispersion and has complexity comparable to the signal-processing operations required to generate its input data. Therefore, we conclude that incoherent dedispersion can now be considered a nonissue for future surveys. We provide implementations of the FDMT algorithm in high-level programming languages, which when run on a desktop computer achieve a faster runtime than the state-of-the-art implementations of brute-force dedispersion on cutting-edge GPUs.

We further present an algorithm that bridges the gap between coherent and incoherent dedispersion, and we show that the computational complexity of this algorithm is orders of magnitude lower than that of coherent dedispersion for pulses of resolvable duration. Using this algorithm, it will be possible to perform blind searches for giant pulse emitting pulsars with the sensitivity of coherent dedispersion searches.

Last, we compute the operation count for a blind search of short astrophysical bursts, such as FRBs, with modern radio interferometers and arrive at the conclusion that it is computationally feasible using existing facilities.

We thank the anonymous referee for suggesting the use of convolving filter banks in the hybrid-coherent dedispersion algorithm. In addition, we thank the referee for his useful comments and corrections that greatly improved the readability and quality of the paper. B.Z. would like to express his deep thanks to Gregg Halinnan for introducing him to the problem of dedispersion. The authors would like to express their thanks to Avishay Gal-Yam, Shrinivas Kulkarni, Matthew Bailes, David Kaplan, Ora Zackay, and Gil Cohen for their useful comments and advice regarding the paper. E.O.O. is the incumbent of the Arye Dissentshik career development chair and is grateful to support by grants from the Willner Family Leadership Institute Ilan Gluzman (Secaucus NJ), Israeli Ministry of Science, Israel Science Foundation, Minerva, and the I-CORE Program of the Planning and Budgeting Committee and The Israel Science Foundation.

REFERENCES

Bannister, K. W., & Cornwell, T. J. 2011, ApJS, 196, 16
Barsdell, B. R., Bailes, M., Barnes, D. G., & Fluke, C. J. 2012, MNRAS, 422, 379
Bhat, N. D. R., Cordes, J. M., Camilo, F., Nice, D. J., & Lorimer, D. R. 2004, ApJ, 605, 759
Bhat, N. D. R., Cordes, J. M., Cox, P. J., et al. 2011, ApJ, 732, 14
Brady, M. L. 1998, SIAM Journal on Computing, 27, 107
Champion, D. J., Petroff, E., Kramer, M., et al. 2015, arXiv:1511.07746
Clarke, N., D'Addario, L., Navarro, R., & Trinh, J. 2014, JAI, 3, 50004
Clarke, N., Macquart, J.-P., & Trott, C. 2013, ApJS, 205, 4
Cordes, J. M., & McLaughlin, M. A. 2003, ApJ, 596, 1142
Gotz, W. A., & Druckmuller, H. J. 1996, Pattern Recognition, 29, 711
Hankins, T. H., Kern, J. S., Weatherall, J. C., & Eilek, J. A. 2003, Natur, 422, 141
Keane, E. F., & Petroff, E. 2015, MNRAS, 447, 2852
Kramer, M., Xilouris, K. M., Lorimer, D. R., et al. 1998, ApJ, 501, 270
Law, C. J., Bower, G. C., Burke-Spolaor, S., et al. 2015, ApJ, 807, 16
Lorimer, D. R., Bailes, M., McLaughlin, M. A., Narkevic, D. J., & Crawford, F. 2007, Sci, 318, 777
Lorimer, D. R., & Kramer, M. 2012, in Handbook of Pulsar Astronomy, ed. D. R. Lorimer & M. Kramer (Cambridge, UK: Cambridge Univ. Press), 2012
Macquart, J.-P., & Koay, J. Y. 2013, ApJ, 776, 125
Magro, A., Karastergiou, A., Salvini, S., et al. 2011, MNRAS, 417, 2642
Manchester, R. N., Lyne, A. G., D'Amico, N., et al. 1996, MNRAS, 279, 1235
McLaughlin, M. A., & Cordes, J. M. 2003, ApJ, 596, 982
Ofek, E. O. 2014, MATLAB package for astronomy and astrophysics, Astrophysics Source Code Library, ascl:1407.005
Petroff, E., van Straten, W., Johnston, S., et al. 2014, ApJL, 789, L26
Spitler, L. G., Lee, K. J., Eatough, R. P., et al. 2014, ApJL, 780, LL3
Taylor, J. H. 1974, A&AS, 15, 367
Thompson, D. R., Wagstaff, K. L., Brisken, W. F., et al. 2011, ApJ, 735, 98
Thornton, D., Stappers, B., Bailes, M., et al. 2013, Sci, 341, 53
van Leeuwen, J., & Stappers, B. W. 2010, A&A, 509, A7
van Straten, W., & Bailes, M. 2011, PASA, 28, 1