On the usage of the Sparse Fourier Transform in ultrasound propagation simulation

Ondrej Olsak
iolsak@fit.vutbr.cz
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic

Jiri Jaros
jarosjir@fit.vutbr.cz
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic

ABSTRACT

The Fourier transform is an algorithm for transforming the signal from the space/time domain into the frequency domain. This algorithm is essential for applications like image processing, communication, medicine, differential equations solvers, and many others. In some of these applications, most of the Fourier coefficients are small or equal to zero. This property of the signals is used by the Sparse Fourier transform which estimates significant coefficients of the signal with a lower time complexity than the Fourier transform. The goal of this paper is to evaluate available implementations of the Sparse Fourier transform on a set of benchmarks solving the ultrasound wave propagation in 1D, 2D, and 3D heterogeneous media. The results show that the fastest available implementation in 1D domains is MSFFT, however, it is not possible to use it in our implementation of the 2D Sparse Fourier transform. Thus the AAFFT 0.9 is selected for our implementation of the 2D Sparse Fourier transform as the most stable and acceptably fast implementation. The results on 3D simulation data show, that by using the SpFFT library it is possible to reduce the computation time of the Fourier transform in ultrasound wave propagation simulation.

CCS CONCEPTS

• **Computing methodologies** → **Massively parallel and high-performance simulations**.

KEYWORDS

Fourier transform, Sparse Fourier transform, high performance computing, k-Wave, ultrasound wave propagation

ACM Reference Format:
Ondrej Olsak and Jiri Jaros. 2023. On the usage of the Sparse Fourier Transform in ultrasound propagation simulation. In *2023 the 10th International Conference on Bioinformatics Research and Applications (ICBRA) (ICBRA 2023), September 22–24, 2023, Barcelona, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3632047.3632064

ICBRA 2023, September 22–24, 2023, Barcelona, Spain

ACM ISBN 979-8-4007-0815-2/23/09
https://doi.org/10.1145/3632047.3632064

1 INTRODUCTION

The Fourier transform is a widely used algorithm for transforming signals from time to frequency domain and analyzing their spectral representation. Applications of the Fourier transform take place in multiple areas including medicine applications like magnetic resonance [20], ultrasound imaging [25], solving of differential equations [23], or numerical calculation of derivation [26]. Since the Fourier transform allows calculation of the derivation more precisely than other methods, such as localized finite difference methods [13], thus it makes it suitable for usage in the ultrasound wave propagation simulation.

The time complexity of the Fourier transform algorithm is $O(N^2)$, which is not suitable for long-time series. To reduce the time complexity of the Fourier transform, the Fast Fourier transform algorithm (FFT) [7] was introduced. This algorithm reduces computational time complexity from $O(N^2)$ to $O(N \log N)$, which is a significant improvement against the original algorithm. Although the Fast Fourier transform algorithm significantly reduced the time necessary to compute the Fourier transform, it is not always necessary to compute all Fourier coefficients, especially if from the nature of the problem there are only a few nonzero coefficients in the frequency domain. One example of such a problem is GPS synchronization [11]. The synchronization process produces only a single spike in the time domain. To find this spike the Sparse Inverse Fourier transform was used. Another application using the sparse property of the input signal is a Time-Based Analog-to-Digital converter [29] working for signals with 3% sparse frequency domain.

The sparsity of the signal means, that the signal has only at most k significant coefficients, where $k << N$ for the signal of length N. The rest of the coefficients are usually zero (for exactly sparse signals) or negligible (for approximately sparse signals) representing noise in the processed signal. A good example of a sparse signal in the frequency domain is monochromatic light or a single-frequency transmitter. This signal property is exploited by the Sparse Fourier transform (SFT) algorithms. The Sparse Fourier transform can find significant coefficients with lower than $O(N \log N)$ time complexity. The algorithm usually consists of three main steps frequency bucketization, frequency estimation, and collision resolution [24]. Multiple algorithms are performing the SFT with different techniques and time complexity [18], for example, sFFT 2.0 [12], AAFFT [15], DMSFT [22] and many others which varies in noise robustness, time complexity and accuracy.

In this paper, some of the available implementations of the SFT will be selected and used to compute the Sparse Fourier transform in the last step of the ultrasound wave propagation simulation. In

the simulations, we focus on a spatial pressure distribution so the last simulation step was selected to get more dense input samples. The number of coefficients of the signal in a frequency domain increases during simulation time due to signal decay, absorption, dispersion, and reflection. During this process, physical phenomena like reflection, refraction, and diffraction occur.

In the following sections, the experiments over the 1D signals concerning computation accuracy and execution time will be described. After the evaluation of the 1D implementations, one of them will be chosen and used in an implementation of the 2D SFT to evaluate possible usage of the Sparse Fourier transform on 2D signals with a focus on computation accuracy. At the end the SpFFT [2] (3D SFT library) will be evaluated on 3D simulation samples.

2 PROBLEM SPECIFICATION

Ultrasound simulations are used in many areas from industry to medicine. One of the rapidly emerging and critically deployed is preoperative treatment planning of ultrasound surgery. To run the simulation, the k-Wave toolbox [28] based on the following governing equations is used [28].

$$\begin{aligned} \frac{\partial u}{\partial t} &= -\frac{1}{\rho_0}\nabla p \\ \frac{\partial \rho}{\partial t} &= -\rho_0 \nabla \cdot u - u \cdot \nabla \rho_0 \\ p &= c_0^2(\rho + d \cdot \nabla \rho_0 + \frac{B}{2A}\frac{\rho^2}{\rho_0} - L_\rho) \end{aligned} \tag{1}$$

Equation (1) can be written in a discrete form using the k-space pseudospectral method [27]. This equation is part of the spatial gradient calculations based on the Fourier collocation spectral method.

$$\frac{\partial}{\partial \xi} p^n = \mathcal{F}^{-1}\{ik_\xi \kappa e^{ik_\xi}\mathcal{F}\{p^n\}\} \tag{2}$$

In Eq. (2) for the Cartesian direction $\xi = x$ in R^1, $\xi = x, y$ in R^2, $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the forward and inverse spatial Fourier transform , i is the imaginary unit, k_ξ represents the wave numbers in the ξ direction, and κ is the k-space operator defined as $\kappa = sinc(c_{ref}k\Delta t/2)$, where c_{ref} is a scalar reference sound speed.

There are several approaches to solve mentioned differential equations such as pseudo-spectral method [28], finite-difference time-domain method [14], etc.

The k-Wave toolbox uses the pseudo-spectral method with the Fourier basis function, which means that a significant portion of the simulation is spent on the computation of the Fourier transform. This makes it a suitable tool for reducing simulation time by using the SFT algorithm.

The idea of pseudo-spectral methods is to transform the solution of the differential equation into a sum of a certain basis function. In the finite-difference time-domain methods, the gradient is computed based on the function values at the neighbor points. The more points are used, the more accurate the derivative estimation is. In spectral methods, the solution depends on the entire domain which makes them more accurate than local methods [10]. In k-Wave, the Fourier collocation spectral method is used [28]. In this method the derivative is much more accurate, thus the finer sampling around 3-5 grid points as opposed to 15-20 for finite difference methods can be used. This reduce the memory requirements 100 times in 3D domain. Additionally, thanks to the k-space correction, a relatively long step can be chosen, with a CFL (Courant-Friedrichs-Lewy) number of 0.3. This means that only three steps are needed to move the wave by one grid point.

In the current implementation, the Fast Fourier transform algorithm is used to find the signal's coefficients in the spectral domain. Computation of the ultrasound propagation simulation consists of many simulation time steps requiring 14 FFTs to be computed in each time step, which contributes to 60% of the simulation time spent by computing the Fourier transform [17]. This means that the Fourier transform takes a significant part of the simulation time.

When the transmitter with a single frequency is used, the signal will spread across the medium and new frequencies will accrue during simulation time. Primarily on the edges of different media due to the change of propagation speed and reflection on the edges of two media.

The assumption is that it should be possible to decrease the computation time by using the Sparse Fourier transform algorithm. To answer the question about the possibility of the usage of the SFT, the experiments on the one, two, and three-dimensional domains will be performed and evaluated in the following sections.

3 ONE-DIMENSIONAL SPARSE FOURIER TRANSFORM

At the beginning of this section, freely available one-dimensional SFT algorithm implementations will be shortly introduced followed by the experiments and their evaluation. All of the following implementations have two common input parameters. The length of the signal N and the number of significant coefficients k. This means that number of significant coefficients must be known before the transform execution.

3.1 Ann Arbor Fast Fourier Transform (AAFFT)

The Ann Arbor Fast Fourier Transform algorithm [15] comes in two variants.

AAFFT 0.5, the implementation of the FADFT-1 [8] with the time complexity $O(k^2 \cdot polylog(N))$ and AAFFT 0.9 that is implementation of FADTF-2 [9] with the time complexity $O(k \cdot polylog(N))$. The difference is that FADTF-2 uses the *unequally spaced Fourier transform* meaning multiplication of some $k \times k$ submatrix of the $N \times N$ Fourier matrix by a length-k vector [9].

In experiments, AAFFT 0.9 was used. The implementation relies on twenty user-defined parameters thus, its usage requires a lot of parameter tuning. These parameters affect the accuracy and computation time of the algorithm. It is impossible to run experiments with all combinations of these twenty input parameters. Therefore, the parameters with the highest impact on the execution time and precision were selected and changed during our experiments, the other parameters were set to a default value. The changing parameters affect the number of coefficient estimation cycles, the number of Taylor series coefficients, and the number of input samples used in the estimation phase of the algorithm.

3.2 Gopher Fast Fourier Transform (GFFT)

The Gopher Fast Fourier Transform [5] implementation comes in four variants with different algorithms [5][16] for finding and estimating coefficients of the input signal. All variants take advantage of aliasing phenomena.

Table 1: Time complexity of different GFFT implementations

Implementation name	Runtime complexity
GFFT-det-slow	$O(N \cdot k \cdot \log^2 N)$
GFFT-det-fast	$O(k^2 \cdot \log^4 N)$
GFFT-rand-slow	$O(N \cdot \log N)$
GFFT-rand-fast	$O(k \cdot \log^5 N)$

A detailed explanation of differences between all mentioned variants can be found in [16]. Table 1 shows the time complexity of different GFFT implementations. Tested implementation requires only one input parameter controlling the accuracy of the Monte-Carlo algorithm to be set by the user.

3.3 Discrete Michigan State Fourier Transform algorithm (DMSFT)

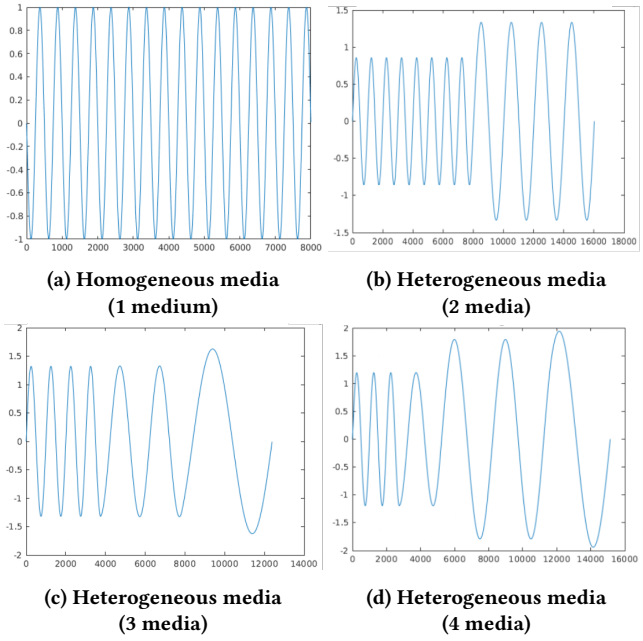
The Discrete Michigan State Fourier Transform algorithm [22] was derived from the previously mentioned GFFT algorithm. It's time complexity is $O(k^2 \cdot \log^{\frac{11}{2}} N)$. This algorithm uses the multiscale error correction to estimate coefficients in noise signals [19]. The implementation of this algorithm requires three user-defined parameters. First gives the number of points used in convolution, second the first main prime number in the sampling scheme, and third controls the number of sets of prime numbers in the sampling scheme.

3.4 Multiscale Sub-linear Time Fourier Transform (MSFFT)

The Multiscale Sub-linear Time Fourier algorithm [6] runs in a time complexity $O(k \cdot \log k \cdot \log \frac{N}{k})$ on average. This implementation has four input parameters. The idea of the MSFFT algorithm is the alike principle of analog-to-digital conversion where a value of the signal can be estimated with high precision by using coarse binary quantization. The authors recommended suitable values for these parameters to achieve a good balance between speed and accuracy.

3.5 Experiments

To evaluate the libraries, the k-Wave toolbox [28] to simulate ultrasound propagation in a 1D grid was used. Simulations with a different number of media were run to get inputs with various numbers of significant coefficients. The transmitter was set to a single frequency to get the simplest simulation. The input signals are shown in Fig. 1. To reduce the amount of noise in the input signal, all coefficients smaller than -50dB were filtered out before the benchmark execution. In real simulations, the threshold of the noise is given by the parameters of the perfectly matched layer (PML) that gives us computation accuracy. The number of media in real simulations is around 30 for segmented data and 256 for data from computed tomography [21].



(a) Homogeneous media (1 medium) **(b) Heterogeneous media (2 media)**

(c) Heterogeneous media (3 media) **(d) Heterogeneous media (4 media)**

Figure 1: Last step of the linear wave propagation simulation performed by the k-Wave toolbox. The X axis represents grid points(distance) and Y axis represents amplitude.

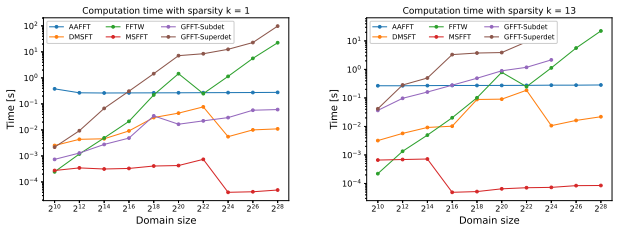
As mentioned before, all implementations require the number of most significant coefficients to search for as an input parameter. To get this value, the Fourier transform was applied to the filtered input signal to get the number of the most significant coefficients k. The number of coefficients in each input signal is the following: one for homogeneous media, 13 for two media, 16 for three media, and 18 for four media.

The field of our interest was the computation time and accuracy of the result in form of L_2 and L_{inf} error that are defined as follow:

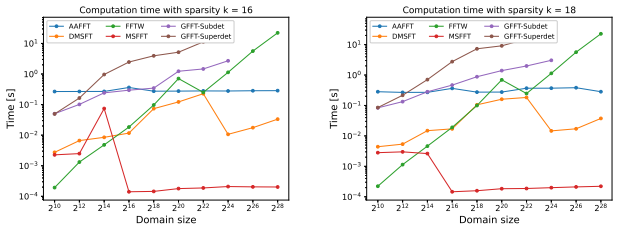
$$L_2 Error = \sqrt{\frac{\sum_{i=0}^{N} |y_i^2 - x_i^2|}{\sum_{i=0}^{N} y_i^2}} \tag{3}$$

$$L_\infty Error = max(|x_i - y_i|) \tag{4}$$

As the reference library, FFTW3 [1], which is a de-facto standard in FFT calculation Fast Fourier transform algorithm was used. The sizes of the input domain were $N = 2^{20}, 2^{22}, 2^{24}, 2^{26}, 2^{28}$ with the input file size from 8MB up to 2048MB. As mentioned before, all these implementations have some parameters, that influence the execution time and accuracy. We run multiple simulations to select the parameters that have the best balance between execution time and accuracy. For each parameter, an interval was selected, and the step by which the parameter is increased within the interval (eg. $I =< 0, 1 >$ and $step = 0.2$ results in a vector of parameter $p = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$). Each implementation was then run with all combinations of these parameters over each input signal. The run with the smallest error and the best execution time was selected as the final value. The table of parameters is attached in appendix A. All simulations were computed on a single compute node of the



(a) Homogeneous media (1 medium) **(b) Heterogeneous media (2 media)**



(c) Heterogeneous media (3 media) **(d) Heterogeneous media (4 media)**

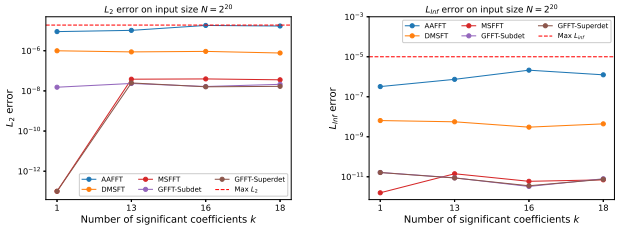
Figure 2: Execution time of SFT implementations on signal of length between 2^x - 2^y with sparsity k = {1, 13, 16, 18}.

Barbora supercomputer [3]. This node is equipped with two Intel Cascade Lake 6240, 2.6GHz, and 192GB of RAM.

Before we look at the results, it is important to mention that all graphs with the results are in the logarithmic scale and the number of coefficients in the signals is extremely low compared to the sizes of the input signal.

The results are shown in Figure 2. First, the MSFFT implementation outperforms all of the other implementations over all given inputs. Second, the AAFFT is the most stable implementation concerning the signal size. With the increasing size of the signal, this implementation's execution time is almost constant, which probably depends on how efficiently the given implementation estimates the coefficients concerning the number of used signal samples. Finally, almost all of the measured implementations outperform FFTW3 on large input domains except GFFT-Superdet. Measured L_2 and L_{Inf} errors of the SFT implementations are shown in Fig. 3. We can see that all implementations have an acceptable level of error (except of the AAFT on the second simulation), which is in the case of ultrasound propagation simulation $L_{Inf} <= 10^{-5}$. All computations were made with double precision values.

We can notice that there are no values for GFFT in the error charts and the execution time measurement is not complete for this library in heterogeneous media. The reason is the initialization process of the GFFT library. To execute the benchmark over the SFT algorithm implementation, it is necessary to load the signal into the structures of the SFT algorithm implementation. The implementation of the GFFT algorithm does not allow an easy way to load the whole input signal because the sampling and other operations are performed during the initialization process. The GFFT source code allows benchmarking where the user selects a number of coefficients and level of noise and GFFT creates only samples necessary for its computation. Thus processing the whole input of this size



(a) L_2 error **(b) L_{inf} error**

Figure 3: L_2 and L_{inf} error of different SFT implementations on the 1D signal with 2^{20} samples

leads to a long initialization process in the case of heterogeneous media so we were not able to measure the results of this library on bigger inputs due to the time-consuming initialization process.

The L_{inf} error of almost all of the evaluated implementations is on the values, that allow their usage in ultrasound propagation simulation. Even if the error of each step were summed up, the simulation L_{inf} error would be lower than 10^{-5} in 1000 simulation time steps. The speedup against FFTW [1] is in the case of MSFFT [6] on the order of 10.000x, and in the case of AAFFT 0.9 [15] 82x for a signal with 4 media and the size of 2^{28}.

4 TWO-DIMENSIONAL SPARSE FOURIER TRANSFORM

The process of computation of 2D SFT is schematically described in Figure 4. The algorithm of the two-dimensional Fourier transform uses a one-dimension Fourier transform in the following way. In the first step, the one-dimensional Fourier transform is applied to each row in the 2D domain. After the first step, each row contains the results of the one-dimensional Fourier transforms. In the second step, the Fourier transform algorithm is called on the results from the first step but in a column direction.

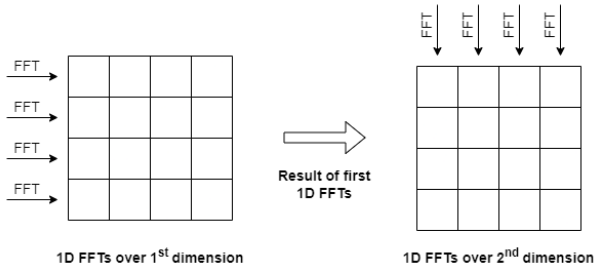
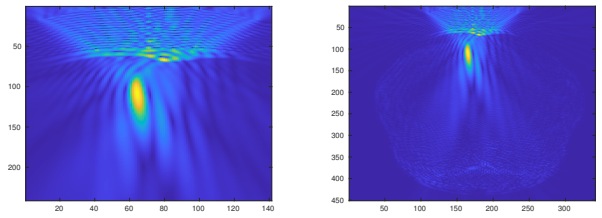


Figure 4: Usage of one-dimensional FFT to create two-dimensional FFT.

For the implementation of the two-dimensional SFT, we needed a 1D SFT implementation, that meets some requirements. The implementation should be able to fill k most significant coefficients with zeros in case the selected k for a given dimension is larger

than the real number of nonzero coefficients. The selected implementation should be able to compute the Fourier transform over all rows and columns without the need to change or modify its settings (input parameters). And the final requirement is that the algorithm implementation allows acceptable integration for usage in the two-dimensional variant of the SFT.



(a) Part of the skull **(b) Full skull**

Figure 5: Last step of ultrasound propagation simulation on the real data 5a focus in part of the head, 5b focus in full head image.

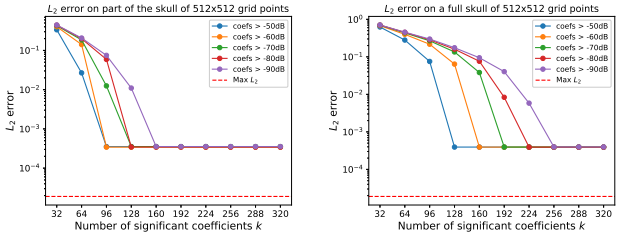
Based on the results from the previous section, we decided to select AAFFT 0.9. This SFT implementation, as the only one from the presented implementations (3), meets all requirements and is suitable for our proof of concept 2D SFT implementation. The reasons why other libraries were not chosen are the following. The GFFT has no suitable implementation for use in two-dimensional SFT (the initialization process is the main issue here). MSFFT is not able to fill the final number of k with zeros and DMSFT requires separate settings for almost every row and column.

To verify the possibility of using the SFT on the real simulation data the 2D implementation of the SFT was used on two real samples of ultrasound wave preparation in a human head. The first one, see Fig. 5a, (denote as SimP_2D) represents the last step of the simulation on the cutout of the human head. The second one, see Fig. 5b, (denote as SimF_2D) is the last step of the simulation in the domain with a full human skull.

We run multiple experiments over each simulation with decreasing number of the most significant coefficients which we looked for (parameter k) and with differently filtered signals where all coefficients that are lower than the given value (-50dB up to -90dB) were filtered out. Fig. 6 shows L_2 error and Fig. 7 shows L_{inf} error for differently filtered input signal.

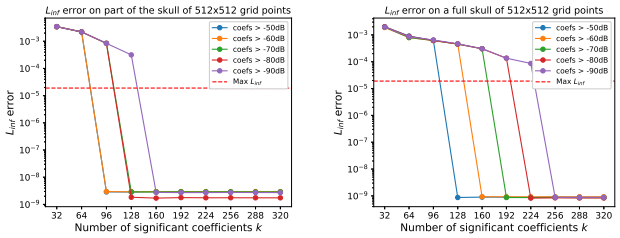
It can be observed that L_2 error does not reach the required value even with an increasing number of coefficients (parameter k). The AAFFT 0.9 was not able to estimate coefficients with higher precision even with different parameter settings. In a simple test simulation in homogeneous media, the L_2 error was reached. On the other hand, the required value of the L_{inf} error is satisfied when k is between 64 - 96 for SimP_2D and 96 - 128 for SimF_2D. It is important to mention that value of the L^2 error depends on the size of the domain, base on it's definition in the Eq. 3.

When looking at Fig. 8, the execution time of our proof of concept 2D SFT implementation is somewhere between 47 to 65 minutes. The AAFFT 0.9 is not a thread safe library, thus it is not possible to perform 1D SFT over rows/columns in parallel. The second cause of such a long execution time is the fact that the number of coefficient estimation iterations had to be increased to achieve better accuracy.



(a) L_2 error for part of the skull **(b) L_2 error for full skull**

Figure 6: L_2 error of the 2D SFT in double precision



(a) L_{inf} error for part of the skull **(b) L_{inf} error for full skull**

Figure 7: L_{inf} error of the 2D SFT in double precision

(a) Part of the skull **(b) Full skull**

Figure 8: Execution time of the 2D SFT implementation

The number of coefficients in the spectral domain is variable during the simulation, where the signal may spread across a media with a different sound speed and density. For the usage of the Sparse Fourier transform in ultrasound propagation simulation, it will be necessary to use some iterative approach to estimate an unknown number of coefficients in each row/column of the media to achieve a two-dimensional Fourier transform with a given accuracy.

5 THREE-DIMENSIONAL SPARSE FOURIER TRANSFORM

For a three-dimensional SFT, we were able to find the SpFFT[2] library. SpFFT implements a three-dimensional Sparse Fourier transform with the support of OpenMP, MPI, CUDA, and ROCm. There is support for both single and double precision implementation.

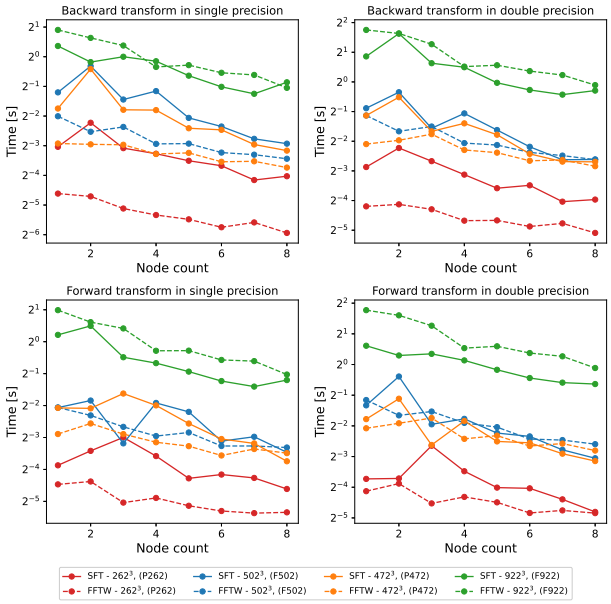


Figure 9: Execution time of the SpFFT for the different number of computation nodes.

Three-dimensional simulation is the main simulation used for treatment planning, thus the experiments were executed over unfiltered real 3D simulation data (last simulation step) to get as close to the real conditions as possible. First, two simulations in the domain with a cutout of the human head were taken. The size of these simulations is 262^3 and 472^3 with input sizes 1GB and 4 GB. Let's denote them as *P262* and *P472* respectively. The second two simulations with the domain containing a full human head were taken. The simulation domain sizes are 502^3 and 922^3 with input sizes 1.25GB and 6.5GB. Let's denote them as *F502* and *F922* respectively. As the reference the FFTW3 [1] library was used. The measurement was executed at Karolina cluster [4] on one up to eight computation nodes, where each of them is equipped with two AMD Zen 2 EPIC 7H12, 2.6GHz, and 256GB of RAM.

The results in Table 2 show that in double precision the SpFFT library meets the required precision unlike for single precision. When we look at the computation speedup in Table 3 for forward transform and in Table 4 for backward transform it can be observed, that in some cases this library is more than two times faster for forward transform and almost two times faster for backward transform than FFTW3 that is currently used in ultrasound wave propagation simulation.

6 CONCLUSION

The performance and accuracy of several SFT algorithm implementations on the last step of one-dimensional ultrasound propagation simulation were measured. After the evaluation of the selected 1D SFT implementations, AAFFT 0.9 was chosen as a suitable 1D SFT implementation for our experimental two-dimensional SFT. In the final section, the SpFFT implementation of the three-dimensional SFT was measured on unfiltered real simulation data.

Table 2: L_2 and L_{inf} error of the 3D SFT implementation in *Single precision* (SP) and *Double precision* (DP) on the real data samples after forward and backward transformation.

	P262	F502	P472	F922
L_2 **error SP**	$7.71e^{-4}$	$1.09e^{-3}$	$7.27e^{-4}$	$8.60e^{-4}$
L_{inf} **error SP**	$6.33e^{-2}$	$1.12e^{-1}$	$8.41e^{-2}$	$1.47e^{-1}$
L_2 **error DP**	$4.08e^{-8}$	$3.20e^{-8}$	$2.91e^{-8}$	$3.62e^{-8}$
L_{inf} **error DP**	$1.60e^{-10}$	$1.46e^{-18}$	$1.89e^{-18}$	$3.12e^{-18}$

Table 3: Forward transformation speedup of the 3D SFT against 3D FFTW on the real data samples.

Node cnt	1	2	3	4	5	6	7	8
P262	0.76	0.89	0.27	0.56	0.72	0.57	0.78	0.97
F502	1.13	0.42	1.33	0.91	1.14	0.94	1.25	1.37
P472	0.81	0.57	1.84	0.67	1.14	0.93	1.26	1.27
F922	2.24	2.48	1.89	1.32	1.70	1.76	1.81	1.44

Table 4: Backward transformation speedup of the 3D SFT against 3D FFTW on the real data samples.

Node cnt	1	2	3	4	5	6	7	8
P262	0.40	0.27	0.32	0.34	0.47	0.38	0.60	0.46
F502	0.84	0.40	1.04	0.50	0.70	0.88	1.10	1.00
P472	0.51	0.37	0.94	0.54	0.66	0.85	1.04	0.90
F922	1.86	1.00	1.56	1.02	1.51	1.55	1.58	1.14

The results have shown that MSFFT is in given cases the fastest implementation with time around 10^{-4} seconds and L_{inf} error around 10^{-11}. Unfortunately, MSFFT was not able to fill k coefficients with zeros which eliminates it from being used in 2D SFT implementation. The results have also shown that the accuracy and computation time of each library is highly dependent on the values of the input parameters.

In the 2D variant of SFT, the sparsity of the signal in each row/column of the 2D domain is different, thus the results with differently filtered input signals were provided. The measurements have shown the 2D SFT can be used on the real simulation data while reaching L_{inf} error below 10^{-10}. In the case of the 3D input data, the results show that SpFFT is more than two times faster for forward transform and almost two times faster for backward transform while holding the required computation precision.

The results indicate that using the Sparse Fourier transform in k-Wave [28] ultrasound propagation simulation should be possible while holding a given computation accuracy. This may lead to lower simulation time, thus reducing the time of the treatment planning.

However, there are some goals to achieve. First, remove the dependency of the 1D SFT on the knowledge of the number of coefficients in the spectral domain of the input signal and expensive parameter tuning before transform execution. Second, the 2D SFT implementation using existing 1D SFT is currently fully sequential, which leads to a long computation time that does not allow us to run full simulations with the use of this 2D SFT implementation.

This paper provides an answer which concludes with the statement that it is theoretically possible to use SFT in the ultrasound propagation simulation. This paper should be the starting point for the future acceleration of the ultrasound wave propagation simulation in the k-Wave toolbox.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254). This project has received funding from the European Unions Horizon Europe research and innovation programme under grant agreement No 101071008. This work was supported by Brno University of Technology under project number FIT-S-23-8141.

REFERENCES

[1] 2014. FFTW home page. https://www.fftw.org/
[2] 2019. SpFFT Documentation. https://spfft.readthedocs.io/en/latest/
[3] 2023. IT4I technical information of the Barbora supercomputer page. https://www.it4i.cz/en/infrastructure/barbora
[4] 2023. IT4I technical information of the Karolina supercomputer page. https://www.it4i.cz/en/infrastructure/karolina
[5] I. Ben Segal and M. A. Iwen. 2010. Signal Approximation via the Gopher Fast Fourier Transform. *AIP Conference Proceedings* 1301, 1 (2010), 494–502. https://doi.org/10.1063/1.3526650 arXiv:https://aip.scitation.org/doi/pdf/10.1063/1.3526650
[6] Andrew Christlieb, David Lawlor, and Yang Wang. 2013. A Multiscale Sub-linear Time Fourier Algorithm for Noisy Data. https://doi.org/10.48550/ARXIV.1304.4517
[7] James W. Cooley and John W. Tukey. 1965. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comp.* 19, 90 (1965), 297–301. http://www.jstor.org/stable/2003354
[8] Anna Gilbert, Sudipto Guha, Piotr Indyk, Senthilmurugan Muthukrishnan, and Martin Strauss. 2002. Near-optimal sparse Fourier representations via sampling. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, 152–161. https://doi.org/10.1145/509907.509933
[9] Anna Gilbert, Senthilmurugan Muthukrishnan, and Martin Strauss. 2004. Improved Time Bounds for Near-Optimal Sparse Fourier Representations. *Proceedings of SPIE - The International Society for Optical Engineering* 5914 (01 2004). https://doi.org/10.1117/12.615931
[10] S. Gottlieb and D. Gottlieb. 2009. Spectral methods. *Scholarpedia* 4, 9 (2009), 7504. https://doi.org/10.4249/scholarpedia.7504 revision #91796.
[11] Haitham Hassanieh, Fadel Adib, Dina Katabi, and Piotr Indyk. 2012. Faster GPS via the Sparse Fourier Transform. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking* (Istanbul, Turkey) *(Mobicom '12)*. Association for Computing Machinery, New York, NY, USA, 353–364. https://doi.org/10.1145/2348543.2348587
[12] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. 2012. Simple and Practical Algorithm for Sparse Fourier Transform. In *Proceedings of the 2012 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1183–1194. https://doi.org/10.1137/1.9781611973099.93 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611973099.93
[13] A. Hosokawa. 2005. Simulation of ultrasound propagation through bovine cancellous bone using elastic and Biot's finite-difference time-domain methods. *The Journal of the Acoustical Society of America* 118, 3 (2005), 1782–1789. https://doi.org/10.1121/1.2000767 arXiv:https://doi.org/10.1121/1.2000767
[14] A. Hosokawa. 2005. Simulation of ultrasound propagation through bovine cancellous bone using elastic and Biot's finite-difference time-domain methods. *The Journal of the Acoustical Society of America* 118, 3 (09 2005), 1782–1789. https://doi.org/10.1121/1.2000767 arXiv:https://pubs.aip.org/asa/jasa/article-pdf/118/3/1782/15275850/1782_1_online.pdf
[15] Mark Iwen, Anna Gilbert, and And Strauss. 2007. Empirical evaluation of a sub-linear time sparse DFT algorithm. *Communications in Mathematical Sciences* 5 (01 2007), 981–998. https://doi.org/10.4310/CMS.2007.v5.n4.a13
[16] M. A. Iwen. 2010. Improved Approximation Guarantees for Sublinear-Time Fourier Algorithms. https://doi.org/10.48550/ARXIV.1010.0014
[17] Jiri Jaros, Bradley Treeby, and Alistair Rendell. 2012. Use of multiple GPUs on shared memory multiprocessors for ultrasound propagation simulations. *Conferences in Research and Practice in Information Technology Series* 127, 43–52.
[18] Zhikang Jiang, Jie Chen, and Bin Li. 2021. Empirical Evaluation of Typical Sparse Fast Fourier Transform Algorithms. *IEEE Access* 9 (2021), 97100–97119. https://doi.org/10.1109/ACCESS.2021.3095071
[19] Zhikang Jiang, Jie Chen, and Bin Li. 2021. Empirical Evaluation of Typical Sparse Fast Fourier Transform Algorithms. *IEEE Access* 9 (2021), 97100–97119. https://doi.org/10.1109/ACCESS.2021.3095071
[20] Michael Lustig, David L. Donoho, Juan M. Santos, and John M. Pauly. 2008. Compressed Sensing MRI. *IEEE Signal Processing Magazine* 25, 2 (2008), 72–82. https://doi.org/10.1109/MSP.2007.914728
[21] Jackson W. Massey and Ali E. Yilmaz. 2016. AustinMan and AustinWoman: High-fidelity, anatomical voxel models developed from the VHP color images. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 3346–3349. https://doi.org/10.1109/EMBC.2016.7591444
[22] Sami Merhi, Ruochuan Zhang, Mark A. Iwen, and Andrew Christlieb. 2017. A New Class of Fully Discrete Sparse Fourier Transforms: Faster Stable Implementations with Guarantees. https://doi.org/10.48550/ARXIV.1706.02740
[23] D.H. Mugler and R.A. Scott. 1988. Fast fourier transform method for partial differential equations, case study: The 2-D diffusion equation. *Computers & Mathematics with Applications* 16, 3 (1988), 221–228. https://doi.org/10.1016/0898-1221(88)90182-4
[24] Elias Rajaby and Sayed Sayedi. 2022. A structured review of sparse fast Fourier transform algorithms. *Digital Signal Processing* 123 (01 2022), 103403. https://doi.org/10.1016/j.dsp.2022.103403
[25] Chaojun Shou, Xiaoyu Chen, Hao-Li Liu, and Po-Hsiang Tsui. 2016. Using Short-Time Fourier Transform to Ultrasound Signals for Fatty Liver Detection. *International Journal of Signal Processing Systems* (08 2016), 300–303. https://doi.org/10.18178/ijsps.4.4.300-303
[26] Sunaina, Mansi Butola, and Kedar Khare. 2018. Calculating numerical derivatives using Fourier transform: some pitfalls and how to avoid them. *European Journal of Physics* 39, 6 (10 2018), 065806. https://doi.org/10.1088/1361-6404/aadda6
[27] Bradley Treeby, Ben Cox, and Jiri Jaros. 2016. k-Wave A MATLAB toolbox for the time domain simulation of acoustic wave fields User Manual. (2016). http://www.k-wave.org/manual/k-wave_user_manual_1.1.pdf
[28] E. Bradley Treeby, Jiri Jaros, P. Alistair Rendell, and T. Ben Cox. 2012. Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using a k-space pseudospectral method. *Journal of the Acoustical Society of America* 131, 6 (2012), 4324–4336. https://doi.org/10.1121/1.4712021
[29] Praveen K. Yenduri, Aaron Z. Rocca, Aswin S. Rao, Shahrzad Naraghi, Michael P. Flynn, and Anna C. Gilbert. 2012. A Low-Power Compressive Sampling Time-Based Analog-to-Digital Converter. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 2, 3 (2012), 502–515. https://doi.org/10.1109/JETCAS.2012.2221832

A BENCHMARK PARAMETERS

Table 5: Table of parameters for 1D SFT libraries

AAFFT 0.9		
Parameter name	**Interval**	**Step**
Num_FreqID_CoefEst_Iterations	<5, 11>	2
KShattering_Sample_Points	<64, 128>	64
FCE_Sample_Points	<128, 256>	128
Norm_Estimation	<5, 11>	2
Max_FCE_Medians	<5, 11>	2
FFCE_Iterations	<5, 11>	2
DMSFT		
Parameter name	**Interval**	**Step**
first_main_prime	<1, 11>	1
number_of_sample_sets	<1, 11>	1
kappa	<4, 6>	1
GFFT		
Parameter name	**Interval**	**Step**
monteCarloAlgAccuracy	<0.5, 2.0>	0.2
MSFFT		
Parameter name	**Value**	
c1	2	
c_sigma	6	
alias_frac	0.25	
beta	2.5	