# Computing the Fourier Transformation over Temporal Data Streams (Invited Talk)

Böhlen, Michael Hanspeter ; Saad, Muhammad

Abstract: In radio astronomy the sky is continuously scanned to collect frequency information about celestial objects. The inverse 2D Fourier transformation is used to generate images of the sky from the collected frequency information. We propose an algorithm that incrementally refines images by processing frequency information as it arrives in a temporal data stream. A direct implementation of the refinement with the discrete Fourier transformation requires $O(N^2)$ $complex multiplications to process an element of the stream. We prop$

DOI: https://doi.org/10.4230/LIPIcs.TIME.2019.1



Originally published at:
Böhlen, Michael Hanspeter; Saad, Muhammad (2019). Computing the Fourier Transformation over Temporal Data Streams (Invited Talk). In: 26th International Symposium on Temporal Representation and Reasoning, TIME 2019, Malaga, Spain, 16 October 2019 - 19 October 2019, 1:1-1:4.
DOI: https://doi.org/10.4230/LIPIcs.TIME.2019.1

# Computing the Fourier Transformation over Temporal Data Streams


**Michael H. Böhlen**
University of Zürich, Switzerland
boehlen@ifi.uzh.ch

**Muhammad Saad**
University of Zürich, Switzerland
saad@ifi.uzh.ch



## Abstract

In radio astronomy the sky is continuously scanned to collect frequency information about celestial objects. The inverse 2D Fourier transformation is used to generate images of the sky from the collected frequency information. We propose an algorithm that incrementally refines images by processing frequency information as it arrives in a temporal data stream. A direct implementation of the refinement with the discrete Fourier transformation requires $O(N^2)$ complex multiplications to process an element of the stream. We propose a new algorithm that avoids recomputations and only requires $O(N)$ complex multiplications.



**2012 ACM Subject Classification** Information systems → Stream management; Theory of computation → Data structures and algorithms for data management

**Keywords and phrases** Data streams, Fourier transform, time-varying data

**Digital Object Identifier** 10.4230/LIPIcs.CVIT.2016.23

**Category** Invited paper

**Funding** *Muhammad Saad*: partially supported by the Swiss National Science Foundation (SNSF) through project number 407550_167177


## 1 Introduction

The discrete Fourier transform (DFT) converts a function of time into a function of frequency. The inverse of the DFT restores a time-varying function from its Fourier transform. In radio astronomy, high resolution images of celestial objects, such as stars and galaxies, are produced by directly measuring radio signals with an array of radio antennas. Each pair of antennas measures a visibility: a complex value that encodes amplitude and phase information of a radio signal. Each measured visibility value quantifies a single frequency component of the radio signal measured at a single time and a single $(u, v)$ coordinate in the $uv$-plane. The main concept in radio astronomy is that the image of the sky has a Fourier transform that can be measured directly by a radio interferometer in the form of visibilities. A single visibility carries limited information about the spatial structure of the source. Hence, to produce a sky image with accurate spatial information, visibility values are measured at different coordinates in the $uv$-plane. The sky image is generated by taking the inverse 2D-DFT of the visibilities in the $uv$-plane.

Algorithms for computing the Fourier transform on streams can be divided into three classes: The first class computes DFT for sliding windows that overlap. If the sliding window advances for each point in the stream, then the DFT for that window can be computed efficiently using Sliding DFT algorithms [2], [3], [4], [6], [7]. The second class consists of algorithms when the window advances by multiple new points. The DFT of such hopping window can be computed as proposed in [5], [8]. The third class are variants of the FFT

algorithm[1], which is used when windows do not overlap. The methods require a quadratic number of complex multiplications. In this paper, we propose a new algorithm, the *Single Point Fourier Transform* (SPFT), for computing DFT of a stream where each new item in the stream updates a 2D-grid. Each time a 2D-grid is updated its DFT can be computed with a linear number of complex multiplications.

## 2 Background

A visibility value $V = c + di$ at coordinate $(u, v)$ is measured for two antennas at a time point $t_n$. The antennas measure and generate a continuous and infinite stream $S$ of visibilities: $S(ant_a, ant_b, t_i) = [V, (u, v)]$. As a running example we use the following stream of visibilities:

$$S(ant_1, ant_2, t_1) = [4 + 5i, (2, 2)], \quad S(ant_1, ant_3, t_2) = [-6 + 3i, (3, 1)],$$
$$S(ant_2, ant_3, t_3) = [4 + i, (2, 3)], \quad S(ant_1, ant_2, t_4) = [9 - 6i, (1, 1)],$$
$$S(ant_1, ant_2, t_5) = [4 + 3i, (2, 2)], \quad S(ant_1, ant_1, t_6) = [1 - 2i, (3, 0)], \ ...$$

$V(t_n)$ denotes the visibility grid that includes all visibilities from time $t_0$ to time $t_n$. $V_{u,v}$ denotes the cell of visibility grid $V$ at coordinate $(u, v)$. Multiple visibility values that fall into the same cell in grid $V$ are added as illustrated in Table 1. For example, value $(8 + 8i)$ in cell $(2, 2)$ of $V(t_6)$ is the result of adding the visibilities of $S(ant_1, ant_2, t_1)$ and $S(ant_1, ant_2, t_5)$.

| $\mathbf{V(t_2)}$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | $4 + 5i$ | |
| 3 | | $-6 + 3i$ | | |

v → u

| $\mathbf{V(t_6)}$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | $9 - 6i$ | | |
| 2 | | | $8 + 8i$ | $4 + i$ |
| 3 | $1 - 2i$ | $-6 + 3i$ | | |

v → u

**Table 1** Visibility grids generated by stream $S$ at times $t_2$ and $t_6$

$I(t_n)$ denotes the sky image that is computed as the inverse Fourier transform of $V(t_n)$. The cardinalities of the visibility grid and the image are identical. $I_{x,y}$ denotes the cell of the sky image $I$ at coordinate $(x, y)$, $x, y = 0, 1, 2, ..., N - 1$. The discrete Fourier transform (DFT) can be used to compute the value of $I_{x,y}$ at time $t_n$:

$$I_{x,y}(t_n) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} V_{u,v}(t_n) \cdot W^{(ux+vy)}, \qquad W = e^{\frac{i2\pi}{N}} \tag{1}$$

The exponents $W^k$, $k = 0, 1, 2, ..., N - 1$ are the primitive $N^{th}$ roots of unity according to Euler's formula: $W^k = (e^{-\frac{i2\pi}{N}})^k = cos(\frac{2\pi}{N}k) + isin(\frac{2\pi}{N}k)$.

When a new visibility value $S(ant_i, ant_j, t_{n+1}) = [a + bi, (u, v)]$ arrives it is possible to compute $I(t_{n+1})$ incrementally from $I(t_n)$:

$$I_{x,y}(t_{n+1}) = I_{x,y}(t_n) + S(t_{n+1}) \cdot W^{ax+by} \tag{2}$$

$I(t_{n+1})$ can trivially be computed with $N^2$ complex multiplication operations as illustrated in Table 2.

| $\mathbf{I(t_n)}$ | 0 | 1 | | $N-1$ |
|---|---|---|---|---|
| 0 | $I_{0,0}(t_{n-1}) + S(t_n) \cdot W^{0a+0b}$ | $I_{0,1}(t_{n-1}) + S(t_n) \cdot W^{0a+1b}$ | ... | $I_{0,N-1}(t_{n-1}) + S(t_n) \cdot W^{0a+(N-1)b}$ |
| 1 | $I_{1,0}(t_{n-1}) + S(t_n) \cdot W^{1a+0b}$ | $I_{1,1}(t_{n-1}) + S(t_n) \cdot W^{1a+1b}$ | ... | $I_{1,N-1}(t_{n-1}) + S(t_n) \cdot W^{1a+(N-1)b}$ |
| | ... | ... | ... | ... |
| $N-1$ | $I_{N-1,0}(t_{n-1}) + S(t_n) \cdot W^{(N-1)a+0b}$ | $I_{N-1,1}(t_{n-1}) + S(t_n) \cdot W^{(N-1)a+1b}$ | ... | $I_{N-1,N-1}(t_{n-1}) + S(t_n) \cdot W^{(N-1)a+(N-1)b}$ |

y

x

**Table 2** Incrementally computing $I(t_n)$ from $I(t_{n-1})$

## 3 Single Point Fourier Transform

We propose the *Single Point Fourier Transform* (SPFT), a method that reduces the $O(N^2)$ complex multiplications to $O(N)$ complex multiplications. Let $S(ant_i, ant_j, t_{n+1}) = [c + di, (u, v)]$. The element-wise subtraction of visibility grid $V(t_n)$ from $V(t_{n+1})$ yields a $N \times N$ grid $Z(t_{n+1})$ where all elements are zero except value $c + di$ at coordinate $(a, b)$:

$$Z(t_{n+1}) = \begin{cases} S(t_{n+1}) & if a = u, b = v \\ 0 & otherwise \end{cases}$$

The 2D-DFT $I(Z(t_{n+1}))$ can be computed as:

$$I(Z(t_{n+1})) = S(t_{n+1}) \cdot T(a, b) \tag{3}$$

where $T(a, b)$ is a 2D matrix consisting of twiddle factors as follows:

$$T_{x,y}(a, b) = W^{(ax+by)\%N} \tag{4}$$

We denote each row and column of twiddle factor matrix $T(a, b)$ by $R_j$ and $C_j$, respectively. The $k^{th}$ elements of these vectors are denoted by $R_j[k]$ and $C_j[k]$, respectively. The exponents of twiddle factors of the rows are the modulo of multiples of coordinate $b$ with a constant $z$ added, i.e., $(z + (i \times b))\%N$, where $z$ is a multiple of $a$. For each row the multiple can be different. For columns $C_0, C_1, ..., C_{N-1}$ the roles of $a$ and $b$ are switched.

$$R_j[k] = W^{(j \times a + k \times b)\%N}, \qquad C_j[k] = W^{(j \times b + k \times a)\%N} \qquad \forall j, k = 0, 1, 2, ..., N-1$$

The key result is that either each row is a rotation of the first row or each column is a rotation of the first column.

$$R_j = CSHIFT(R_0, (j * nshift)\%N), \qquad C_j = CSHIFT(C_0, (j * nshift)\%N)$$

For coordinate $(a, b)$ and a $2^m \times 2^m$ matrix, there can be a column or row shift or both. There is not a case when neither a row nor column shift is applicable.

## 4 Conclusion and Outlook

We proposed an algorithm to efficiently compute the 2D-DFT of a temporal stream. The SPFT algorithm computes the 2D-DFT iteratively for each update by reusing computations

| Algorithm | # of complex multiplications | GridSize (N × N) | | | | |
|---|---|---|---|---|---|---|
| | | 32 × 32 | 64 × 64 | 128 × 128 | 256 × 256 | 512 × 512 |
| DFT | $4N^2$ | 4,096 | 16,384 | 65,536 | 262,144 | 1,048,576 |
| FFT | $4N^2 \log N$ | 20,480 | 98,304 | 458,752 | 2,097,152 | 9,437,184 |
| SPFT | $4N$ | 128 | 256 | 512 | 1,024 | 2,048 |

**Table 3** Number of complex multiplications required for a single point update

to minimize the number of complex multiplications. Rather than recomputing the Fourier transform for all points of the temporal stream, our approach only computes DFT for the current point in a stream and adds it to the DFT of the previous values. Our approach requires 20-40 times less operations than FFT for different grid sizes.

In the future it would be interesting to investigate techniques to improve the cost for matrix additions, e.g., by distributing the computation of the additions. Further, the batching of observations, which benefits FFT, should be investigated for applications where this is feasible.

## References


1 W. Cochran, J. Cooley, D. Favin, H. Helms, R. Kaenel, W. Lang, G. Maling, D. Nelson, C. Rader, and P. Welch. What is the fast fourier transform? *IEEE Transactions on Audio and Electroacoustics*, 15(2):45–55, June 1967. doi:10.1109/TAU.1967.1161899.

2 K. Duda. Accurate, guaranteed stable, sliding discrete fourier transform [dsp tips tricks]. *IEEE Signal Processing Magazine*, 27(6):124–127, Nov 2010. doi:10.1109/MSP.2010.938088.

3 E. Jacobsen and R. Lyons. The sliding dft. *IEEE Signal Processing Magazine*, 20(2):74–80, March 2003. doi:10.1109/MSP.2003.1184347.

4 C. Park. Fast, accurate, and guaranteed stable sliding discrete fourier transform [sp tips amp;amp;tricks]. *IEEE Signal Processing Magazine*, 32(4):145–156, July 2015. doi:10.1109/MSP.2015.2412144.

5 C. Park and S. Ko. The hopping discrete fourier transform [sp tips amp;amp;tricks]. *IEEE Signal Processing Magazine*, 31(2):135–139, March 2014. doi:10.1109/MSP.2013.2292891.

6 B. G. Sherlock and D. M. Monro. Moving discrete fourier transform. *IEE Proceedings F - Radar and Signal Processing*, 139(4):279–282, Aug 1992. doi:10.1049/ip-f-2.1992.0038.

7 B.G Sherlock. Windowed discrete fourier transform for shifting data. *Signal Processing*, 74(2):169 – 177, 1999. URL: http://www.sciencedirect.com/science/article/pii/S0165168498002096, doi:https://doi.org/10.1016/S0165-1684(98)00209-6.

8 A. Srivastava and V. Karwal. Windowed r-point update algorithm for discrete fourier transform. In *2013 INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING AND COMMUNICATION (ICSC)*, pages 185–190, Dec 2013. doi:10.1109/ICSPCom.2013.6719780.