Efficient acoustic modelling of large acoustic spaces using finite difference methods

S Durbridge Bowers & Wilkins, West Sussex, UK

A Hill Department of Electronics, Computing & Mathematics, University of Derby, Derby, UK

# INTRODUCTION

* Introduce reason for doing study
  + Improvements in the flexibility, accuracy and performance of simulation tools could help towards making predictions, performing low-cost exploration (rapid prototyping) and system design workflows easier, faster and more intuitive. – WHAT
  + Time domain numerical methods used for performing acoustic simulation, could provide useful visual information, as well as reasonably accurate measurement data. – WHAT
* Lit Review – A bit more about time domain numerical methods for acoustic simulations
  + One of the early proponents of work on time domain numerical methods for acoustics was Bootledooren[1]; whose work involved porting the finite difference time domain (FDTD) and finite volume time domain (FVTD) methods from electromagnetic simulation, for use in low frequency acoustic simulation. – WHAT
  + This work has been followed on by many researchers such as Murphy[2], Bilbao[3] and Hamilton[4], to expand and improve potential use of these methods. – WHAT
  + In his thesis doctoral thesis[5], Hill presented a simple and effective implementation of the FDTD for low frequency modelling, that was the basis for the work presented below. – WHAT
  + Following key work such as that by Trefethen[6], a number of project have begun to explore the Fourier pseudo-spectral time domain method (PSTD), most notably the OpenPSTD project from Eindhoven University of Technology[7]. – WHAT
  + Caunce and Angus[8] 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 the potential improvements in execution speed by performing spectral differentiation on a GPGPU using the Fourier PSTD method. This study was the basis for the PSTD work in this study. – WHAT
  + Meanwhile in the field of microcontroller development, Doerr[9] produced work on the spare 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 and a lot of time to simulate. The sparse finite difference time domain method presented by Doerr essentially presents a moving window method of reducing the size of the portion of the domain being solved for at any one time. - WHAT
  + Methods such as FDTD present benefits for low frequency simulation over other simulation methods such as ray based and direct calculation, as features such as the modal behaviour of the acoustic system being modelled is accounted for within the solving method. 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 Eigen-modes of general acoustic systems. – WHAT/WHY
* Defined marker –
  + Regardless of the method used for solving in the simulation, it should be possible to setup and solve simulation in reasonable time. – PERTURBATION
* Introduce the problem
  + However, large scale and high frequency simulations are difficult to perform using these methods, not only because of the large computational resources required to perform the simulations; often the time required to perform these simulations is also severely limiting. – WHAT
  + This is due to the various conditions for accuracy and stability that must be met when solving partial differential equations numerically. – WHY
  + These conditions are difficult to overcome, but if it is possible to reduce the execution time (and ideally the memory requirements) of a simulation, using time domain numerical methods could become more accessible for less specialist users such as the slightly more academic sound engineer, loudspeaker designers and undergraduate students. - WHAT
* A slightly deeper look at the problem
  + The FDTD method in a more basic implementation involves representing an acoustic system such as a room, as a set of matrices that represent points of pressure and points of velocity within the system. – WHAT
  + The conceptual distance between the points in the system is defined by the stability of the equations being solved, and the highest frequency of interest. The number of points is also dictated by the size of the system. – WHAT
  + A wave equation is split up into two reciprocating parts, using velocity points to calculate surrounding pressure points, and pressure points to calculate surrounding velocity points. This is performed in a leapfrog fashion in steps over time, the conceptual length of the step is also determined by the highest frequency of interest and the stability or the solving method. – WHAT
  + More specifically, the Courant-Freidrichs-Lewey (CFL) condition is a condition that dictates the minimum length of time step and spatial step required for a convergent solution. –WHAT The equation for Courant number in the one dimensional case is given below:
  + The maximum value of is determined by the stability of the method being used to solve the PDE, and for a simple explicit FDTD simulation is typically 1. - WHY
  + As the amount of memory required for a simulation scales with frequency, it also scales with domain size as the constraints are points per distance. It is difficult to perform large simulations up to high frequency, as the amount of memory required to perform a simulation can quickly become greater than that available in non-specialist computer systems. – WHAT
  + Another fundamental problem with the FDTD method is the requirement to constantly perform non-contiguous memory accesses to perform calculations. –WHAT
  + Computer memory access (particularly in the CPU cache) is optimised for contiguous accesses in one particular direction. The FDTD method can require the system to access memory in an orthogonal direction to the optimum around 50% of the time, and also requires the system to index into two large blocks of memory simultaneously. – WHY
* Introduce PSTD and SFDTD and explain why this may speed things up
  + Two similar methods to FDTD that may execute faster are the PSTD and SFDTD methods. – WHAT
  + The PSTD method follows a similar form to the FDTD methods in most respects. The differentiation in the method is performed in the frequency domain; each domain matrix is multiplied by the impulse response of an ideal differentiator in the frequency domain, before being used to calculate the new values of the reciprocating field. - WHAT
  + While this method has the potential to be much faster than FDTD by leveraging the speed of optimised memory access and discrete Fourier transforms, this method requires a PML to overcome Gibbs phenomenon and can suffer from aliasing due to the non-periodic nature of the system being simulated. – WHAT/WHY
  + The SFDTD method involves windowing around the portions of the domain that have above a threshold of energy. This window is then used as a guide, and only the necessary portions of the domain are computed. - WHAT
  + This method is still very much in early development and there is little literature in acoustics that have explored this method. – WHAT
  + As such, a robust and well validated implementation of SFDTD in acoustics has yet to be reported. – WHAT
  + Further, the method may only be useful for speed improvements before the level of the diffuse field is relatively high i.e. when the early strong reflections are propagating across the domain. This method would also ideally use a high order FDTD stencil that doesn’t suffer from numeric dispersion. - WHAT
  + Figures backing up the problems
* Explain what this paper is all about – This paper is a cursory glance into a deep problem, that might be helped with some cunning future work - Split out the content of the paper – the tests, results and such
  + The aim of this paper is to explore the improvements of execution speed of the PSTD and SFDTD methods, over the FDTD method. – WHAT
  + In the following section of this paper, a series of simulation test cases are described. – WHAT
  + Following this, the results of the acoustic output and the execution speed of the simulation methods are compared. – WHAT
  + Finally, the execution speed profile of each method is reviewed, highlighting where the speed bottlenecks occur in each method. – WHAT
  + The work described in this paper was undertaken using the Matlab language and IDE, as part of an MSc project at the University of Derby.

# Experiment

* Introduce the experiment conditions: - WHERE (and why where)
  + In order to test the execution speed of the FDTD, PSTD and SFDTD methods, all three 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
  + This computer system uses standard, easily available consumer grade parts and was configured using inbuilt automatic tools, thus requiring little specialist configuration knowledge.
* Defining the baseline – FDTD – HOW
  + Initially the FDTD solving method was implemented as a function, based on the work by Hill[5]. First a 2D version, and then a 3D version was implemented. – WHAT
  + The differentiation in the FDTD method is performed by indexing into discrete points of pressure and velocity potential matrices, and calculating new local values of each based on the old and surrounding values of the related variable at each point in the doman. – WHAT
  + Following this, a test was executed to check that a stimulus is propagated across the domain without great distortion, spectral shifting or unstable behaviour. – WHAT
  + The domain setup was a 5m wide by 4m deep by 3m tall rectangle, and had partially absorbing boundaries with an absorption coefficient of 0.45. The maximum analysis frequency was 5kHz. – WHAT
  + The stimulus used was three sets of 10 cycles of 1kHz windowed tone burst, with a rest period of 3 times the length of the tone. The tone burst stimulus lasted for 0.1s, following which there was 0.1s of silence to allow for decay of the reverberation. – WHAT
  + The signal source was position as close to 1m away from a corner of the domain as possible. 5 points of the domain (near the corners and the centre) were ‘recorded’ for the length of the simulation. – WHAT
* Reviewing the results – WHAT & WHY
  + The FIGURE below shows the normalise source and receiver signals in the time domain, and in the frequency domain using Welch’s power spectral density estimation method built into matlab. – WHAT
  + The output shows that the frequency of the propagations across the domain is the same as the stimulus, and no high level oscillatory components are present. – WHAT
  + The time domain behaviour of the simulation appears to show sensible propagation delay between measurement points, with decay that would suggest the simulation is convergent. – WHAT
  + Using the inbuilt code profiler tools in Matlab, was possible to analyse the performance of the FDTD method and determine where the speed was restricted. The figure below shows the speed of the lines of code in the solving function.
  + The slowest parts of the solving method are the parts where the differentiation is occurring, where the system is having to perform multiple memory accesses to separate large matrices. Managing or reducing these accesses may help speed up solving performance. – WHAT
  + NOTE: NOISEFLOOR OF THE MEASUREMENT IS HIDDEN
  + NOTE: PLOT SCALE
* PSTD – HOW
  + The PSTD method was implemented as a set of Matlab functions in a similar way to the FDTD method, and was based on the work by Caunce & Angus[8]. – WHAT
  + To implement partially absorbing boundary conditions, work by Spa *et al*[10].
  + The differentiation in the PSTD method is performed by performing a discrete Fourier transform on 1 dimension of the domain, and multiplying the frequency domain spatial data with the impulse response of an ideal differentiator. The inverse discrete Fourier transform of the differentiated spatial domain data is then used to calculate the new values of the reciprocating field. The differentiation is performed singularly in all spatial dimensions of interest. – WHAT
  + This method of differentiation may be preferential to the FDTD method, because the differentiation for calculating any one point includes differentiation of all points in the domain that are linearly coupled. This not only increases the order of accuracy of the method, but use of optimised libraries for the Fourier transform and SIMD can be leveraged by the compiler, to increase the speed of computations for the differentiation. – WHY
  + The same simulation test as that described above for the FDTD method, was used with the PSTD implementation. The figure below shows the output of the method in the same format.
  + The frequency domain response of the system gives a centre frequency of wave propagation at 1kHz, the same as the stimulus tone. The width and shape of the window is however not an ideal hump, and some aliasing appears in the response. Further, the rate and level of signal decay would indicate that for the same desired absorption coefficients, the absorption is greater than on the FDTD method. Although the quality of the output of the system is questionable, it would appear that the overall performance is acceptable enough to use this algorithm for speed testing. - WHAT
* SFDTD – HOW
  + The SFDTD method was implemented as a set of Matlab functions, based on the same work by Hill mentioned above, and with inspiration from the work of Doerr[9].
  + Though Doerr’s work in computation electromagnetics is interested, its list based method for accounting for window shape and position is perhaps not appropriate for an elastic wave system where a diffusely fluctuating field is desirable for calculation. – WHAT
  + The approach taken for implementing SFDTD in this study was to create a normalising indexing window, based on the absolute pressure of the domain. The normalised shape of the domain (around a threshold) was then smoothed using a Gaussian image filtering technique, to ensure that the window surrounding points of high enough pressure within the domain are also used for differentiation. This allows wave fronts to propagate naturally across the domain unimpeded by the window itself. Due to time constraints with this study, little work was done to optimise the threshold value and 40dB was used throughout the study. Further work should be undertaken to determine ideal smoothing window shapes and threshold values. – WHY
  + The next step was to try the SFDTD method to see if we can get speed improvements earlier in the simulation – WHY- FIGURE 7 EXAMPLE OF THE CONFIGURATION
  + Threshold value set to X – WHAT
  + Because it’s a minimum noise floor, and we spend loads of time below that not doing too much- but really its from dead reckoning – WHY
  + Same set of domains and fmax used for reasonable comparison – WHAT and WHY
  + The method would potentially be faster for the early reflections, because we chop down the size of the computation – WHAT & WHY
  + To give an idea of the behaviour of the strong wavefronts in a space and how these behave – FURTHER WHY
  + Because of the time limitations, the method wasn’t really optimised – WHAT
  + Explain that although the domain sizes and frequency used weren’t that big or high, it’s a starting point to go further– WHAT
* A little bit more about the method – HOW of the thing measuring
  + To examine the execution speed performance of each time domain method, each method was used to solve a series of increasingly large rectangular 3D domains. – AIM
  + The execution speed for each of the 2000 time-step iterations of each method was measured using the inbuilt Tic/Toc functionality of Matlab. - WHAT
  + A set of 5 domain sizes were used, the size and number of cells for each time domain method are given in table 1.

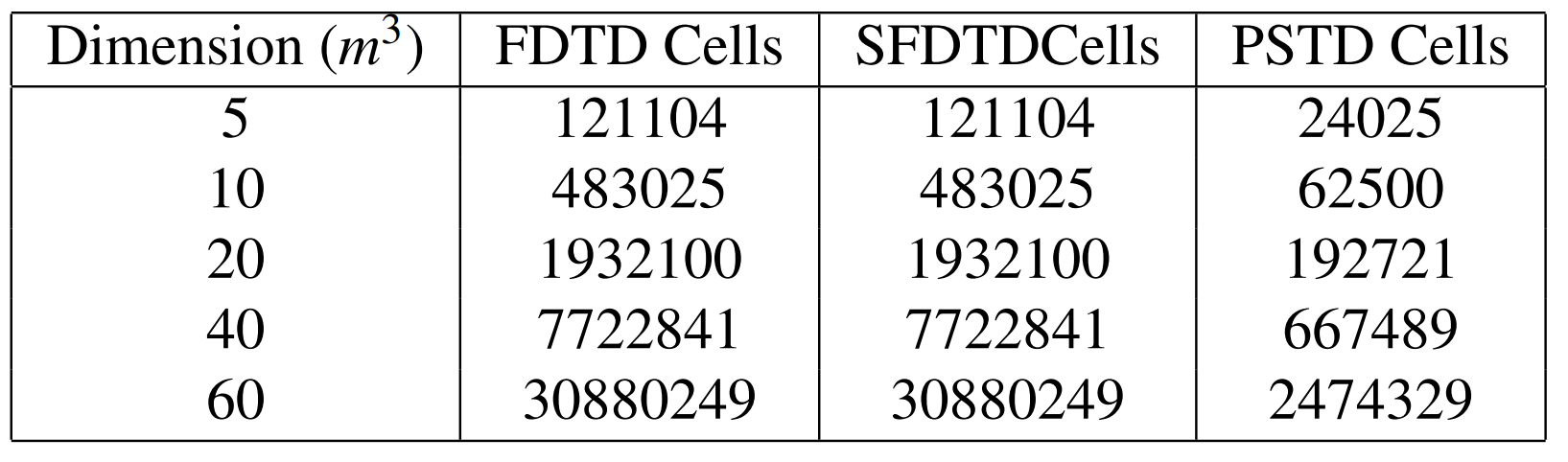


Table 1Set of domain sizes and domain cells for each time domain method

* + These domain sizes were chosen by choosing a scaling factor up to the maximum domain size and variables that could fit in the computer’s memory. Ideally, some future work on very large domain sizes would handle temporary data storage in binary files on hard-disk, allowing a simulation system to only have necessary large files in memory at any one time. – WHY/WHAT
* Setup of the methods – HOW of the surrounding code
  + When running each simulation with the different time domain methods, the supporting code around the simulation kept a similar format and only the execution of the time step solving was measured. No plotting was performed during the speed tests; due to the single threaded nature of Matlabs internal engine, this would have significant performance implications on the overall speed of the simulation execution. – WHY
  + The maximum frequency of interest for the simulations was 500Hz, which was chosen due to the size constraints of arrays in memory as per the domain sizes described above. – WHY
  + 1 – FDTD setup – dx dt etc
  + 2 – PSTD setup – dx dt etc
  + 3 – SFDTD setup – dx dt etc
  + WHY the setups? We want a reasonable behaviour, but it will never be true to real life – CAVEATS AND LIMITATIONS ARE IMPORTANT HERE
* Setup of the source and the stimulus – WHY?
  + Single omnidirectional source with tone-bursts & MLS
  + WHY that source? To find out if there is smear or other nasty behaviour, and stability too – FIGURES OF THIS DATA
  + Specifically, we would like to know the models are outputting similar things
  + Omnidirectional soft-source i.e. transparent, but not aligned with the impedance of the grid

# Results

* PSTD Results
* Reiterate the conditions of the experiment
  + X domain sizes and Y maximum frequency – WHAT
  + The figures below show the spectral output of the simulation at the measurement positions– WHAT
* Explanation of what the results show
  + It can be seen what the frequency responses in-front and behind with one piece of deck make little to no change In the audience area – WHAT
  + On and under the stage subwoofer positions there is a clear difference in frequency response with the deck in place, with a 15dB increase at 55Hz, and a collapse in the cardioid polar response when measured both on and under the stage – WHAT
  + With the subwoofer just in front of the stage, you get better on axis performance than the other two subwoofer positions
* Focus on the execution time profile
  + Its necessary to compare the frequency responses in more than on axis and off axis – WHAT
  + The 6 +/- 20degree measurement points were ignored because people aren’t normally there – WHY
  + The groups (onstage and audience) were averaged to give an idea of system behaviour in the audience and on the stage – WHAT & WHY
  + The averaged frequency responses with the single deck stage were taken from the no stage ones to give a deviation – WHAT & WHY
  + These responses are given in FIGURE 10 – WHAT FIGURES SHOWING DEVIATION
* Rounding off this part of the analysis of the speed of the PSTD simulations
  + This analysis provides ‘conclusive’ evidence that the best place to have a subwoofer around one piece of stage deck is in front of the deck – WHAT
  + This place exceeds or matches the front-to-back rejection ratio of the subwoofer with no deck – WHY
  + The under and on stage placements show reduction in the stage rejection in the subwoofers passband – WHAT
* SFDTD Results
* Introduce the SFDTD results analysis
  + An identical analysis to the small stage was done with PSTD – WHAT reiteration
  + The frequency responses are given in the below FIGURES
  + Threshold level of window was set to X – WHAT reiteration
* Focus on the execution time profile – what was slow
  + There are some similar and dissimilar trends – WHAT
  + The under stage location appears to be the worst choice in both cases, even though the large stage data was modelled – WHY/WHAT
  + So more investigation is needed – WHAT
  + Between 60 and 90Hz, there is less front-back rejection for the in-front of stage position than with no stage – WHAT
* Explanation of why SFDTD was faster at the beginning than the middle – BIG EXPLANATION OF WHY SOME UNEXPECTED VARIANCE
  + A possible explanation is the big stage is acoustically larger in the subwoofer passband, so most frequencies interact with the stage - WHAT
  + A wall was just behind the stage, so a strong reflection may have interacted with the measurements – WHY
  + The propagation distance of the reflection was 9.7 meters for the first driver of the subwoofer, a wavelength relative to 17.68Hz - WHAT
  + Odd integer multiples of this arrive at the sub 180degrees out of phase with the direct output of the sub, causing cancellation in front of the drive unit – WHY
  + Key frequencies of this odd order multiples are 53Hz and 89Hz – WHAT
  + The propagation distance of the reflection for the second driver is 8.7 meters, equating to integer multiples of 59 and 99Hz – WHAT
  + The significance of this loss of stage rejection requires further work – WHAT
* The execution speed comparison
  + Finally
  + Mean front-back SPL rejection over two frequency ranges (38 – 110Hz) and (20-300Hz) – WHAT
  + Wider range to account for stage and room resonances –WHY
  + Is given in table 1 & 2 – WHAT
  + The data in those tables give a clear and concise summary – WHAT
  + Placing a cardioid subwoofer on top or under a stage, regardless of stage size, will reduce the front-back rejection ratio a lot! – WHAT
  + Placing the subwoofer in front of the stage will allow the rejection to be maintained – THE BIG OUTCOME OF THE STUDY – THE WHAT

# CONCLUSION & Further Work

* Review of the results and final outcome
  + The results in the paper aid further understanding of the problem – the effect of the stage on the polar response – WHAT
  + If a sub is under or on top, the speaker won’t have the same great directivity – WHAT
  + The result is high SPLS on stage which is no good – WHAT
  + Placing the sub in front of the stage is the best for maintaining the polar response – WHAT
  + The results here and the results from earlier work show that its best to place the sub in front of stage – WHAT BIG WHAT
* Caveats
  + This analysis is meaningless if the subwoofers are flown above the stage – Counter point
  + If you do this, do beam steering so that there is less noise on the stage
* Further work
  + While this work shows some good evidence, more work needs to be done to fully know what is going on – WHAT
  + These recommendations for further work – WHAT
    - Repeat experiment in anechoic space with full stage
    - Repeat experiment in a large scale live event
    - Examine how multi-unit cardioid subwoofer arrays interact with each-other in different shape arrays with different stage positions
    - Investigate effects the stage has on transient response
* Fourth paragraph
  + Although more work needed to understand, it is clear that you should think about where to place ground based subs at events, especially when directivity is important. – WHAT WIDER
  + Most commercially available software omits any stage effects, so it’s essential to know what the stage does to sub cardioid performance, so not to mess up and to get best response shape – WHAT

# References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] 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.

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

[10] 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.