EFFICIENT ACOUSTIC MODELLING OF LARGE SPACES USING FINITE DIFFERENCE METHODS

S Durbridge Bowers & Wilkins, West Sussex, UK
AJ Hill Department of Electronics, Computing & Mathematics, University of Derby, UK

1 INTRODUCTION

Improvements in the flexibility, accuracy and performance of acoustic simulation tools could help to make predictions, low-cost exploration (rapid prototyping) and system design workflows easier, faster and more intuitive. Time domain numerical methods used for performing acoustic simulation provide useful visual information (Figure 1), as well as reasonably accurate measurement data. These are often easily scaled and modified to handle a wide variety of simulations, where other wave based modelling methods such as Finite Element and Boundary Element Methods may be more difficult to scale in size and complexity due to computational limitations.

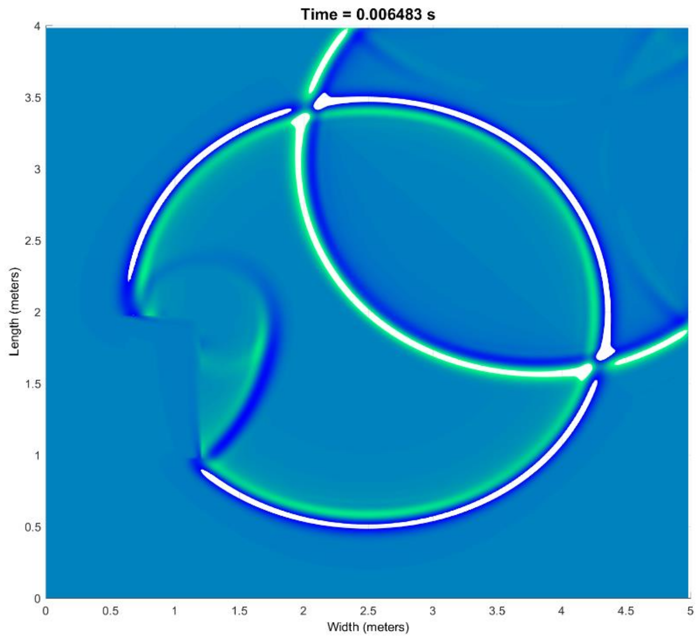


Figure 1 Example visual output of an FDTD simulation of reflection behavior with two sound sources and a single obstacle in an almost anechoic domain [1]

One of the early proponents of work on time domain numerical methods for acoustic simulation was Bootledooren [2]; whose work involved porting the finite difference time domain (FDTD) and finite volume time domain (FVTD) methods from electromagnetics to acoustics, to use in low frequency acoustic simulations. This work has been followed on by researchers such as Murphy [3], Bilbao [4], Savioja [5] and Hamilton [6] to expand and improve potential use of these methods.

Despite a mature body of supporting work, high frequency and large domain simulations using finite difference methods are still relatively uncommon. Previous work focused on combining finite difference and ray based methods to simulate large domains [7], but few commercial products have utilized this research. In his doctoral thesis [8], one of the authors presented a simple and effective implementation of the FDTD method for low frequency modelling, which was the basis for the work presented in this paper.

Following key work such as that by Trefethen [9], a number of wave modelling projects have begun to explore the Fourier pseudo-spectral time domain method (PSTD), most notably the OpenPSTD project from Eindhoven University of Technology [10]. Caunce and Angus [11] recognised the limitations of implementing the FDTD method on general purpose graphical processing units (GPGPU) that can be used to improve the speed of FDTD solving and introduced potential improvements in execution speed by using spectral differentiation on a GPGPU with the Fourier PSTD method. Early work by Caunce and Angus was the basis for the PSTD solver in this study.

In the field of microcontroller development, Doerr [12]–[14] produced work on the sparse finite difference time domain method for electromagnetic simulation. PIC microcontrollers are often modelled as vastly large electromagnetic simulations of networks of channels that take large computational resources, resulting in long simulation times. Doerr suggested that a large proportion of the microcontroller domain is made of a dielectric material and is not necessary for the purpose of the simulation; it should therefore be possible compute only parts of the domain around electromagnetic waves, thus reducing computation time. The sparse finite difference time domain method (SFDTD) presented by Doerr essentially uses a moving window method to reduce the size of the portion of the domain being solved at any given time step within the simulation.

Methods such as FDTD present benefits for low frequency simulation over other simulation methods such as ray based and direct calculation, because characteristics such as modal behaviour is accurately accounted for within the model. This is due to solving the acoustic wave equation in second order partial differential form; ray based methods assume planar radiation of sound waves and ignore the radial propagation of the systems being modelled. Regardless of the time domain method used for solving a simulation, it should be possible to set up and solve a simulation in reasonable time without requiring specialist computing equipment.

The aim of this work is to explore potential improvements in execution speed of the PSTD and SFDTD methods over the FDTD method. In the following section of this paper, a series of simulation test cases are described. Following this, the results of the acoustic output and the execution speed of the simulation methods are compared. Finally, the execution speed profile of each method is reviewed, highlighting where the speed bottlenecks occur in each method.

1.1 Background information

Large scale and high frequency simulations are often difficult to perform using time domain methods, not only because of the large computational resources required to perform the simulations. The time required to perform these simulations can be severely limiting. This is due to the requirement to equally represent discretisation of the entire domain and various conditions for accuracy and stability must be met when solving partial differential equations numerically. These conditions can be difficult to overcome, but if it is possible to reduce execution time (and ideally memory requirements) of a simulation, using time domain numerical methods could become more accessible for less academic users such as sound engineers, loudspeaker designers and undergraduate students.

The basic FDTD method for solving acoustic models involves representing an acoustic system, such as a room, as a set of interwoven grids of sound pressure and particle velocity within a fully discretised system (Figure 2). The necessary distance between the points in the system is defined by the stability of the equations being solved, the physical properties of the system and the highest frequency of interest. A wave equation is split up into two reciprocating components, using velocity points to calculate surrounding pressure points, and pressure points to calculate surrounding velocity points. This is performed in a leapfrog fashion in time, the necessary length of the step also being determined by the highest frequency of interest and the stability or the solving method.

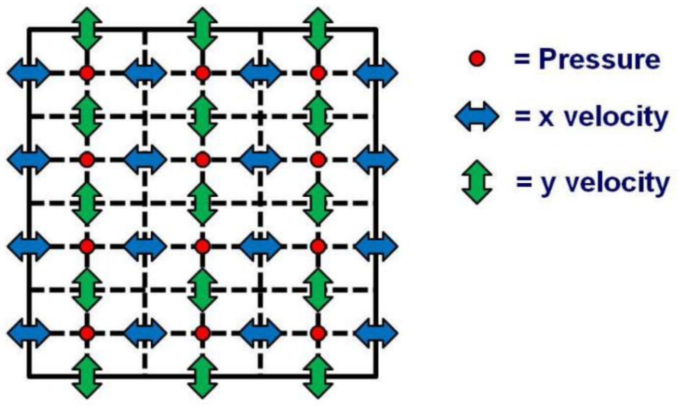


Figure 2 Generic 2D FDTD stencil [8]

The Courant-Freidrichs-Lewey (CFL) condition [15] dictates the minimum time and spatial steps required for a convergent solution when using a time domain method to solve a set of partial differential equations (PDE). The equation for Courant number in the one-dimensional case is:

$$C = \frac{|u|\Delta t}{\Delta x} \leq C_{max} \quad (1)$$

The maximum value of C_{max} is determined by the stability of the method being used to solve the PDE, and for a simple explicit FDTD simulation typically equals 1. Following Nyquist sampling theorem, the maximum spatial step in an FDTD simulation is $\lambda/2$, tough anecdotal discussion and some literature suggests between $\lambda/6$ and $\lambda/10$ is a better target range for a spatial step, Δx. Simulating a 60 m x 40 m x 30 m arena up to a frequency of 500 Hz may require 48,326,250 points in the pressure and each velocity matrix. Adjusting Eq. 1 for the maximum time step in a 3D simulations gives [16]:

$$\Delta t \leq \frac{1}{c}\sqrt{\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}} = 0.0442s \quad (2)$$

This time step equates to 92 steps/second in the simulation, resulting in matrices of at least 1.93GB when using the native double precision floating point arithmetic of Matlab. When considering a 3 dimensional arena simulation 8GB of memory could be required for the four matrices which describe the domain's pressure and velocity. It can be difficult to perform large simulations up to high frequencies, as the amount of memory required to perform a simulation can quickly become greater than that available in non-specialist computer systems. Figure 3 shows the size of each matrix required to perform the simulation for maximum frequencies ranging from 500 Hz to 20 kHz.

Another fundamental problem with FDTD is the requirement to constantly perform non-contiguous memory accesses to perform calculations. Computer memory access (particularly in the CPU cache) is optimised for contiguous accesses in one direction. FDTD can require the system to access memory in an orthogonal direction to optimum around 50% of the time and also requires the system to index into two large blocks of memory simultaneously.

Two similar methods to FDTD that may execute faster for large domains are PSTD and SFDTD. The PSTD follows a similar form to FDTD in most respects. The primary difference is that differentiation in PSTD is performed in the frequency domain. Each matrix is multiplied by the impulse response of an ideal differentiator in the frequency domain, before being used to calculate new values of the reciprocating field. While this method has the potential to be much faster than FDTD (by leveraging the speed of optimised memory access and discrete Fourier transforms), it requires a perfectly matched layer (PML) to overcome Gibbs phenomenon and can suffer from aliasing due to the non-periodic nature of the system being simulated [17]. A PML is a boundary layer surrounding the computational domain that is non-reflective at any frequency or angle of incidence. The PML act as a vacuum and absorbs all energy that enters the boundary layer [18].

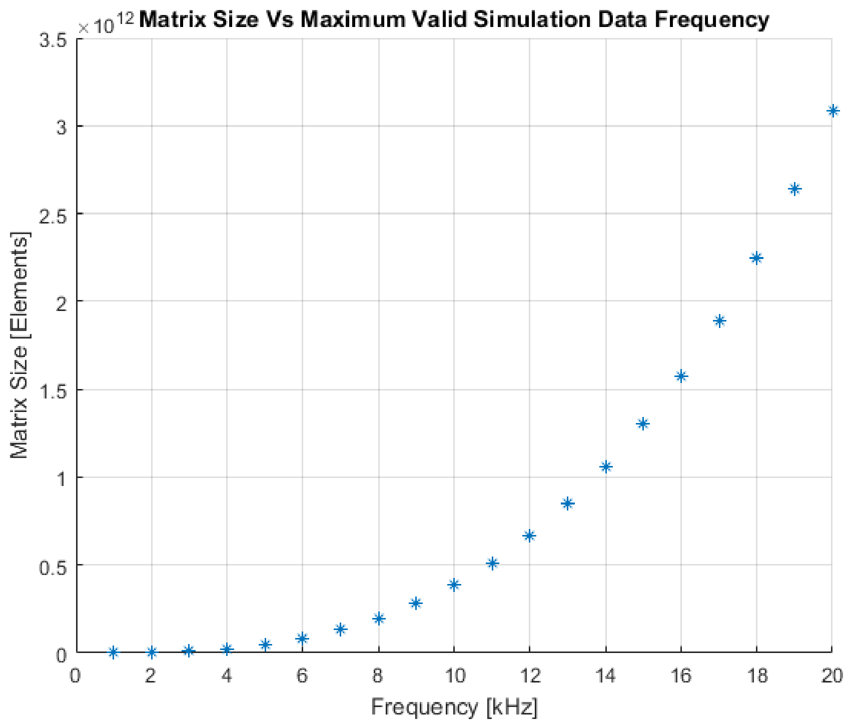


Figure 3 Domain matrix size vs frequency for an FDTD simulation of a 60 m x 40 m x 30 m arena

SFDTD involves windowing portions of the domain that are above a given threshold. This window is used as a guide allowing for just the necessary portions of the domain to be computed. This method is very much in early development in acoustics and previously published literature is scarce. As such, a robust and well validated implementation of SFDTD in acoustics has yet to be reported. Furthermore, the method may only be useful for speed improvements before the level of the diffuse field is relatively high (i.e. when strong reflections are propagating across the domain). This method would ideally use a high order FDTD topology that doesn't suffer as distinctly from numeric dispersion as the 2nd order approach [19, 20].

2 ACOUSTIC MODEL IMPLEMENTATION

In order to test the execution speed of FDTD, PSTD and SFDTD, all three methods were implemented as functions in Matlab. Code development and speed testing was executed on a PC with the following characteristics:

• **Operating System**:	Windows 10
• **CPU:**	i5 4960k Overclocked to 4.5GHz and 1.227V
• **RAM:**	16GB DDR3 ram at 3875 MHz
• **Motherboard:**	Asus Gryphon Armour Edition with Z97 chipset
• **GPU:**	Nvidia GTX 1070

Th computer used standard consumer grade components and was configured using inbuilt automatic tools, thus requiring little specialist configuration knowledge.

Initially FDTD was implemented as a function based on previous work by one of the authors [8]. A 2D version and a 3D version were implemented. The differentiation in FDTD is performed by indexing discrete points of pressure and velocity matrices and calculating updated local values of each based on the previous and surrounding values of the related variable at each point in the domain. Following this, a test was executed to check that a stimulus is propagated across the domain without significant distortion. The modelled domain was a 5 m x 4 m x 3 m rectangle and had partially absorbing boundaries with a frequency-dependent absorption coefficient of 0.45. The maximum analysis frequency was 5 kHz. Stimuli used were three sets of 10 cycles of 1 kHz windowed tone burst, with a rest period of 3 times the length of each burst. In this case, each series of three bursts lasted for 0.1 s, following which there was 0.1 s of silence to allow for decay of the reverberation. The single source was positioned 1 m away from a corner of the domain. Five locations of the domain (near the corners and the centre) were measured. Figure 4 shows the normalised source and receiver signals in the time and frequency domains.

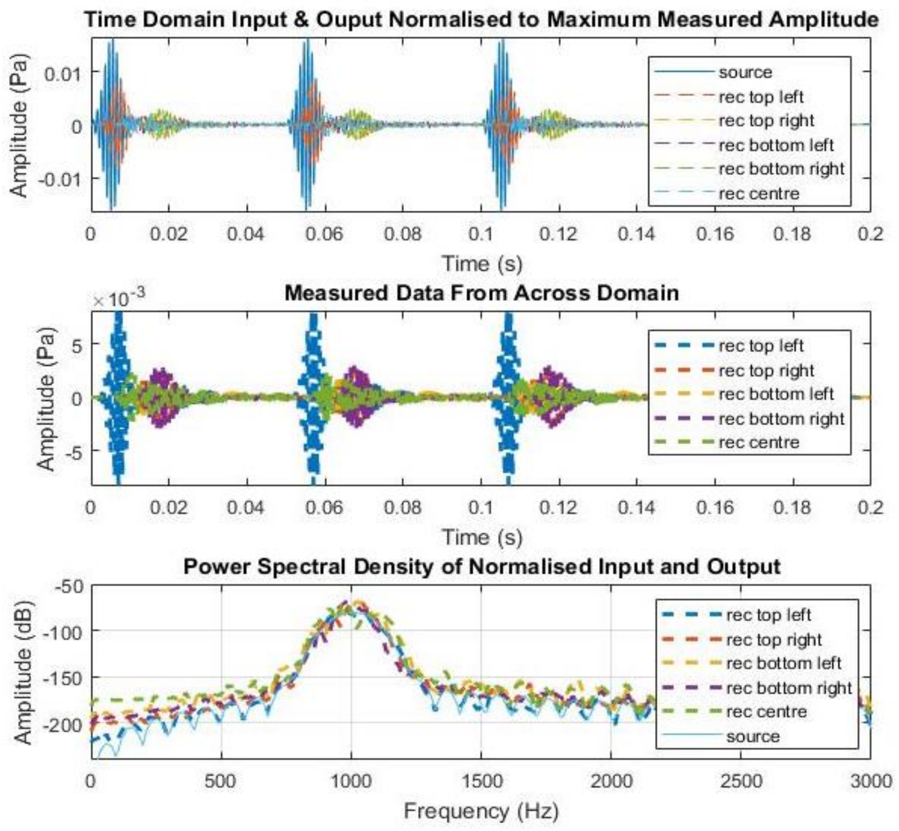
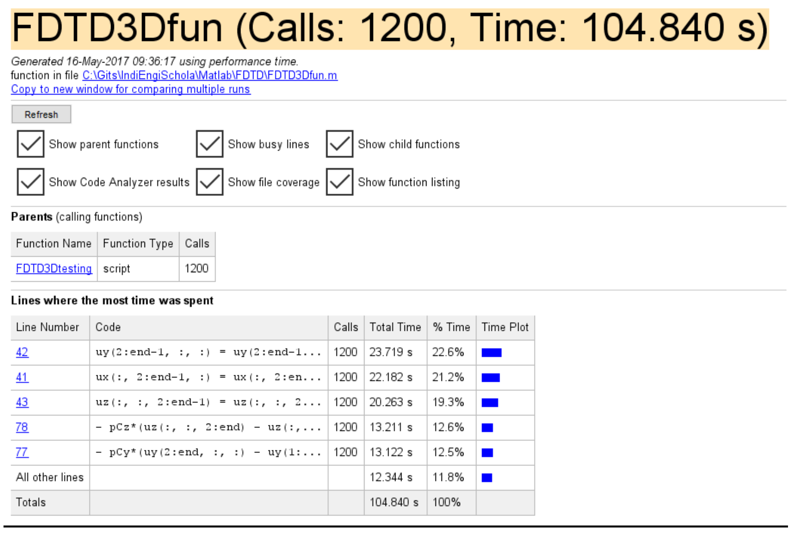


Figure 4 Response of a FDTD simulation showing: stimulus vs measured signals (normalized) (top), time domain response at different positions in the domain (middle) and smoothed frequency domain representation of the measurements at each position in the domain (bottom)

The output shows that the modelled frequency across the domain is the same as the stimulus and no high amplitude oscillatory components are present. The time domain behaviour of the simulation appears to show the expected propagation delay between measurement points, with decay that would suggest the simulation is convergent.

Using the inbuilt code profiler tool in Matlab it was possible to analyse the performance of this FDTD implementation to determine where the processing bottlenecks are. Figure 5 shows the Matlab profiler output of analysis of the 3D FDTD model. The slowest parts of this implementation are where differentiation is occurring (i.e. where the system is having to perform multiple memory accesses to separate large matrices). Managing or reducing these accesses may help improve efficiency.



FDTD3Dfun (Calls: 1200, Time: 104.840 s)

Generated 16-May-2017 09:36:17 using performance time.
function in file C:\Gits\IndiEngiSchola\Matlab\FDTD\FDTD3Dfun.m
Copy to new window for comparing multiple runs

Refresh

✓ Show parent functions ✓ Show busy lines ✓ Show child functions
✓ Show Code Analyzer results ✓ Show file coverage ✓ Show function listing

Parents (calling functions)

Function Name	Function Type	Calls
FDTD3Dtesting	script	1200

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
42	`uy(2:end-1, :, :) = uy(2:end-1...`	1200	23.719 s	22.6%	🟦
41	`ux(:, 2:end-1, :) = ux(:, 2:en...`	1200	22.182 s	21.2%	🟦
43	`uz(:, :, 2:end-1) = uz(:, :, 2...`	1200	20.263 s	19.3%	🟦
78	`- pCz*(uz(:, :, 2:end) - uz(:,...`	1200	13.211 s	12.6%	🟦
77	`- pCy*(uy(2:end, :, :) - uy(1:...`	1200	13.122 s	12.5%	🟦
All other lines			12.344 s	11.8%	🟦
Totals			104.840 s	100%	

Figure 5 3D FDTD execution code profiling

PSTD was implemented as a set of Matlab functions in a similar way to FDTD and was based on the work by Caunce & Angus [11]. To implement partially absorbing boundary conditions, work by Spa *et al* [21] was utilized. The differentiation in PSTD is handled by performing a discrete Fourier transform on one dimension of the domain and multiplying the frequency domain spatial data with the impulse response of an ideal differentiator (Figure 6). The inverse discrete Fourier transform of the differentiated spatial domain data is then used to calculate updated values of the reciprocating field. The differentiation is performed singularly in all spatial dimensions of interest. This method of differentiation may be preferential to FDTD, because the differentiation for calculating any one point includes differentiation of all points in the domain that are linearly coupled.

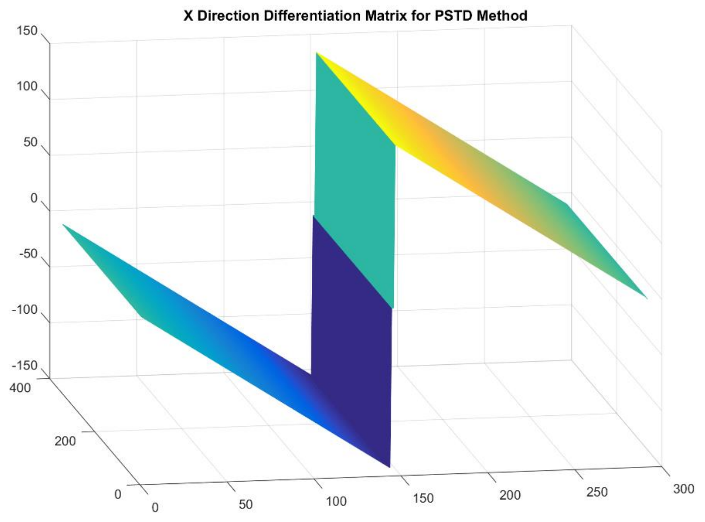


Figure 6 Differentiation matrix for 2D PSTD on the real axis, differentiating in the x-dimension

This differentiation approach not only increases accuracy, but the use of optimised libraries for the Fourier transform and vectorisation can be leveraged by the compiler to increase the speed of computations for differentiation. The same simulation as described for FDTD was used with PSTD, with a maximum frequency of 5 kHz instead of the 2 kHz of used for the FDTD simulation. This sampling rate overhead was chosen due to aliasing occurring during early testing and development of the PSTD implementation. Figure 7 shows the PSTD output with the third plot adjusted for the higher maximum frequency.

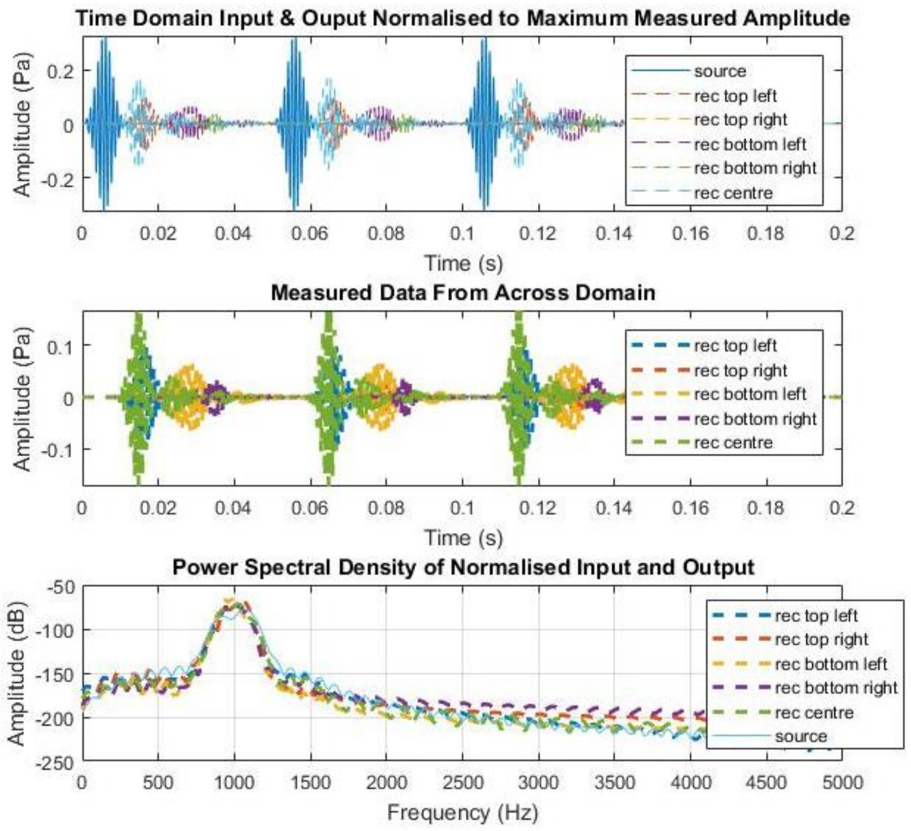


Figure 7 Response of a PSTD simulation showing: stimulus vs measured signals (normalized) (top), time domain response at different positions in the domain (middle) and smoothed frequency domain representation of the measurements at each position in the domain (bottom)

The frequency domain response shows a centre frequency at 1 kHz, the same as the stimulus. The width and shape of the window is similar to that from the FDTD simulation, but with a spectral tilt leaning toward low pass behaviour. The rate at which the propagating wave reached measurement points appears to be different to the results from the FDTD simulation, suggesting that the source location may be different between the two simulations. Though this is likely to be true due to the grid point quantisation difference between the two implementations, further analysis of PSTD may be required to achieve comparable results to the FDTD method. Although further work could be undertaken to diagnose this discrepancy, the overall performance of the system is acceptable enough to use this algorithm for speed testing.

SFDTD was implemented based on the previous FDTD approach along with inspiration from the work of Doerr [12]. Doerr's work in computational electromagnetics uses a list-based method for handling window shape and position, which may not be appropriate for an elastic wave system where a diffusely fluctuating field is desirable. The SFDTD approach in this study was to create a normalised indexing window based on points above a given threshold of absolute pressure within the domain.

The normalised shape of the domain (around a threshold) was smoothed using a Gaussian image filtering technique to ensure that the window's surrounding points of significant pressure are also used within the differentiation (Figure 8). This allows wave fronts to propagate naturally across the domain unimpeded by the window. The window is used to restrict the number of points within the domain that are calculated. Due to time constraints with this study, threshold optimization wasn't performed and a value of 40 dB was used throughout. Further work must be undertaken to determine ideal smoothing methods and threshold values. The output of the SFDTD model are given in Figure 9.

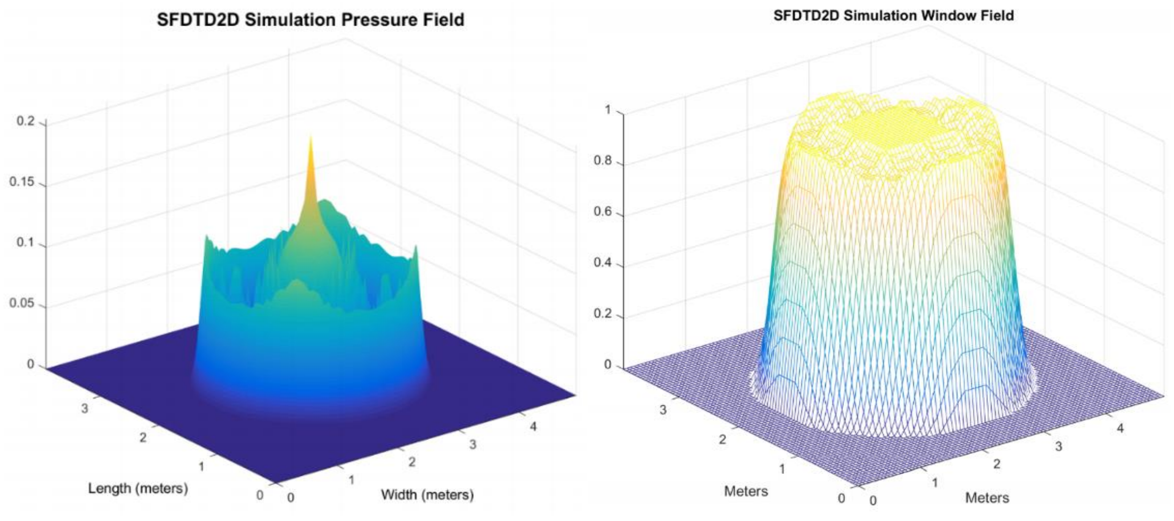


Figure 8 SFDTD 2D model: example travelling stimulus (left) and windowed domain (right)

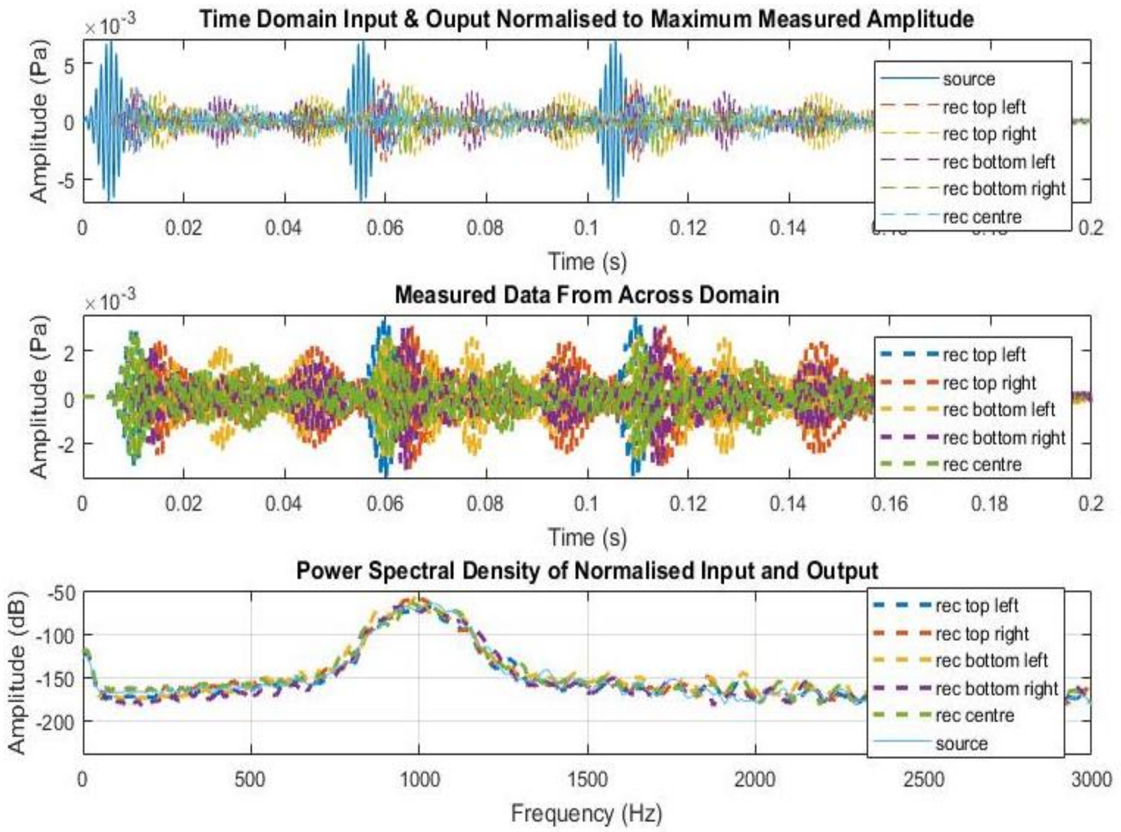


Figure 9 Response of a SFDTD simulation showing: stimulus vs measured signals (normalized) (top), time domain response at different positions in the domain (middle) and smoothed frequency domain representation of the measurements at each position in the domain (bottom)

The centre frequency of the received signals is centred at 1 kHz which is the same as the stimulus. The frequency domain analysis of the measured signals highlights a potential DC offset at all measurement points. This may be caused by the behaviour of the soft source excitation method, as highlighted by Murphy *et al* [3] or a change in the domain impedance caused by use of the window. Also, the amplitude of the first stimulus at the top left measurement point is significantly lower than in the other simulation methods. This may be due to the expanding shape of the initial burst, which could be truncating the numerical dispersion and small fluctuations that are sufficiently far from wave fronts or areas of energy maxima. This behaviour may be a trade-off for faster computation in the early stages of a simulation. Further investigation is required, though. Figure 10 shows the computation time for a set of 2D simulations using the SFDTD method.

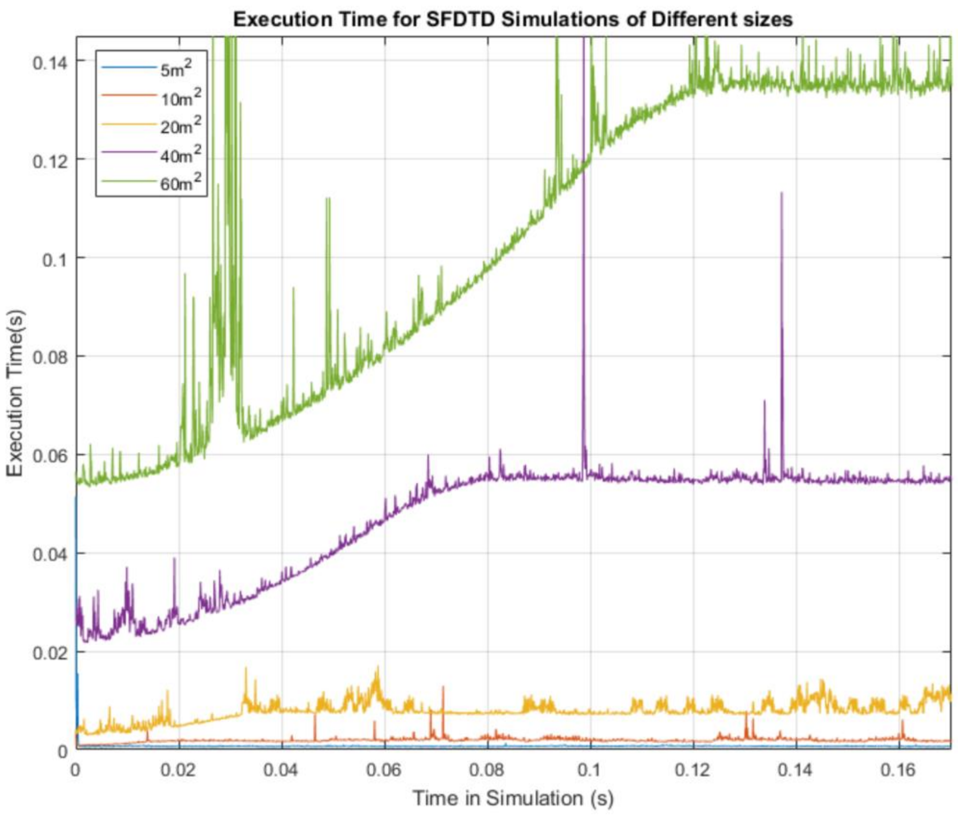


Figure 10 2D SFDTD execution time per time step for a range of domain sizes

This figure highlights the behaviour of the SFDTD method, reducing early computation times before a steady state diffuse field is reached. Using this method to reduce computation time may be appropriate when calculating the early reflection behaviour of a large room, when the wave fronts that are propagating are distinct compared to the size of the domain, but doesn't offer any advantages over standard FDTD when examining a diffuse field.

3 EXECUTION SPEED COMPARISION

To examine the execution speed, each approach was used to model an increasingly-large rectangular 3D domain. The execution speed for the first 2000 time steps of each method was measured using the inbuilt Tic/Toc function in Matlab. A set of 5 domain sizes were (Table 1).

Dimension (m^3)	FDTD Cells	SFDTDCells	PSTD Cells
5	121104	121104	24025
10	483025	483025	62500
20	1932100	1932100	192721
40	7722841	7722841	667489
60	30880249	30880249	2474329

Table 1 Set of domain sizes and domain cells for each time domain method

These domain sizes were chosen by finding the maximum domain size that could fit in the computer's memory and downscaling to create 5 steps. Further work on very large domain sizes should handle temporary data storage in binary files on hard-disk, allowing a simulation to handle matrices of an optimal size in memory. The PSTD domain sizes require significantly fewer cells than the S/FDTD methods, due to the nature of stability in the PSTD method. As PSTD differentiation is undertaken using all linearly contiguous cells, the order of the differentiation is higher than that of FDTD and fewer points per wavelength are required for frequency domain accuracy [22].

When running each simulation, the supporting code around the simulation was kept to a similar format and only execution time was measured. No plotting was performed during the speed tests due to the single threaded nature of internal engine of Matlab. This would have significant performance implications on the overall speed of the simulation. The maximum frequency of interest for the simulations was 500 Hz, which was chosen due to the memory constraints described above.

3.1 Results

Figures 11 shows the mean time step execution speed for each model and domain size. These results suggest that using the PSTD method gives significantly faster average execution times than using the FDTD and SFDTD methods. This may be because of the utilization of optimised computation methods, and the relaxed domain attributes required for a simulation (i.e. smaller matrices than required in FDTD and SFDTD to simulate large domains, up to higher frequencies). However, implementing partially-absorbing boundary conditions, handling obstacles, and minimizing aliasing may be non-trivial work, where FDTD and SFDTD may provide easier-to-adjust simulation setups for different problems.

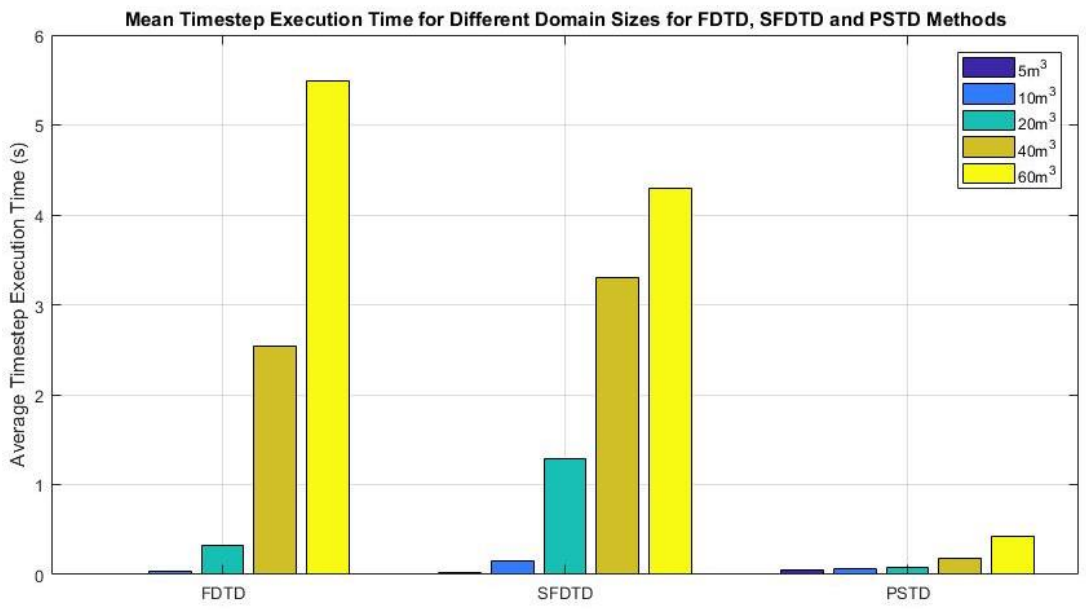


Figure 11 Mean execution time for each model for a range of domain sizes (linear time scale)

The results also show that SFDTD reduced average computation times only with the largest domain size and increased computation time for all other sizes. This is partially due to the non-optimised implementation of the method and the threshold of the window, as well as the extra number of computations required to create and then reference the window function (although, modelling larger domains is likely to give more significant improvements over standard FDTD). As shown in Figure 10, before the stimulus has much effect on the domain, the number of extra calculations being undertaken to create the window may be large enough to offset any benefits that such a window might give in terms of total domain used in computation, once wave fronts are propagating across the domain.

4 CONCLUSION & FURTHER WORK

This study gives an indication of potential improvements in execution speed of time domain methods, notably when using spectral methods. A window-based method of reducing domain computation area may improve the execution speed in very large simulations, but further work is required to confirm this. Differentiation is likely to be the process that is slowest. Addressing the speed of differentiation using different matrix sizes, indexing methods or strategies may improve execution speed of finite difference methods.

This research presented here has some key limitations. The implementations of each method presented are not mature and required fine-tuning to provide high quality results, both for measuring acoustic behaviour and achieving optimal performance. None of the methods presented acoustic behaviour that was easy to validate, and the number of time steps used in the final tests were small (due to memory limitations). A change in the initial conditions of the speed tests may have given different execution speed results, due to SFDTD's window function taking a long time to compute.

While this work shows good evidence that improvements can be made to the execution time, there is a further work required to solidify and validate these results. This work includes, but is not limited to:

- Experimenting with the process of SFDTD window calculation
- Determining the optimal window threshold for the SFDTD method
- Experimenting with methods to reduce the number of points required to represent a domain
- Examination and improvement of the output of PSTD, including performance of the PML and experimentation with Chebyshev PSTD
- Investigation into obstacles and partially absorbing boundary conditions in PSTD

Although more work needs to be done to improve the efficiency of time domain methods, this work shows potential for improvement. Further development of these methods could provide scalable and intuitive tools for simulating and analysing acoustic propagation within large domains without the need for specialist computing equipment.

5 REFERENCES

[1] S. Durbridge, “Efficient Acoustic Modelling of Large Spaces using Time Domain Methods Acoustic Modelling Applications,” pp. 1–8, 2017.

[2] J. De Poorter and D. Botteldooren, “Acoustical finite-difference time-domain simulations of subwavelength geometries,” *J. Acoust. Soc. Am.*, vol. 104, no. 3, pp. 1171–1177, 1998.

[3] D. T. Murphy, A. Southern, and L. Savioja, “Source excitation strategies for obtaining impulse responses in finite difference time domain room acoustics simulation,” *Appl. Acoust.*, vol. 82, pp. 6–14, 2014.

[4] S. Bilbao, “Optimized FDTD schemes for 3-D acoustic wave propagation,” *IEEE Trans. Audio, Speech Lang. Process.*, vol. 20, no. 5, pp. 1658–1663, 2012.

[5] L. Savioja, “Real-time 3D finite-difference time-domain simulation of low- and mid-frequency room acoustics,” *Proc Int Conf Digit. Audio Eff.*, pp. 1–8, 2010.

[6] B. Hamilton and S. Bilbao, “Fourth-Order and Optimised Finite Difference Schemes for the 2-D Wave Equation,” *Research.Ed.Ac.Uk*, vol. 2, no. 2, pp. 1–8, 2013.

[7] J. Van Mourik and D. T. Murphy, “Hybrid Acoustic Modelling of Historic Spaces Using Blender,” no. c, 2014.

[8] A. J. Hill, “Analysis , Modeling and Wide-Area Spatiotemporal Control of Low-Frequency Sound Reproduction,” University of Essex, 2012.

[9] L. N. Trefethen, “Spectral Methods in Matlab,” *Lloydia Cincinnati*, vol. 10, p. 184, 2000.

[10] M. Hornikx, T. Krijnen, and L. Van Harten, “OpenPSTD: The open source pseudospectral time-domain method for acoustic propagation,” *Comput. Phys. Commun.*, vol. 203, pp. 298–308, 2016.

[11] J. A. S. Angus and A. Caunce, “A GPGPU Approach to Improved Acoustic Finite Difference Time Domain Calculations,” in *128th Audio Engineering Society Convention*, 2010.

[12] C. Doerr, “SPARSE FINITE-DIFFERENCE TIME DOMAIN SIMULATION,” US 2014/0365188 A1, 2014.

[13] C. Doerr, “Sparse Finite Difference Time Domain Method,” *IEEE Photonics Technol. Lett.*, vol. 25, no. 23, p. 1, 2013.

[14] C. Doerr, “3D Sparse Finite-Difference Time-Domain Simulation of Silicon Photonic Integrated Circuits,” vol. i, pp. 4–6, 2015.

[15] S. Bilbao, *Wave and Scattering Methods for Numerical Simulation*, 1st ed. London: John Wiley & Sons, 2004.

[16] K. Kunz and R. Luebbers, “The Finite Difference Time Domain Method for Electromagnetics.” p. 464, 1993.

[17] C. Spa, A. Garriga, and J. Escolano, “Impedance boundary conditions for pseudo-spectral time-domain methods in room acoustics,” *Appl. Acoust.*, vol. 71, no. 5, pp. 402–410, 2010.

[18] J. P. Berenger, “A perfectly matched layer for the absorption of electromagnetic waves,” *J. Comput. Phys.*, vol. 114, pp. 185–200, 1994.

[19] J. Van Mourik and D. Murphy, “Explicit higher-order FDTD schemes for 3D room acoustic simulation,” *IEEE/ACM Trans. Speech Lang. Process.*, vol. 22, no. 12, pp. 2003–2011, 2014.

[20] B. Hamilton and S. Bilbao, “Fourth-Order and Optimised Finite Difference Schemes for the 2-D Wave Equation,” *Research.Ed.Ac.Uk*, vol. 2, no. 2, pp. 1–8, 2013.

[21] C. Spa, J. Escolano, and A. Garriga, “Semi-empirical boundary conditions for the linearized acoustic Euler equations using Pseudo-Spectral Time-Domain methods,” *Appl. Acoust.*, vol. 72, no. 4, pp. 226–230, 2011.

[22] M. Hornikx, R. Waxler, and J. Forssén, “The extended Fourier pseudospectral time-domain method for atmospheric sound propagation.,” *J. Acoust. Soc. Am.*, vol. 128, no. 4, pp. 1632–46, 2010.