International Journal of HIGH PERFORMANCE COMPUTING APPLICATIONS

*The International Journal of High Performance Computing Applications*
1–19
© The Author(s) 2015
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1094342015581024
hpc.sagepub.com
SAGE

# Full-wave nonlinear ultrasound simulation on distributed clusters with applications in high-intensity focused ultrasound

**Jiri Jaros[1], Alistair P Rendell[2] and Bradley E Treeby[3]**

**Abstract**
Model-based treatment planning and exposimetry for high-intensity focused ultrasound requires the numerical simulation of nonlinear ultrasound propagation through heterogeneous and absorbing media. This is a computationally demanding problem due to the large distances travelled by the ultrasound waves relative to the wavelength of the highest frequency harmonic. Here, the *k*-space pseudospectral method is used to solve a set of coupled partial differential equations equivalent to a generalised Westervelt equation. The model is implemented in C++ and parallelised using the message passing interface (MPI) for solving large-scale problems on distributed clusters. The domain is partitioned using a 1D slab decomposition, and global communication is performed using a sparse communication pattern. Operations in the spatial frequency domain are performed in transposed space to reduce the communication burden imposed by the 3D fast Fourier transform. The performance of the model is evaluated using grid sizes up to $4096 \times 2048 \times 2048$ grid points, distributed over a cluster using up to 1024 compute cores. Given the global nature of the gradient calculation, the model shows good strong scaling behaviour, with a speed-up of 1.7x whenever the number of cores is doubled. This means large-scale simulations can be distributed across high numbers of cores on a cluster to minimise execution times with a relatively small overhead. The efficacy of the model is demonstrated by simulating the ultrasound beam pattern for a high-intensity focused ultrasound sonication of the kidney.

**Keywords**
High-intensity focused ultrasound, Fourier pseudospectral methods, Westervelt equation, FFTW, large-scale problems, distributed computing

## 1 Introduction

High-intensity focused ultrasound (HIFU) is a non-invasive therapy in which a tightly focused beam of ultrasound is used to rapidly heat tissue in a localised region until the cells are destroyed (Kennedy et al., 2003; ter Haar, 2007). In recent years, HIFU has been used in clinical trials for the treatment of tumours in many organs, including the prostate, kidney, liver, breast and brain (Kennedy et al., 2003; Clement, 2004; Jolesz and Hynynen, 2008; Zhang and Wang, 2010). While the number of HIFU devices on the market continues to grow, one hurdle that currently prevents wider clinical use is the difficulty in accurately predicting the region of damaged tissue given a particular patient and set of treatment conditions. In principle, this could be calculated using appropriate acoustic and thermal models (Paulides et al., 2013). However, the modelling problem is both physically complex and computationally challenging. For example, the heterogeneous material properties of human tissue can cause the ultrasound beam to become strongly distorted (Liu et al., 2005), the exact values for the material properties and their temperature dependence are normally unknown (Connor and Hynynen, 2002), and the rate

[1]Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic
[2]Research School of Computer Science, Australian National University, Canberra, Australia
[3]Department of Medical Physics and Biomedical Engineering, University College London, London, United Kingdom

**Corresponding author:**
Jiri Jaros, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic.
Email: jarosjir@fit.vutbr.cz

**Table 1.** Examples of possible domain sizes and frequency ranges encountered in high-intensity focused ultrasound (HIFU). The grid sizes are based on using a uniform Cartesian grid at the Nyquist limit of two points per minimum wavelength (PPMW) assuming a sound speed of 1500 m/s. The memory usage is based on storing a single matrix at the specified grid size in single-precision (4 bytes per grid element).

| Domain Size ($cm^3$) | Maximum Freq (MHz) | Domain Size (wavelengths) | Grid Size (at 2 PPMW) | Memory Per Matrix (GB) |
|---|---|---|---|---|
| $5 \times 5 \times 5$ | 5 | $333^3$ | $667^3$ | 1.1 |
| | 10 | $667^3$ | $1333^3$ | 8.8 |
| | 20 | $1333^3$ | $2667^3$ | 71 |
| | 50 | $3333^3$ | $6667^3$ | 1100 |
| $10 \times 10 \times 10$ | 5 | $667^3$ | $1333^3$ | 8.8 |
| | 10 | $1333^3$ | $2667^3$ | 71 |
| | 20 | $2667^3$ | $5333^3$ | 570 |
| | 50 | $6667^3$ | $13333^3$ | 8800 |
| $20 \times 20 \times 20$ | 5 | $1333^3$ | $2667^3$ | 71 |
| | 10 | $2667^3$ | $5333^3$ | 570 |
| | 20 | $5333^3$ | $10667^3$ | 4500 |
| | 50 | $13333^3$ | $26667^3$ | 71000 |

and mechanism for tissue damage are both temperature and cell specific (Lepock, 2003).

The question of how best to model the physical interactions between ultrasound waves and biological tissue has been widely studied, and work in this area is ongoing. However, the problem that has attracted much less attention, but which is equally important, is the issue of computational scale. This arises because of two related factors. The first is that the generated acoustic pressures are of sufficient magnitude that the wave propagation is nonlinear. This causes the ultrasound waves to steepen as they propagate, which generates high-frequency harmonics of the source frequency (this is usually between 0.5 and 2 MHz for HIFU treatments where the transducer is positioned outside the body). At low focal intensities, nonlinear effects cause energy to be generated up to at least the 10th harmonic (Wojcik et al., 1995). At very high focal intensities where strongly shocked waves are produced, as many as 600 harmonics might be required to model the focal heating accurately (Khokhlova et al., 2010). Thus, the frequency content of the propagating ultrasound waves can be very broadband.

The second factor is that the domain of interest encompassing the HIFU transducer and the treatment area is normally on the order of centimetres to tens of centimetres in each Cartesian direction. Compared to the size of the acoustic wavelength at the maximum frequency of interest, this equates to wave propagation over hundreds or thousands of wavelengths. To illustrate the scale of the problem, a list of representative domain sizes is given in Table 1. If the governing equations describing the HIFU field are solved on a uniform Cartesian grid where the grid spacing is defined to meet the Nyquist limit of two points per minimum wavelength, the resulting grid sizes can exceed $10^{12}$ grid points. If conventional finite difference schemes are used (which arguably is still the most common approach for time-domain modelling of broadband acoustic waves), the required grid sizes can be much greater. This is due to the large number of grid points per wavelength needed to avoid numerical dispersion over these length-scales. In many cases of practical interest, the grid sizes needed simply makes the simulations intractable.

To avoid the computational complexity of directly solving nonlinear acoustic equations in 3D, simplifying assumptions are normally made. In particular, one-way or evolution-type models have been very successful in simulating HIFU fields in homogeneous media (Averkiou and Cleveland, 1999; Curra et al., 2000; Khokhlova et al., 2001; Yuldashev and Khokhlova, 2011). In one-way models, the governing equations are formulated in retarded time, and the domain is discretised in $x$, $y$ and $\tau$ (where $\tau$ is the retarded time variable) instead of $x$, $y$ and $z$. The simulation then progresses by propagating the time history of the wave-field from plane to plane in the $z$-dimension (this is illustrated in Figure 1). If a small time window is used, this approach can significantly reduce the amount of memory required for broadband simulations. For example, Yuldashev and Khokhlova (2011) used grid sizes in the $x$-$y$ plane with $10{,}000 \times 10{,}000$ grid points modelling up to 500 harmonics using a shared-memory computer with 32 GB of RAM. The main restriction of one-way models is that a heterogeneous distribution of tissue properties cannot be included (except via the use of phase layers (Yuldashev et al., 2010)), and scattered or reflected waves are not modelled. This means the significant distortion of HIFU beams that can occur in a clinical setting cannot be accounted for (Liu et al., 2005).

In addition to one-way models, several full-wave nonlinear ultrasound models in 3D have been reported

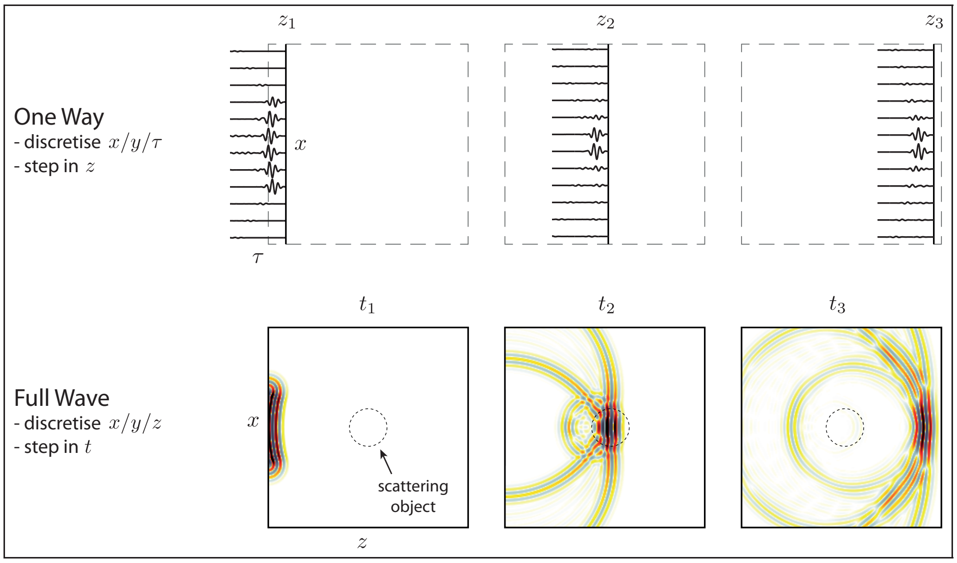

**Figure 1.** Schematic illustrating the difference between one-way and full-wave ultrasound models. In one-way models, the governing equations are formulated in retarded time, and the domain is discretised in *x*, *y* and $\tau$ (where $\tau$ is the retarded time variable). The simulation then progresses by propagating the time history of the wave-field from plane to plane in the z-dimension. In full-wave models, the domain is discretised in *x*, *y* and z and the simulation progresses by stepping through time. One-way models are typically more memory efficient, while full-wave models can account for heterogeneous material properties, reflected and scattered waves.

based on the finite difference time domain (FDTD) method (Connor and Hynynen, 2002; Pinton et al., 2009; Okita et al., 2011). For example, Pinton et al. (2009) presented a solution to the Westervelt equation (a particular nonlinear wave equation) using a second-order in time, fourth-order in space finite difference scheme. This was used to investigate the effects of non-linear wave propagation on HIFU therapy in the brain (Pinton et al., 2011). Simulations were run on 112 cores of a distributed cluster with grid sizes and run times on the order of $800 \times 800 \times 800$ and 32 hours. Okita et al. (2011) used a similar model to study HIFU focusing through the skull, using 128 cores of a distributed cluster with grid sizes and run times on the order of $800 \times 600 \times 600$ and 2 hours respectively. They also demonstrated excellent weak-scaling results for a benchmark using up to $3.45 \times 10^{11}$ grid points distributed across 98,304 cores on the K computer (a super-computer installed at the RIKEN Advanced Institute for Computational Science in Japan) (Okita et al., 2014). In another study, Pulkkinen et al. (2014) used a hybrid FDTD model to study transcranial focused ultrasound using grid sizes up to $1338 \times 1363 \times 1120$ running on 96 cores of a distributed cluster with compute times on the order of 40 hours.

Computationally, FDTD schemes have excellent weak-scaling properties and good computational performance for a given grid size. However, as mentioned above, the significant drawback is that for large domain sizes, very dense grids are needed to counteract the accumulation of numerical dispersion. One way of reducing this computational burden is to compute spatial gradients using the Fourier pseudospectral method (Hesthaven et al., 2007). This eliminates the numerical dispersion that arises from the discretisation of spatial derivatives, and significantly reduces the number of grid points needed per acoustic wavelength for accurate simulations. Computational efficiency can be further enhanced by using the *k*-space pseudospectral method. This extends the pseudospectral method by exploiting an exact solution to the linearised wave equation to improve the accuracy of computing temporal gradients via a finite-difference scheme (Mast et al., 2001; Tabei et al., 2002). This approach was first introduced in electromagnetics by Haber et al. (1973), and in acoustics by Bojarski (1982, 1985). Since then, both pseudospectral and *k*-space schemes have been applied to linear (Fornberg, 1987; Liu, 1999; Mast et al., 2001; Tabei et al., 2002; Cox et al., 2007; Daoud and Lacefield, 2009; Tillett et al., 2009) and nonlinear (Wojcik et al., 1998; Treeby et al., 2011; Albin et al., 2012; Jing et al., 2012; Treeby et al., 2012) ultrasound simulations in heterogeneous media for relatively modest grid sizes. Compared to FDTD schemes, the increase in accuracy arises due to the global nature of the basis functions (in this case complex exponentials) used to interpolate between the grid points. However, for large-scale problems where the numerical model must be implemented using a parallel computer with distributed resources, the global nature of the gradient calculation also introduces new challenges. This is due to the significant amount of global communication required between the compute cores (Daoud and Lacefield, 2009).

Another approach for minimising the grid size and number of time steps needed for accurate ultrasound simulations, is the iterative nonlinear contrast source (INCS) method (Verweij and Huijssen, 2009; Huijssen and Verweij, 2010; Demi et al., 2011). In this approach, terms describing the contributions of nonlinear effects and heterogeneous material parameters are reformulated as contrast source terms. The resulting wave equation is then solved iteratively using Green's function methods (Verweij and Huijssen, 2009). While this works well for weakly nonlinear fields, the requirement for storing the complete time history of the wavefield, and the evaluation of a 4D convolution at every time step makes it difficult to extend this approach to the large-scale problems encountered in HIFU.

Building on work by Tabei et al. (2002) and others, we recently proposed an efficient full-wave nonlinear ultrasound model based on the *k*-space pseudospectral method (Treeby et al., 2012). Here, we present an extension of this model for performing large-scale HIFU simulations on a distributed computer cluster with grid sizes up to $4096 \times 2048 \times 2048$. A brief overview of the governing equations and the *k*-space pseudospectral model is given in Section 2. The software implementation and parallelisation strategies chosen for mapping the spectral gradient calculations onto a distributed computer cluster are then discussed in Section 3. In Section 4, performance and scaling results are presented for running simulations on up to 1024 cores. Results from a representative large-scale simulation of a HIFU treatment of the kidney are then presented in Section 5. Finally, summary and discussion are presented in Section 6.

## 2 Nonlinear ultrasound model

### 2.1 Nonlinear governing equations

For modelling the propagation of intense ultrasound waves in the human body, the governing equations must account for the combined effects of nonlinearity, acoustic absorption and heterogeneities in the material properties (sound speed, density, acoustic absorption and nonlinearity parameter). Following Treeby et al. (2012), the required governing equations can be written as three coupled first-order partial differential equations derived from the conservation laws and a Taylor series expansion for the pressure about the density and entropy

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= -\frac{1}{\rho_0}\nabla p + \mathbf{F} && \text{(momentum conservation)} \\ \frac{\partial \rho}{\partial t} &= -\rho_0 \nabla\cdot\mathbf{u} - \mathbf{u}\cdot\nabla\rho_0 - 2\rho\nabla\cdot\mathbf{u} + \mathrm{M} && \text{(mass conservation)} \qquad (1) \\ p &= c_0^2\left(\rho + \mathbf{d}\cdot\nabla\rho_0 + \frac{B}{2A}\frac{\rho^2}{\rho_0} - \mathrm{L}\rho\right) && \text{(pressure} - \text{density relation)} \end{aligned}$$

Here $\mathbf{u}$ is the acoustic particle velocity, $\mathbf{d}$ is the acoustic particle displacement, $p$ is the acoustic pressure, $\rho$ is the acoustic density, $\rho_0$ is the ambient (or equilibrium) density, $c_0$ is the isentropic sound speed and $B/A$ is the nonlinearity parameter which characterises the relative contribution of finite-amplitude effects to the sound speed. These equations account for cumulative nonlinear effects (nonlinear effects that build up over space and time) up to second-order in the acoustic variables, equivalent to the Westervelt equation (Westervelt, 1963; Hamilton and Morfey, 2008). All the material parameters are allowed to be heterogeneous. Two linear source terms are also included, where $\mathbf{F}$ is a force source term which represents the input of body forces per unit mass in units of N kg$^{-1}$ and M is a mass source term which represents the time rate of the input of mass per unit volume in units of kg m$^{-3}$ s$^{-1}$.

The nonlinear term in the mass conservation equation accounts for a convective nonlinearity in which the particle velocity affects the wave velocity. Using the linearised form of the equations given in equation (1), this term can be written in a number of different ways. Following Aanonsen et al. (1984), the substitution of equations valid to first-order in the acoustic variables into terms that are second-order in the acoustic variables leads to third-order errors, which can be neglected. This leads to

$$-2\rho\nabla\cdot\mathbf{u} \approx \frac{2}{\rho_0}\rho\frac{\partial \rho}{\partial t} = \frac{1}{\rho_0}\frac{\partial \rho^2}{\partial t} \approx \frac{1}{\rho_0 c_0^4}\frac{\partial p^2}{\partial t} \qquad (2)$$

In equation (1), the nonlinear term is written in the first form shown in equation (2) as a spatial gradient of the particle velocity. This is significant because spatial gradients can be computed accurately using spectral methods, and don't require any additional storage. For comparison, the equivalent term in the Westervelt equation appears in the final form in equation (2), as a temporal gradient of the square of the pressure (Westervelt, 1963). However, using this expression requires the use of a finite difference scheme and storage of the pressure field at previous time steps.

The operator L in the pressure-density relation in equation (1) is an integro-differential operator that accounts for acoustic absorption that follows a frequency power law of the form $\alpha = \alpha_0\omega^y$. This type of absorption has been experimentally observed in human soft tissues, where $y$ is typically between 1 and 2 (Duck, 1990). The operator has two terms both dependent on the fractional Laplacian and is given by (Chen and Holm, 2004; Treeby and Cox, 2010b)

$$\mathrm{L} = \tau \frac{\partial}{\partial t}\left(-\nabla^2\right)^{\frac{y}{2}-1} + \eta\left(-\nabla^2\right)^{\frac{y+1}{2}-1} \tag{3}$$

Here $\tau$ and $\eta$ are absorption and dispersion proportionality coefficients given by $\tau = -2\alpha_0 c_0^{y-1}$ and $\eta = 2\alpha_0 c_0^{y}\tan(\pi y/2)$, where $\alpha_0$ is the power law prefactor in Np (rad /s)$^{-y}$ m$^{-1}$ and $y$ is the power law exponent. The two terms in L separately account for power law absorption and dispersion for $0 < y < 3$ and $y \neq 1$ (Treeby and Cox, 2010b, 2011).

## 2.2 Discrete equations

Following Tabei et al. (2002) and Treeby et al. (2012), the continuous governing equations given in the previous section can be discretised using the $k$-space pseudospectral method. If the mass conservation equation and the pressure-density relation given in equation (1) are solved together, the $()\cdot\nabla\rho_0$ terms cancel each other, so they are not included in the discrete equations to improve computational efficiency. The mass and momentum conservation equations in equation (1) written in discrete form then become

$$\frac{\partial}{\partial \xi} p^n = \mathcal{F}^{-1}\left\{ i k_\xi\, \kappa\, e^{i k_\xi \Delta\xi/2} \mathcal{F}\{p^n\}\right\} \tag{4a}$$

$$u_\xi^{n+\frac{1}{2}} = u_\xi^{n-\frac{1}{2}} - \frac{\Delta t}{\rho_0}\frac{\partial}{\partial \xi}p^n + \Delta t\, \mathrm{F}_\xi^{\mathrm{n}} \tag{4b}$$

$$\frac{\partial}{\partial \xi} u_\xi^{n+\frac{1}{2}} = \mathcal{F}^{-1}\left\{ i k_\xi\, \kappa\, e^{-i k_\xi \Delta\xi/2} \mathcal{F}\left\{u_\xi^{n+\frac{1}{2}}\right\}\right\} \tag{4c}$$

$$\rho_\xi^{n+1} = \frac{\rho_\xi^n - \Delta t \rho_0 \frac{\partial}{\partial\xi} u_\xi^{n+\frac{1}{2}}}{1 + 2\Delta t \frac{\partial}{\partial \xi} u_\xi^{n+\frac{1}{2}}} + \Delta t\, \mathrm{M}_\xi^{\mathrm{n}+\frac{1}{2}} \tag{4d}$$

Equations (4a) and (4c) are spatial gradient calculations based on the Fourier collocation spectral method, while equations (4b) and (4d) are update steps based on a $k$-space corrected finite difference scheme (Tabei et al., 2002). These equations are repeated for each Cartesian direction in $\mathbb{R}^n$ where $\xi = x$ in $\mathbb{R}^1$, $\xi = x, y$ in $\mathbb{R}^2$, and $\xi = x, y, z$ in $\mathbb{R}^3$. Here $\mathcal{F}\{\ldots\}$ and $\mathcal{F}^{-1}\{\ldots\}$ denote the forward and inverse spatial Fourier transforms over $\mathbb{R}^n$, $i$ is the imaginary unit, $\Delta t$ is the size of the time step and $k_\xi$ represents the set of wavenumbers in the $\xi$-direction defined according to

$$k_\xi = \begin{cases} \left[-\frac{\mathrm{N}_\xi}{2}, -\frac{\mathrm{N}_\xi}{2}+1, \ldots, \frac{\mathrm{N}_\xi}{2}-1\right]\dfrac{2\pi}{\Delta\xi\,\mathrm{N}_\xi} & \text{if } \mathrm{N}_\xi \text{ is even} \\ \left[-\frac{(\mathrm{N}_\xi-1)}{2}, -\frac{(\mathrm{N}_\xi-1)}{2}+1, \ldots, \frac{(\mathrm{N}_\xi-1)}{2}\right]\dfrac{2\pi}{\Delta\xi\,\mathrm{N}_\xi} & \text{if } \mathrm{N}_\xi \text{ is odd} \end{cases} \tag{5}$$

Here $\mathrm{N}_\xi$ and $\Delta\xi$ are the number and spacing of the grid points in the $\xi$-direction assuming a regular Cartesian grid. The $k$-space operator $\kappa$ in equation (4) is used to correct for the numerical dispersion introduced by the finite-difference time step, and is given by $\kappa = \mathrm{sinc}(c_{\mathrm{ref}} k \Delta t/2)$, where $k = |\mathbf{k}|$ is the scalar wavenumber, and $c_{\mathrm{ref}}$ is a single reference value of the sound speed (see discussion in Treeby et al. (2012) for further details).

The acoustic density and the mass source term (which are physically scalar quantities) are artificially divided into Cartesian components to allow an anisotropic perfectly matched layer (PML) to be applied. For the simulations presented here, Berenger's original split-field formulation of the PML is used (Berenger, 1996) as described in Tabei et al. (2002). The exponential terms $e^{\pm i k_\xi \Delta\xi/2}$ within equations (4a) and (4c) are spatial shift operators that translate the result of the gradient calculations by half the grid point spacing in the $\xi$-direction. This allows the components of the particle velocity to be evaluated on a staggered grid. Note, the ambient density $\rho_0$ in equation (4b) is understood to be the ambient density defined at the staggered grid points. The superscripts $n$ and $n+1$ denote the function values at current and next time points and $n-\frac{1}{2}$ and $n+\frac{1}{2}$ at the time-staggered points. The time-staggering arises because the update steps, equations (4b) and (4d), are interleaved with the gradient calculations, equations (4a) and (4c). An illustration of the staggered grid scheme is shown in Figure 2.

The corresponding pressure-density relation written in discrete form is given by

$$p^{n+1} = c_0^2\left(\rho^{n+1} + \frac{B}{2A}\frac{1}{\rho_0}\left(\rho^{n+1}\right)^2 - \mathrm{L_d}\right) \tag{6}$$

where the total acoustic density is given by $\rho^{n+1} = \sum_\xi \rho_\xi^{n+1}$ and $\mathrm{L_d}$ is the discrete form of the power law absorption term which is given by (Treeby and Cox, 2010b)

$$\mathrm{L_d} = \tau\mathcal{F}^{-1}\left\{k^{y-2}\,\mathcal{F}\left\{\frac{\partial\rho^n}{\partial t}\right\}\right\} + \eta\mathcal{F}^{-1}\left\{k^{y-1}\,\mathcal{F}\left\{\rho^{n+1}\right\}\right\} \tag{7}$$

To avoid needing to explicitly calculate the time derivative of the acoustic density (which would require storing a copy of at least $\rho^n$ and $\rho^{n-1}$ in memory), the temporal derivative of the acoustic density is replaced using the linearised mass conservation equation $d\rho/dt = -\rho_0\nabla\cdot\mathbf{u}$, which yields

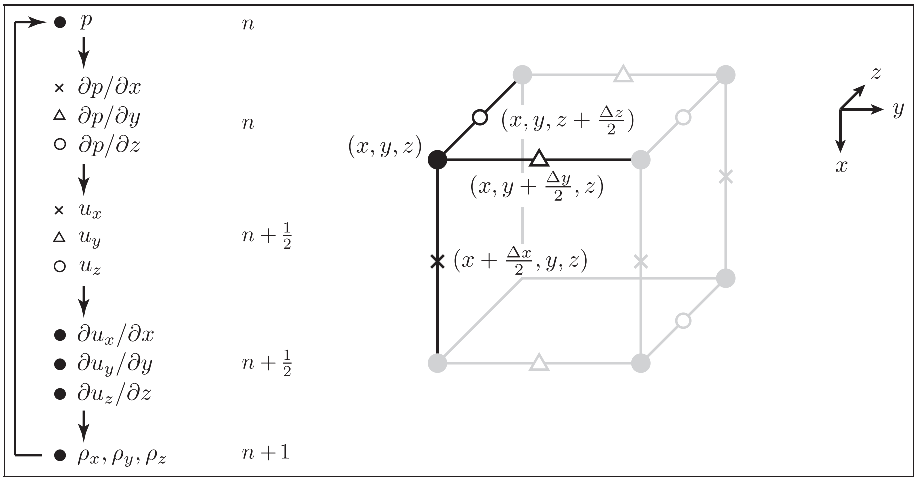

**Figure 2.** Schematic showing the computational steps in the solution of the coupled first-order equations using a staggered spatial and temporal grid in 3D. Here $\partial p/\partial x$ and $u_x$ are evaluated at grid points staggered in the $x$-direction (crosses), $\partial p/\partial y$ and $u_y$ are evaluated at grid points staggered in the $y$-direction (triangles) and $\partial p/\partial z$ and $u_z$ are evaluated at grid points staggered in the $z$-direction (open circles). The remaining variables are evaluated on the regular grid points (dots). The time-staggering is denoted $n$, $n+\frac{1}{2}$, and $n + 1$.

$$\begin{aligned} \mathrm{L_d} = & -\tau\,\mathcal{F}^{-1}\left\{k^{y-2}\,\mathcal{F}\left\{\rho_0 \sum\nolimits_\xi \frac{\partial}{\partial \xi} u_\xi^{n+\frac{1}{2}}\right\}\right\} \\ & + \eta\,\mathcal{F}^{-1}\left\{k^{y-1}\,\mathcal{F}\left\{\rho^{n+1}\right\}\right\} \end{aligned} \tag{8}$$

Further details about the formulation, stability and accuracy of the $k$-space scheme can be found in Mast et al. (2001), Tabei et al. (2002), Cox et al. (2007), Treeby and Cox (2010b) and Treeby et al. (2012).

## 3 Implementation of the *k*-space pseudospectral method for distributed clusters

### *3.1 Overview*

The $k$-space and pseudospectral methods gain their advantage over finite difference methods due to the global nature of the spatial gradient calculations. This permits the use of a much coarser grid for the same level of accuracy. However, even using spectral methods, the computational and memory requirements for the nonlinear HIFU problems discussed in Section 1 are still considerable, and in most cases, significantly exceed the resources of a single workstation. In this context, the development of an efficient numerical implementation that partitions the computational cost and memory usage across a large-scale parallel computer is desired. However, the global nature of the gradient calculation, in this case using the 3D fast Fourier transform (FFT), introduces additional challenges for the development of an efficient parallel code. Specifically, while the FDTD method only requires small quantities of data to be exchanged between the processes responsible for adjacent portions of the domain, performing a FFT requires a globally synchronising all-to-all data exchange. This global communication can become a significant bottleneck in the execution of spectral models.

Fortunately, considerable effort has already been devoted to the development of distributed memory FFT libraries that show reasonable scalability of up to tens of thousands of processing cores (Frigo and Johnson, 2005; Pekurovsky, 2012). These libraries have found particular utility in turbulence simulations where grid sizes of up to $4096 \times 4096 \times 4096$ have been used (Yeung et al., 2012). They have also been considered in previous implementations of linear $k$-space pseudospectral models using grid sizes of up to $512 \times 512 \times 512$ (Daoud and Lacefield, 2009; Tillett et al., 2009). The challenges for the current work were thus to determine how best to exploit these libraries within the context of the nonlinear $k$-space model, how to maximise the grid sizes possible for a given total memory allocation, how best to manage the generated output data and how to maximise performance and scalability.

The execution of the $k$-space pseudospectral model described in Section 2 can be divided into three phases: pre-processing, simulation and post-processing. During the pre-processing phase, the input data for the simulation is generated. This involves defining the domain discretisation based on the physical domain size and maximum frequency of interest, defining the spatially varying material properties (e.g. using a CT scan of the patient (Schneider et al., 1996)), defining the properties of the ultrasound transducer and drive signal and defining the desired output data (e.g. the peak positive pressure or time-averaged acoustic intensity in the region of the HIFU target (ter Haar et al., 2011)). The simulation phase involves reading the input data,

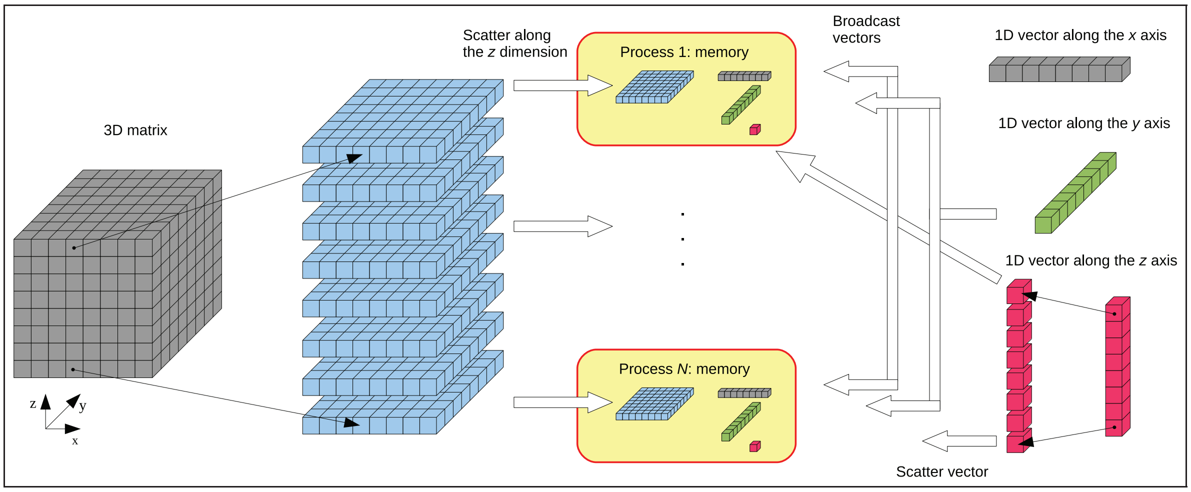
3D matrix
Scatter along the z dimension
Process 1: memory
Process N: memory
Broadcast vectors
1D vector along the x axis
1D vector along the y axis
1D vector along the z axis
Scatter vector
z
y
x

**Figure 3.** Illustration of the 1D slab decomposition used to partition the 3D domain within a distributed computing environment. The 3D matrices are partitioned along the z-dimension and distributed over *P* MPI processes. 1D vectors oriented along the *x*- and *y*-dimensions are broadcast, while the vectors along the z-dimension are scattered. All scalar variables are broadcast and replicated on each process.

running the actual simulation following the discrete equations discussed in Section 2.2, and storing the output data. The post-processing phase involves analysing the (potentially large) output files and presenting this data in a human-readable form. Here, the discussion is focused primarily on the parallel implementation of the simulation phase. Some discussion of the pre- and post-processing stages is given in Section 5.

The discrete equations solved during the simulation phase are given in equations (4a) to (4d) and equation (6). Examining these equations, the data stored in memory during the simulation phase comprises of twenty-one real 3D matrices defined in the spatial domain, and three real and three complex 3D matrices defined in the spatial Fourier domain (in addition to vector and scalar values). The 3D matrices contain the medium properties at every grid point, the time-varying acoustic quantities, the derivative and absorption operators and temporary storage. The operations performed on these datasets include 3D FFTs, element-wise matrix operations, injection of the source signal and the collection of output data.

The implementation of the discrete equations was written in C++ as an extension to the open-source k-Wave acoustics toolbox (Treeby and Cox, 2010a; Treeby et al., 2012). The standard message passing interface (MPI) was used to perform all interprocess communications, the MPI version of the FFTW library was used to perform the Fourier transforms (Frigo and Johnson, 2005) and the input/output (I/O) operations were performed using the hierarchical data format (HDF5) library. To maximise performance, the code was also written to exploit single instruction multiple data (SIMD) instructions such as streaming SIMD extensions (SSE). Further details of the implementation are given in the following sections.

## 3.2 Domain decomposition and the FFT

To divide the computational domain across multiple interconnected nodes in a cluster, a one-dimensional domain decomposition approach was used in which the 3D domain is partitioned along the *z*-dimension into 2D slabs. The slabs are then distributed over *P* MPI processes, where each MPI process corresponds to one physical CPU core. The total number of processes is constrained by $P \leq N_z$, where $N_z$ is the number of grid points in the *z*-dimension (and thus the number of slabs). This decomposition approach was used as it is directly supported by the FFTW library, while other approaches, such as 2D partitioning are not.

Figure 3 shows how the various spatial data structures are distributed to processes. For each 3D matrix there are a maximum of $\lceil N_z/P \rceil$ 2D slabs stored on each process. For 1D quantities oriented along the *z*-axis, the data is partitioned and scattered over the processes in a similar manner. For 1D quantities oriented along either the *x*- or *y*-axis (and for scalar quantities), the data is broadcast and replicated on every process. The exception is for the source and sensor masks, which list the grid indices where the input data is defined and where the output data is collected. These are distributed such that individual processes are assigned the portion of the list that corresponds to parts of the source and/or sensor that fall within its local section of the domain. As the source and sensor masks do not usually cover the whole domain, many processes are likely to receive no source or sensor related data.

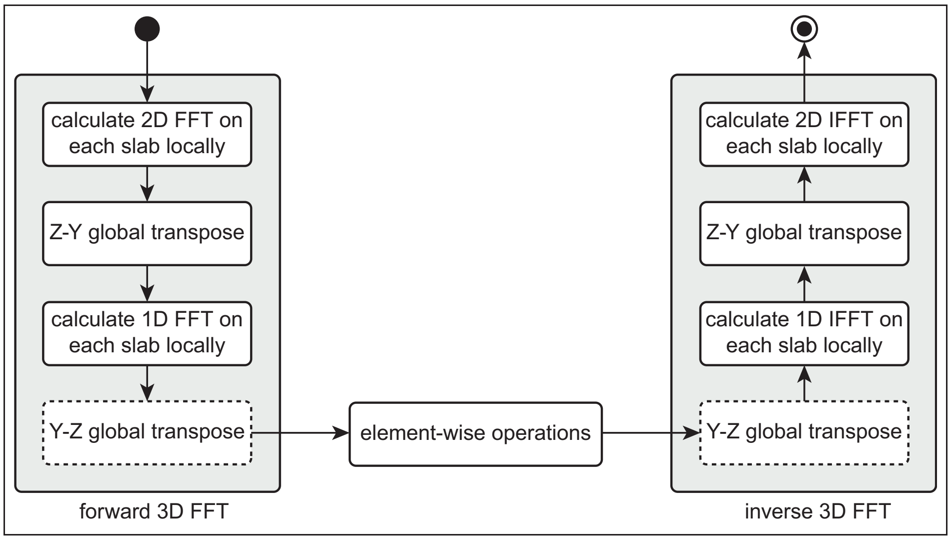
calculate 2D FFT on each slab locally
Z-Y global transpose
calculate 1D FFT on each slab locally
Y-Z global transpose
forward 3D FFT
element-wise operations
Y-Z global transpose
calculate 1D IFFT on each slab locally
Z-Y global transpose
calculate 2D IFFT on each slab locally
inverse 3D FFT

**Figure 4.** Chain of operations needed to calculate spatial gradients using the Fourier pseudospectral method. First, the 3D forward FFT is calculated, then element-wise operations within the spatial frequency domain are applied. Finally, the result is transformed back to the spatial domain using an inverse 3D FFT. The transposes depicted in the dashed boxes can be omitted if the element-wise matrix operations are performed in the transposed domain.

The calculation of the spatial gradients following equations (4a) and (4c) involves performing a 3D FFT, followed by one or more element-wise matrix operations, followed by an inverse 3D FFT. Using the domain decomposition scheme outlined above, each 3D FFT is executed by first performing a series of 2D FFTs in the $xy$-plane (i.e. on each slab) using local data. This is then followed by an all-to-all global communication to perform a $z\leftrightarrow y$ transposition. This step is necessary as the FFT can only be performed on local data (it cannot stride across data belonging to multiple processes). The global transposition is then followed by a series of 1D FFTs performed in the transposed $z$-dimension, followed by another global transposition from $y\leftrightarrow z$ to return the data to its original layout. This chain of operations is illustrated in Figure 4. In this decomposition, the main performance bottleneck is the two global transpositions required per FFT.

Examining Figure 4, it is apparent that the last global $y\leftrightarrow z$ transposition of the forward FFT is paired with an identical but reverse transposition immediately after the element-wise operations. As the intervening operations are independent of the order of the individual elements, it is possible to eliminate these two transpositions such that operations in the spatial frequency domain are performed in transformed space (Frigo and Johnson, 2012). This has a significant effect on performance, with compute times reduced by 35-40% depending on the number of processes used. Note, in this case, variables defined in the spatial frequency domain must instead be partitioned in transformed space along the $y$-dimension, with the total number of MPI processes constrained by $P \leq \min(N_y, N_z)$. To avoid having idle processes during calculations involving either regular or transposed data, the number of processes $P$ should ideally be a chosen to be a divisor of both $N_y$ and $N_z$.

As the input data to the forward FFT is always real, the output of the Fourier transform in the first dimension is symmetric. Computational efficiency can thus also be improved by using the real-to-complex and complex-to-real FFTW routines which only calculate and return the unique part of the transform. To further improve performance, the element-wise matrix operations executed in between these transforms, as well as those defined in equations (4b) and (4d), were merged into compute kernels optimised to maximise temporal data locality and performance. In some cases, the latter involved manually inserting calls to SIMD intrinsic functions into loop structures that the compiler was unable to vectorise otherwise.

## 3.3 Input and output

The open-source HDF5 library was chosen to manipulate the input and output files because of its ability to organise complex datasets in heterogeneous computing environments. This library is available on most supercomputers and is also supported by numerical computing languages such as MATLAB and Python which is useful for pre- and post-processing. The HDF5 library provides two interfaces: serial and parallel. The serial interface targets shared memory computers and assigns a single thread to be responsible for all I/O. This interface was used to generate the input files during the pre-processing phase using a single large-memory node. The parallel interface targets clusters and essentially provides a user-friendly wrapper to the low level MPI-I/O library. This allows multiple processes to share the same file and read or write to it simultaneously. This interface was used during the simulation phase as it provides much higher I/O performance than either serialised accesses or master-slave architectures (where a single process serves all I/O requests). The parallel HDF5 interface also allows for two different I/O access patterns: independent and collective. In most cases, the collective mode is preferred as it enables the HDF5 runtime to rearrange and reshape data from different parallel processes before writing it to disk. This mode was used for all I/O operations during the simulation phase, the only exception being when writing scalar values to the output file.

Within the input and output files, all datasets were stored as three-dimensional arrays in row-major order, with dimensions defined by the tuple ($N_x$, $N_y$, $N_z$). For scalars and 1D and 2D arrays, the unused tuple dimensions were set to 1. For example, scalar variables were stored as arrays of dimension (1, 1, 1), 1D vectors oriented along the $y$-axis were stored as arrays of dimension (1, $N_y$, 1), etc. For datasets containing complex data, the lowest used dimension was doubled and the data stored in interleaved format. Additional information about the simulation (for example, the control flags and parameters) was stored within the file header.

To maximise I/O performance, the datasets within the input and output files were stored in a chunked layout, where each dataset is divided into multiple blocks or chunks that can be accessed individually. This is particularly useful for improving the throughput of partial I/O, where only portions of a dataset are accessed at a time. The use of chunking also provides a convenient way to overcome one of the current MPI limits, namely the 2 GB maximum size of a message (this is due to the routine headers in the MPI standard being defined using signed integers). Without chunking, this limit is easy to exceed, particularly during MPI gather operations where hundreds of small messages are aggregated. In addition to partial I/O, chunking enables the use of on-the-fly data compression. This is particularly beneficial for the input file, as there are often large areas of the domain with similar material properties, for example, the layer of water or coupling medium between the transducer and patient. These homogeneous regions give rise to good compression ratios, and thus reduce the amount of communication necessary when loading the input file. However, while both the serial and parallel HDF5 interfaces can read compressed datasets, only the serial interface can write such datasets. Thus compression was only used for the input files.

For the 3D datasets, the chunk size was chosen to be a single 2D slab matching the decomposition discussed in Section 3.2. This is the smallest portion of data read by each MPI process. Each slab is only ever accessed by one process at a time, and is usually a reasonable size in terms of I/O efficiency. For the 2D and 1D quantities, the data was divided into chunks of fixed size along the lowest used dimension. A chunk size of 8 MB was found to give reasonable I/O performance. Note, it is possible to further tune the chunk size for different size datasets to maximise I/O performance on a given platform. This can be useful on parallel cluster file systems that use hundreds of disk drives spread over many nodes, in conjunction with complex internode communication patterns.

## 3.4 Simulation stages

An overview of the stages within the simulation phase of the parallel implementation of the $k$-space pseudospectral model is shown in Figure 5. When the simulation begins, parameters such as the domain size and the number of time steps are loaded using a HDF5 broadcast read. In this operation, all processes select the same item to be read from the file. The HDF5 library then joins all I/O requests such that the data is only physically read from disk once and then broadcast over all processes using an MPI routine. Next, the 1D decomposition of the domain is calculated as discussed in Section 3.2, and memory for the various simulation and temporary quantities is allocated. Note, the use of SIMD instructions imposes several requirements on the data layout and its memory alignment. Firstly, all multidimensional matrices must be stored as linear arrays in row-major order. Secondly, complex quantities must be stored in interleaved format. Finally, data structures must be allocated using an FFTW memory allocation routine that ensures they are aligned on 16 B boundaries.

After memory allocation, the contents of the input file are loaded and distributed over all processes. The 3D datasets (e.g. the medium properties) are loaded in

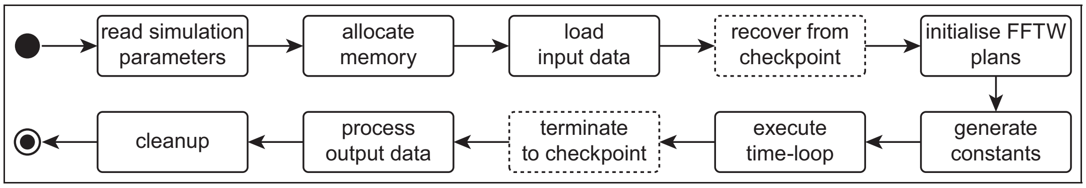

**Figure 5.** Overview of the stages within the simulation phase of the parallel implementation of the $k$-space pseudospectral model.

chunks, with each process identifying its local portion of the domain and invoking a collective HDF5 read operation. The 1D vectors that are scattered are loaded in a similar fashion, while 1D vectors and scalar values that are replicated are read using a HDF5 broadcast read. The source and sensor masks are loaded in chunks and broadcast to all processes, with each process extracting the grid indices that fall within its portion of the domain. Once the distribution of the source mask is known, the drive signal for the transducer is then sent to the relevant processes. Finally, the distribution of the sensor mask is used to allocate appropriate buffers to store the output data. These buffers are eventually flushed into the output file.

Next, execution plans for the FFT are generated. This step is required by FFTW prior to performing the first transform of a particular rank, size and direction. It involves the library trying several different FFT implementations with the objective of finding the fastest execution pathway on the current hardware (Frigo and Johnson, 2005). Consideration is given to many architectural aspects including the CPU, the memory system and the interconnection network. Depending on the domain size, this step can take a considerable amount of time. However, as it is only executed once and there are many thousands of FFTs in a typical simulation, the benefit of having a fast implementation is usually significant (this is discussed further in Section 4.2). Unfortunately, while it is possible to save plans between code executions, there is usually little benefit in doing so. This is because the plans are problem-size specific, and depend on the number and distribution of cores within the parallel system being used. On shared clusters in particular, the distribution of cores is likely to change between runs according to the cluster utilisation and management system.

After the FFT plans are generated, simulation constants such as the $k$-space operator ($\kappa$) and the absorption parameters ($\tau$, $\eta$, $k^{y-2}$, $k^{y-1}$) are generated. The simulation time-loop then begins following equations (4a) to (4d) and equation (6). For each time step, there are six forward and eight inverse FFTs. There are two fewer forward FFTs as the three spatial components in equation (4a) share the same $\mathcal{F}\{p^n\}$. The source injection is implemented as an operation that updates the value of the acoustic pressure or velocity at the relevant grid points within the domain. In a similar vein, the output data is collected by storing the acoustic parameters at the grid points specified by the sensor mask. Aggregate quantities (e.g. the peak positive pressure) are kept in memory until the simulation is complete, while time-varying quantities are flushed to disk at the end of each time step using a collective HDF5 write routine.

When running larger simulations, a checkpoint-restart capability is also used. This allows large simulations to be split into several phases, which is useful for staying within wall-clock limitations imposed on many clusters, in addition to providing a degree of fault tolerance. During checkpointing, an additional HDF5 output file is created which stores the current values of the time-varying acoustic quantities as well as the aggregated output quantities. Restart is performed in the same way as a new simulation, however, after the input file has been loaded, the checkpoint file is opened, and the acoustic quantities and the aggregated values are restored.

## 4 Performance evaluation

### 4.1 Benchmark platform

To evaluate the utility of the implemented $k$-space pseudospectral model in the context of large-scale nonlinear ultrasound simulations for HIFU, a number of performance metrics were investigated. These included the strong and weak scaling properties of the code, the absolute simulation time and cost, and the effect of the underlying hardware architecture. The tests were performed using the VAYU supercomputer run by the National Computational Infrastructure (NCI) in Australia. This system comprises 1492 nodes, each with two quad-core 2.93 GHz Intel Nehalem architecture Xeon processors and 24 GB of memory configured in a non-uniform memory access (NUMA) design. The compute nodes are connected to the cluster via an on-board infiniband interface with a theoretical bandwidth of 40 Gb/s, while the interconnection network consists of four highly integrated 432-port infiniband switches. This relatively simple network design reduces the impact of process placement on interprocess network bandwidth and latency. The I/O disk subsystem is physically separated from the compute nodes. The Lustre parallel distributed file system is used to manage 832 TB of disk space distributed over 1040 disk drives. This configuration offers very high bandwidth, however, latency for small I/O transactions (e.g. storing less than 1 MB of data) can be relatively high. Also, as the I/O subsystem is shared amongst all users, notable performance fluctuations can occasionally be observed. VAYU, similar to many other clusters, runs a Linux based operating system controlled by the OpenPBS batch queuing systems. Compute cores granted to the job are always dedicated and simulation cost is determined in terms of service units (wall-clock time multiplied by the number of CPU cores used).

The performance metrics were evaluated using a common set of benchmarks. The benchmark set was designed to cover a wide range of domain sizes, from small simulations that can be run on desktop systems, up to large-scale simulations that approach the limits of what is currently feasible using a supercomputer. The simulations accounted for nonlinear wave propagation in an absorbing medium in which all material properties were heterogeneous. The time varying output pressure was recorded over a 2D $xy$-plane centred in the $z$-direction. The grid sizes increased from $2^{24}$ grid points ($256 \times 256 \times 256$) to $2^{34}$ grid points ($4096 \times 2048 \times 2048$), where each successive benchmark was doubled in size, first by doubling the grid size in the $x$-dimension, then the $y$-dimension, then the $z$-dimension, and so on. The benchmarks were executed on VAYU using different numbers of compute cores ranging from 8 (one node) to 1024 (128 nodes) in multiples of 2. The minimum and maximum number of cores used for a particular grid size were limited by the available memory per core from the bottom (24 GB per node or 3 GB per core), and the size of the simulation domain from the top ($P \leq \min(N_y, N_z)$). Compute times were obtained by averaging over 100 time-steps, excluding pre- and post-processing.

## 4.2 Strong scaling

Strong scaling describes how the execution time changes when using an increasing number of compute cores for a fixed problem size. Ideally, whenever the number of compute cores is doubled, the execution time should reduce by a factor of 2. However, in practice, all but embarrassingly parallel applications include some level of overhead that causes them to eventually reach a point where increasing the number of compute cores does not reduce the execution time, or can even make it worse.

Figures 6(a) and 6(b) show the strong scaling results obtained for the eleven different benchmark problem sizes using two different FFTW planing flags, `FFTW_MEASURE` and `FFTW_EXHAUSTIVE`. These flags determine the extent to which FFTW attempts to find the best FFT implementation during the plan generation phase. The flag `FFTW_MEASURE` is the default, while the flag `FFTW_EXHAUSTIVE` triggers a much more extensive search covering the full range of options implemented by the FFTW library. When using the `FFTW_MEASURE` flag, the results show almost ideal strong scaling for domain sizes up to $2^{31}$ grid points ($2048 \times 1024 \times 1024$) and core counts up to 256. Within this region, all the curves have approximately equal gradients corresponding to a speed-up of roughly 1.7x when doubling the number of cores. Beyond 256 cores, acceptable scaling is obtained for many cases, however, there is virtually no improvement in execution time for grid sizes of $2^{29}$ and $2^{30}$ grid points. Further increasing the core count to 1024 leads to significantly worse performance, with the exception of the grid size of $2^{30}$ grid points ($1024 \times 1024 \times 1024$) where, interestingly, the performance increases markedly.

The erratic performance of FFTW at large core counts when using the `FFTW_MEASURE` flag is due to differences in the communication pattern chosen during the planning phase. Specifically, using the integrated performance monitoring (IPM) profiler, it was found that in most cases, the global transposition selected by FFTW was based on a simple all-to-all MPI communication routine which exchanges $P(P-1)$ messages amongst $P$ compute cores. This becomes increasingly inefficient when the message size drops into the tens of kB range and the number of messages rises into the millions. This is the case for the very large simulations on high core counts shown in Figure 6(a). However, FFTW also includes other communication algorithms, including one that uses a sparse communication pattern. This routine was selected by `FFTW_MEASURE` for the simulation with $2^{30}$ grid points and 1024 cores, leading to significant performance gains.

The sparse communication pattern used by FFTW combines messages local to the node into a single message before dispatch. In the case of VAYU, this means that 8 messages are combined together within each node before sending to other nodes. This decreases the number of messages by a factor of 64 and also increases the message size. When using `FFTW_EXHAUSTIVE`, the sparse communication pattern was always selected. As shown in Figure 6(b), the impact of this on performance is considerable. Almost all anomalies are eliminated, and the scaling remains close to ideal over the whole range of grid sizes and core counts considered. This is consistent with profiling data, which revealed that 30-40% of the execution time was associated with FFT communication, another 30-40% due to FFT operations and the remaining 15-20% due to element-wise matrix operations. These results indicate that given large enough grid sizes, reasonable scalability to even larger numbers of compute cores should be possible. Small deviations in the scaling results are likely to be caused by the degree of locality in the process distribution over the cluster, as well as by process interactions when performing I/O operations. For comparison, the time to execute the FFTW planning phase using `FFTW_MEASURE` for a grid size of $2^{33}$ grid points ($2048 \times 2048 \times 2048$) was 239 s when using 512 cores, and 164 s using 1024 cores. The corresponding times using `FFTW_EXHAUSTIVE` were 999 s and 834 s. This is equivalent to the time taken to compute approximately 100 time steps. As a typical simulation of this scale requires tens of thousands of time steps, the planning times represent a very small proportion of the total simulation time.

In addition to the benchmark set considering nonlinear wave propagation in a heterogeneous and absorbing medium, additional benchmarks were performed considering (1) nonlinear wave propagation in a homogeneous and absorbing medium, and (2) linear wave propagation in a homogeneous and lossless medium. The strong-scaling results for both cases were

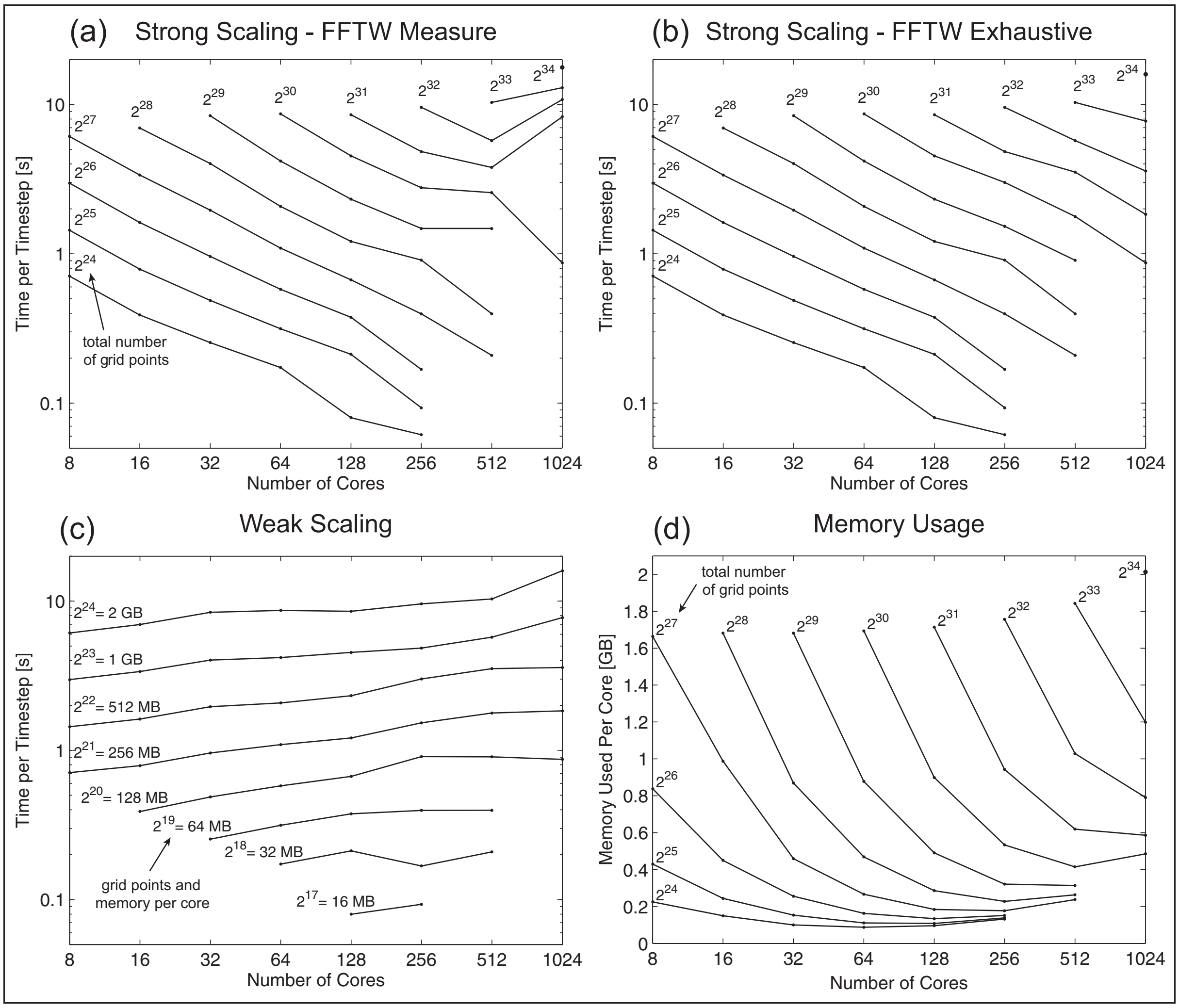

**Figure 6.** Performance results for nonlinear ultrasound simulations in heterogeneous and absorbing media. (a), (b) Strong scaling showing the variation of the execution time with the number of CPU cores for a fixed grid size. The domain size varies from $2^{24}$ ($256^3$) grid points to $2^{34}$ (4096×2048×2048) grid points. The plots clearly show the difference in strong scaling between `FFTW_MEASURE` and `FFTW_EXHAUSTIVE` flags used for planning the execution of FFTs and related global communications. (c) Weak scaling showing the potential of more cores to solve bigger problems in the same wall-clock time. The curves show the working dataset size per core growing from 16 MB ($2^{17}$ grid points) up to 2 GB ($2^{24}$ grid points). (d) Memory usage per core for the strong scaling results given in part (b).

similar to those given above. This is not surprising given that the FFT is the dominant operation in all test cases. In terms of absolute performance, the results for case (1) benefit from both a reduced memory footprint and a reduced memory bandwidth requirement. Specifically, because the medium is homogeneous, the medium parameters can be described using scalar variables that are easily cached, rather than large 3D matrices that are usually read from main memory each time they are used. This reduces the memory requirements by approximately 30% and the execution times by 5-10%. For case (2), an additional advantage comes from a reduction in the number of FFTs performed within each time step (ten instead of fourteen). This reduction is mirrored almost exactly in the observed execution times, which were typically 30% less than the execution times for an absorbing medium.

## *4.3 Weak scaling*

Weak scaling describes how the execution time changes with the number of compute cores when the problem size (e.g. number of grid points) per core remains fixed. In the ideal case, doubling the number of compute cores should enable a problem twice as big to be solved in the same wall-clock time. However, for the *k*-space pseudospectral model, the computational work grows slightly greater than linear with $O(N \log(N))$ in the number of grid points due to the FFT, while the number of point-to-point communication events increases in the parallel implementation with $O(P^2)$ in the number of processors due to the matrix transpositions. As a consequence, the weak scaling curves will always grow (for a small number of cores, the computation time will be dominant while for large numbers of cores the communication overhead will take over). Figure 6(c) shows the weak scaling results for the benchmark set defined in Section 4.1 when using the `FFTW_EXHAUSTIVE` planning flag. Despite the fact that the cost of the underlying all-to-all communication step grows with the square of the number of processes, the graph shows relatively good weak scaling performance. The trends suggest that simulations with even larger grid sizes could be solved with reasonable efficiency using higher numbers of compute cores.

## 4.4 Memory usage

Figure 6(d) shows how the memory usage per core changes in-line with the strong scaling results given in Figure 6(b). Whenever the number of cores is doubled, the domain is partitioned into twice as many 2D slabs. Theoretically, the memory required by each core should be halved. However, in practice, there is an overhead associated with the domain decomposition that grows with the number of cores, and eventually becomes dominant. This behaviour is clearly observed for almost all the simulation sizes in the benchmark set. The overhead is comprised of several components, each with a different origin.

First, the MPI runtime allocates a significant amount of memory for communication buffers. When the MPI library is not restricted, it allocates as many communication buffers for each process as there are distinct messages to be received (one for each sender). Moreover, if the sparse communication pattern is used, additional scratch space must be allocated where smaller messages can be combined and separated on send and receive. For simulations using high process counts where the memory used per core for the local partition is quite low (tens of MB), the memory allocated for MPI communication buffers can become the dominant component.

Second, in addition to communication buffers, when the domain is partitioned into more parts over an increasing number of processes, the total memory consumed by locally replicating scalar variables and 1D arrays is increased. Moreover, additional memory is needed for storing the code itself. While the size of the compiled binary file is only on the order of 20 MB, a private copy is needed for every process. Thus, when 1024 processes are used, storing the code consumes more than 20 GB of memory. For small simulations, this can become a significant portion of the total memory consumption.

## 4.5 Simulation cost and execution time

For the user, another important metric that must be considered when running large-scale simulations is their financial cost. For this purpose, the simulation cost is defined as the product of the wall-clock execution time and the number of compute cores used. On VAYU, this cost is expressed in terms of service units (SUs). These are directly related to the number of core-hours used, scaled by a few other factors such as the priority the user assigns to the job. SUs represent an accountant's view on the parallel economy. On a large shared parallel computer, each user is typically allocated a share of the resource. On VAYU this corresponds to a certain number of SUs per quarter, and it is left to the user to determine how best to use this allocation. As scaling is never ideal, the more compute cores assigned to a job, the higher the effective cost. However, for time critical problems, using the highest possible number of cores may still be desirable.

Figure 7(a) shows the anticipated total simulation cost for the grid sizes in the benchmark set against the number of compute cores used. As the grid size increases, more time steps are also necessary to allow the ultrasound waves to propagate from one side of the domain to the other. Using the diagonal length of the domain and assuming a Courant-Friedrichs-Lewy (CFL) number of 0.2, the number of time steps grows from 2200 for a grid size of $256 \times 256 \times 256$ to 25,000 for a grid size of $4096 \times 2048 \times 2048$ grid points. The results in Figure 7(a) show that for a given grid size, the simulation cost remains fairly constant as a function of core count, with the ratio between the highest and lowest costs always less than 2 (this is to be expected given the strong-scaling results).

NCI, the owners of VAYU, charge US\$ 0.1 per SU to commercial projects. Using this value, the approximate cost of each simulation in US dollars is also shown in Figure 7(a). If run to completion, the largest simulation performed would take around 4.5 days on 1024 compute cores and would cost slightly over US\$10,000. At this point in time, such a large simulation is clearly not routine. However, VAYU is now a relatively old system, and with continued price performance trends it is not impossible to see the cost of such a simulation dropping to a few hundred dollars within the next few years.

## 4.6 Comparison of different architectures

Thus far, only the performance of the MPI implementation of the *k*-space pseudospectral model running on the VAYU cluster has been considered. For comparison, Figure 7(b) illustrates the execution times per time step for two equivalent models implemented in (a) MATLAB and (b) C++ but parallelised for shared-

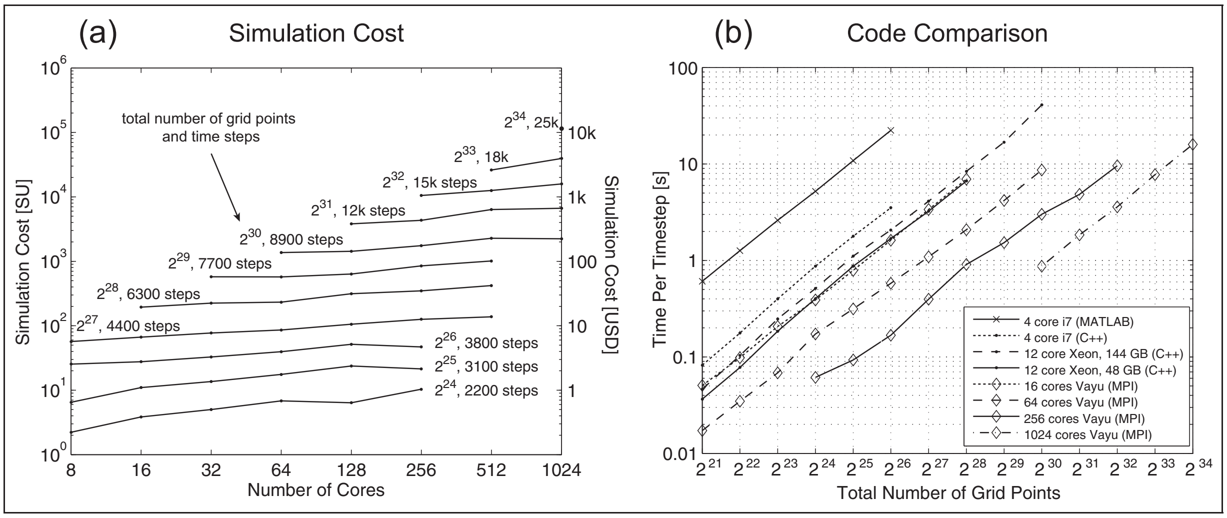

**Figure 7.** (a) The simulation cost in terms of service units (core-hours) displayed on the left *y*-axis and USD on the right *y*-axis. The number of time steps is derived from the time the wave needs to propagate along the diagonal length of the domain. (b) Execution time per time-step running different implementations of the *k*-space pseudospectral model on different machines.

memory architectures using OpenMP rather than MPI (some of the details of this implementation are discussed in Jaros et al. (2012)). For these comparisons, the benchmark set was extended to include smaller grid sizes starting at $2^{21}$ grid points ($128 \times 128 \times 128$). Three different computer configurations were considered: (1) a desktop machine with 12 GB of RAM and a single Intel Core i7 950 quad-core CPU running at 3.2 GHz, (2) a server with 48 GB of RAM and two Intel Xeon X5650 hex-core CPUs running at 2.66 GHz and (3) an identical server with 144 GB of RAM. The two server configurations differ in that the memory controller is not able to operate at 1333 MHz when serving 144 GB of RAM and drops its frequency to 800 MHz. The effect of this is to reduce the memory bandwidth by about 33%.

Each line in Figure 7(b) represents a different combination of computer configuration and implementation. The slowest performing combination is the MATLAB implementation running on the quad-core i7 system (this was the starting point for the development of the C++ codes (Treeby and Cox, 2010a)). Using the shared-memory C++ implementation on this machine reduces the execution time by a factor of about 8. Moving to the 12-core Xeon systems more than doubles this factor, although there are clear differences between the 48 GB and 144 GB configurations due to the fact that the model is memory bound. In comparison, to match the performance of the shared memory code running on the 12-core Xeon system, the MPI version of the code requires 16-cores on VAYU. The higher core-count needed by the MPI version of the code for equivalent performance reflects the fact that a non-trivial overhead is incurred since all communications between cores must pass through the MPI library (and in many cases, over the network). Of course, the benefit of the MPI implementation is the possibility of running the code over multiple nodes on a cluster, enabling both much larger domain sizes and much faster execution times. For example, for a grid size of $2^{26}$ grid points, the MPI code running on 256 cores is approximately two orders of magnitude faster than the MATLAB implementation on the quad-core i7 system, and one order of magnitude faster than the shared memory C++ implementation running on the 12-core Xeon systems.

## 5 Application example

To illustrate the utility of the developed nonlinear ultrasound model for solving real-world problems, a complete large-scale nonlinear ultrasound simulation representing a single HIFU sonication of the kidney was performed. The medium properties for the simulation were derived from an abdominal CT scan using the MECANIX dataset (OsiriX Imaging Software)

This was re-sampled using linear interpolation to give the appropriate resolution. The density of the tissue was calculated from Hounsfield units using the data from Schneider et al. (1996), and the sound speed was then estimated using the empirical relationship given by Mast (2000). The remaining material properties were assigned book values (Duck, 1990). The HIFU transducer was defined as a circular bowl with a width of 10 cm and a focal length of 11 cm. The shape of the transducer within the 3D Cartesian grid was defined using a 3D extension of the midpoint circle algorithm (Treeby and Cox, 2010a). The transducer was positioned behind the patient as shown in Figure 8(a), and

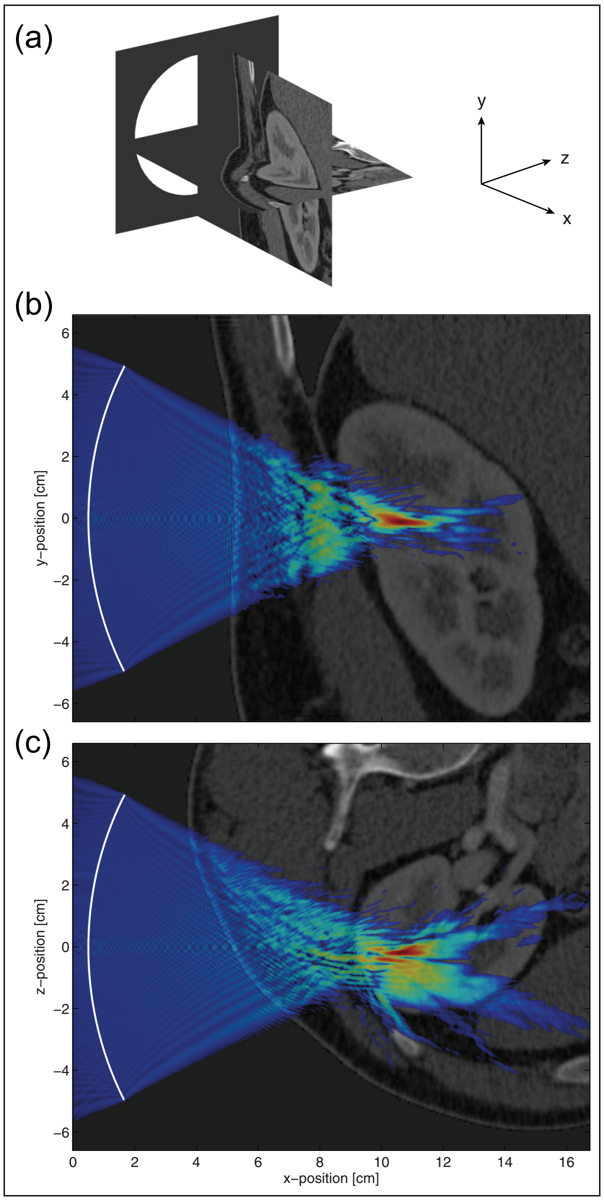

**Figure 8.** (a) Sagittal and transverse slices through the abdominal CT scan used to define the material properties for simulating a HIFU treatment of the kidney. The approximate position of the HIFU transducer is shown with a white circle. (b)-(c) Saggittal and transverse slices through the simulated distribution of maximum pressure overlaid onto the corresponding CT slices. The pressure distribution is displayed using a log-scale and is thresholded at -30 dB. The distortion of the HIFU focus due to the body wall and the fat layer surrounding the kidney is clearly visible.

was driven at 1 MHz by a continuous wave sinusoid. The acoustic intensity at the transducer surface was set to 2 W/cm$^2$ to simulate a treatment that would likely operate largely in a thermal regime (i.e. with minimal cavitation). Outside the body, the medium was assigned the properties of water (Duck, 1990). The total domain size was 17 cm $\times$ 14.3 cm $\times$ 14.3 cm and the grid spacing in each Cartesian direction was set to 93 $\mu$m, giving a total grid size of $2048 \times 1536 \times 1536$ grid points and a maximum supported frequency of 8 MHz (i.e. eight harmonics of the source frequency). The simulation length was set to 220 $\mu$s with a CFL number of 0.18, giving a total of 19,800 time steps. Simulations of this scale and complexity have not previously been possible.

The simulation was executed using 768 cores on VAYU with seven checkpoint-restart stages. The total wall-clock time was 31 hours and 20 minutes, and the total memory used was 780 GB. The compressed input file was 45 GB, while the output file was 450 GB. This comprised 36 GB to store the peak positive and peak negative pressure across the domain, and 414 GB to store the time varying pressure and particle velocity for 6 periods in steady state over a 45 mm $\times$ 30 mm $\times$ 30 mm region surrounding the HIFU focus. Transverse and sagittal slices through the peak positive pressure overlaid onto the corresponding CT data used to define the material properties are shown in Figure 8(b)-(c). The distortion of the ultrasound focus due to the body wall and the fat layer surrounding the kidney is clearly visible. These effects have been noted clinically, and remain a barrier to the application of HIFU in the kidney (Illing et al., 2005). Thus, one possible future application of the developed *k*-space model would be a systematic investigation into the conditions necessary for viable HIFU ablation in the kidney (for example, the maximum thickness of the fat layer). In any case, the example serves to illustrate the utility of the implementation.

## 6 Summary and discussion

A full-wave model for simulating the propagation of nonlinear ultrasound waves through absorbing and heterogeneous media is presented in the context of model-based treatment planning for HIFU. The governing equations are discretised using the *k*-space pseudospectral method. The model is implemented in C++ using MPI to enable large-scale problems to be solved using a distributed computer cluster. The performance of the model is evaluated using grid sizes up to $4096 \times 2048 \times 2048$ grid points and up to 1024 compute cores. This is significantly larger than most ultrasound simulations previously presented, both in terms of grid size and the number of wavelengths this accurately represents. Given the global nature of the gradient calculation, the model shows good strong scaling behaviour, with a speed-up of 1.7x whenever the number of cores is doubled. This means large-scale problems can be spread over increasing numbers of compute cores with only a small computational overhead. The overall efficiency of the parallel implementation is on the order of 60%, which corresponds to the ratio between computation and communication. The large communication overhead is due to the global all-to-all transposition that must be performed for every FFT (a second transposition is avoided by performing operations in the spatial frequency domain in transformed space). Finally, the efficacy of the model for studying real-world problems in HIFU is demonstrated using a large-scale simulation of a HIFU sonication of the kidney.

In the context of the large-scale problems outlined in Section 1 and Table 1, the model developed here allows many problems of interest to be solved using a full-wave model for the first time. This is relevant for studying the aberration of HIFU beams in the body when the focal intensities are relatively low. This has many possible applications, for example, in treatment planning, exposimetry, patient selection and equipment design. However, solving even larger problems involving high focal intensities where many 10's or 100's of harmonics may be present (Yuldashev and Khokhlova, 2011) is still currently out of reach. Looking forward, there are two numerical strategies that might allow the model to be extended further. First, the governing equations could be solved on a non-uniform grid (Treeby, 2013). In the current model, the grid spacing is globally constrained by the highest frequency harmonic that exists anywhere in the domain. However, in practice, the high-frequency harmonics are usually restricted to a small region near the ultrasound focus. Using a non-uniform grid would allow the grid points to be clustered around steep regions of the wavefield, and thus significantly reduce the total number of grid points needed for accurate simulations (Treeby, 2013). Second, a domain decomposition approach could be used in which FFTs are computed locally on each partition using local Fourier basis (Israeli et al., 1994; Albin et al., 2012). This would replace the global communication required by the 3D FFT with local communications between processes responsible for neighbouring partitions.

Considering the computer code, the main limitation is related to the 1D decomposition used to partition the domain. Although this approach exhibits relatively good scaling characteristics, it limits the maximum number of cooperating processes to be $P \leq \min(N_y, N_z)$. This limitation is particularly relevant looking towards the exascale era, where supercomputers integrating over 1M cores are predicted to appear before 2020 (Dongarra et al., 2011). Being prepared for this sort of compute facility requires simulation tools that can efficiently employ hundreds of thousands of compute cores. Moreover, while the trend in supercomputing is to integrate more and more compute cores, the total amount of memory is growing much more slowly (Dongarra et al., 2011). Effectively, this means the memory available per core will remain constant or even decrease in next generation systems. As an example, VAYU has 11,936 cores with 3 GB/core, while its successor RAIJIN has 57,472 cores with only 2 GB/core. This is relevant because with the current 1D decomposition, the maximum grid size that can be solved is ultimately limited by the memory per core. Both of these drawbacks could be solved by using a 2D partitioning approach where the 2D slabs are further broken into 1D pencils, with every process assigned a subset of pencils rather than a complete 2D slab. This would make higher numbers of compute cores (and thus memory) accessible to the simulation, where $P \leq \min\left(N_y^2, N_z^2\right)$.

Another challenge for the current implementation is the amount of output data generated by the code. When recording the time-varying acoustic pressure and particle velocity in a central region (e.g. near the HIFU focus), a single simulation can easily generate 0.5 TB of output data. Copying, post-processing, visualising and archiving such large amounts of data quickly becomes impractical. New techniques for on-the-fly post-processing are thus needed. The use of localised data sampling also introduces a work imbalance into the simulation code. If the output data is only collected from a small region of the domain, only a small subset of the processes actually store data to the disk, with the rest idle. The effective bandwidth to disk could thus be improved by redistributing the data to idle processes after it is collected, allowing more cores to be used for disk operations. Similarly, if some processes only collect a very small amount of data (e.g. from a single grid point in the local partition), the I/O subsystem can become congested by many small write requests resulting in poor performance. In this case, it would be better to collect the output data within each node before writing to disk. These improvements will be explored as part of a future work.

## Funding

This work was supported in part by the Australian Research Council/Microsoft Linkage Project LP100100588, the Engineering and Physical Sciences Research Council UK, and the SoMoPro II Programme co-financed by the European Union and the South-Moravian Region. Computational resources were provided by the Australian National Computing Infrastructure (NCI) National Facility at the Australian National University (VAYU supercomputer), and iVEC at Murdoch University (EPIC supercomputer).

## References

Aanonsen SI, Barkve T, Tjotta JN and Tjotta S (1984) Distortion and harmonic generation in the nearfield of a finite amplitude sound beam. *J. Acoust. Soc. Am.*, 75(3): 749–768.

Albin N, Bruno OP, Cheung TY and Cleveland RO (2012) Fourier continuation methods for high-fidelity simulation of nonlinear acoustic beams. *J. Acoust. Soc. Am.*, 132(4):2371–87.

Averkiou MA and Cleveland RO (1999) Modeling of an electrohydraulic lithotripter with the KZK equation. *J. Acoust. Soc. Am.*, 106(1):102–112.

Berenger JP (1996) Three-dimensional perfectly matched layer for the absorption of electromagnetic waves. *J. Comput. Phys.*, 127(2):363–379.

Bojarski NN (1982) The k-space formulation of the scattering problem in the time domain. *J. Acoust. Soc. Am.*, 72(2): 570–584.

Bojarski NN (1985) The k-space formulation of the scattering problem in the time domain: An improved single propagator formulation. *J. Acoust. Soc. Am.*, 77(3):826–831.

Chen W and Holm S (2004) Fractional Laplacian time-space models for linear and nonlinear lossy media exhibiting arbitrary frequency power-law dependency. *J. Acoust. Soc. Am.*, 115(4):1424–1430.

Clement GT (2004) Perspectives in clinical uses of high-intensity focused ultrasound. *Ultrasonics*, 42(10):1087–93.

Connor CW and Hynynen K (2002) Bio-acoustic thermal lensing and nonlinear propagation in focused ultrasound surgery using large focal spots: A parametric study. *Phys. Med. Biol.*, 47(11):1911–28.

Cox BT, Kara S, Arridge SR and Beard PC (2007) k-space propagation models for acoustically heterogeneous media: Application to biomedical photoacoustics. *J. Acoust. Soc. Am.*, 121(6):3453–3464.

Curra FP, Mourad PD, Khokhlova VA, Cleveland RO and Crum LA (2000) Numerical simulations of heating patterns and tissue temperature response due to high-intensity focused ultrasound. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 47(4):1077–89.

Daoud MI and Lacefield JC (2009) Distributed three-dimensional simulation of B-mode ultrasound imaging using a first-order k-space method. *Phys. Med. Biol.*, 54(17):5173–5192.

Demi L, van Dongen KWA and Verweij MD (2011) A contrast source method for nonlinear acoustic wave fields in media with spatially inhomogeneous attenuation. *J. Acoust. Soc. Am.*, 129(3):1221–1230.

Dongarra J, Beckman P, Moore T, et al. (2011) The international exascale software project roadmap. *Int. J. High Perform. Comput. Appl.*, 25(1):3–60.

Duck FA (1990) *Physical Properties of Tissue: A Comprehensive Reference Book*. London: Academic Press.

Fornberg B (1987) The pseudospectral method: Comparisons with finite differences for the elastic wave equation. *Geophysics*, 52(4):483–501.

Frigo M and Johnson SG (2005) The design and implementation of FFTW3. *Proc. IEEE*, 93(2):216–231.

Frigo M and Johnson SG (2012) FFTW user manual. Technical Report, Massachusetts Institute of Technology, USA,November.

Haber I, Lee R, Klein HH and Boris JP (1973) Advances in electromagnetic plasma simulation techniques. In *Proc. Sixth Conf. Num. Sim. Plasmas*, pages 46–48.

Hamilton MF and Morfey CL (2008) Model Equations. In Hamilton MF and Blackstock DT (eds) *Nonlinear Acoustics*. New York: Acoustical Society of America, pp.41–63.

Hesthaven JS, Gottlieb S and Gottlieb D (2007) *Spectral Methods for Time-Dependent Problems*. Cambridge: Cambridge University Press.

Huijssen J and Verweij MD (2010) An iterative method for the computation of nonlinear, wide-angle, pulsed acoustic fields of medical diagnostic transducers. *J. Acoust. Soc. Am.*, 127(1):33–44.

Illing RO, Kennedy JE, Wu F, et al. (2005) The safety and feasibility of extracorporeal high-intensity focused ultrasound (HIFU) for the treatment of liver and kidney tumours in a Western population. *Brit. J. Cancer,* 93(8): 890–895.

Israeli M, Vozovoi L and Averbuch A (1994) Domain decomposition methods with local fourier basis for parabolic problems. *Contemp. Math*, 157:223–230.

Jaros J, Treeby BE and Rendell AP. Use of multiple GPUs on shared memory multiprocessors for ultrasound propagation simulations. In: *10th Australasian Symposium on Parallel and Distributed Computing* (eds Chen J and Ranjan R), Melbourne, Australia, 31 January - 3 February 2012, volume 127, pp. 43–52. Melbourne: ACS.

Jing Y, Wang T and Clement GT (2012) A k-space method for moderately nonlinear wave propagation. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 59(8): 1664–1673.

Jolesz FA and Hynynen KH (2008) *MRI-guided focused ultrasound surgery*. New York: Informa Healthcare.

Kennedy JE, ter Haar GR and Cranston D (2003) High intensity focused ultrasound: Surgery of the future? *Brit. J. Radiol.*, 76(909):590–599.

Khokhlova VA, Souchon R, Tavakkoli J, Sapozhnikov OA and Cathignol D (2001) Numerical modeling of finite-amplitude sound beams: Shock formation in the near field of a cw plane piston source. *J. Acoust. Soc. Am.,* 110(1): 95–108.

Khokhlova VA, Bessonova OV, Soneson JE, Canney MS, Bailey MR and Crum LA (2010) Bandwidth limitations in characterization of high intensity focused ultrasound fields in the presence of shocks. In: *9th International Symposium on Therapeutic Ultrasound*, Tokyo, Japan, 9–12 June 2010, pp. 363–366.

Lepock JR (2003) Cellular effects of hyperthermia: Relevance to the minimum dose for thermal damage. *Int. J. Hyperthermia*, 19(3):252–66.

Liu LH, McDannold N and Hynynen K (2005) Focal beam distortion and treatment planning in abdominal focused ultrasound surgery. *Med. Phys.*, 32 (5):1270–1280.

Liu QH (1999) Large-scale simulations of electromagnetic and acoustic measurements using the pseudospectral time-domain (PSTD) algorithm. *IEEE. T. Geosci. Remote*, 37(2):917–926.

Mast TD (2000) Empirical relationships between acoustic parameters in human soft tissues. *Acoustics Research Letters Online*, 1(2):37–42.

Mast TD, Souriau LP, Liu DLD, Tabei M, Nachman AI and Waag RC (2001) A k-space method for large-scale models of wave propagation in tissue. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 48(2):341–354.

Okita K, Ono K, Takagi S and Matsumoto Y (2011) Development of high intensity focused ultrasound simulator for large-scale computing. *Int. J. Numer. Meth. Fluids*, 65:43–66.

Okita K, Narumi R, Azuma T, Takagi S and Matumoto Y (2014) The role of numerical simulation for the development of an advanced HIFU system. *Comput. Mech.*, 54(4): 1023–1033.

OsiriX Imaging Software, DICOM sample image sets. Available from: http://www.osirix-viewer.com/datasets. [19 June 2013].

Paulides MM, Stauffer PR, Neufeld E, et al. (2013) Simulation techniques in hyperthermia treatment planning. *Int. J. Hyperthermia*, 29(4):346–357.

Pekurovsky D (2012) P3DFFT: A framework for parallel computations of Fourier transforms in three dimensions. *SIAM J. Sci. Comput.,* 34(4): C192–C209.

Pinton G, Aubry JF, Fink M and Tanter M (2011) Effects of nonlinear ultrasound propagation on high intensity brain therapy. *Med. Phys.,* 38(3): 1207–1216.

Pinton GF, Dahl J, Rosenzweig S and Trahey GE (2009) A heterogeneous nonlinear attenuating full-wave model of ultrasound. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 56(3):474–488.

Pulkkinen A, Werner B, Martin E and Hynynen K (2014) Numerical simulations of clinical focused ultrasound functional neurosurgery. *Phys. Med. Biol*., 59(7):1679–1700.

Schneider U, Pedroni E and Lomax A (1996) The calibration of CT Hounsfield units for radiotherapy treatment planning. *Phys. Med. Biol*., 41(1):111–124.

Tabei M, Mast TD and Waag RC (2002) A k-space method for coupled first-order acoustic propagation equations. *J. Acoust. Soc. Am*., 111(1):53–63.

ter Haar G (2007) Therapeutic applications of ultrasound. *Prog. Biophys. Mol. Biol*., 93(1–3):111–129.

ter Haar G, Shaw A, Pye S, et al. (2011) Guidance on reporting ultrasound exposure conditions for bio-effects studies. *Ultrasound Med. Biol.,* 37(2): 177–183.

Tillett JC, Daoud MI, Lacefield JC and Waag RC (2009) A k-space method for acoustic propagation using coupled first-order equations in three dimensions. *J. Acoust. Soc. Am*., 126(3):1231–1244.

Treeby BE (2013) Modeling nonlinear wave propagation on nonuniform grids using a mapped k-space pseudospectral method. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 60(10):2208–2013.

Treeby BE and Cox BT (2010a) k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields. *J. Biomed. Opt*., 15(2): 021314.

Treeby BE and Cox BT (2010b) Modeling power law absorption and dispersion for acoustic propagation using the fractional Laplacian. *J. Acoust. Soc. Am*., 127(5):2741–2748.

Treeby BE and Cox BT (2011) A k-space Greens function solution for acoustic initial value problems in homogeneous media with power law absorption. *J. Acoust. Soc. Am*., 129(6): 3652–3660.

Treeby BE, Tumen M and Cox BT (2011) Time domain simulation of harmonic ultrasound images and beam patterns in 3D using the k-space pseudospectral method. In: *Medical Image Computing and Computer-Assisted Intervention, Part I*. Heidelberg: Springer, vol 6891 pp.363–370.

Treeby BE, Jaros J, Rendell AP and Cox BT (2012) Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using a k-space pseudospectral method. *J. Acoust. Soc. Am.,* 131(6): 4324–4336.

Verweij MD and Huijssen J (2009) A filtered convolution method for the computation of acoustic wave fields in very large spatiotemporal domains. *J. Acoust. Soc. Am*., 125(4): 1868–1878.

Westervelt PJ (1963) Parametric acoustic array. *J. Acoust. Soc. Am.,* 35(4): 535–537.

Wojcik G, Mould J, Ayter S and Carcione L (1998) A study of second harmonic generation by focused medical transducer pulses. In: *IEEE International Ultrasonics Symposium*, Sendai, Japan, 5–8 October 1998, pp.1583–1588

Wojcik GL, Mould J, Abboud N, et al. (1995) Nonlinear modeling of therapeutic ultrasound. In: *IEEE International Ultrasonics Symposium*, Seattle, WA, 7–10 November 1995, pp. 1617–1622. IEEE.

Yeung PK, Donzis DA and Sreenivasan KR (2012) Dissipation, enstrophy and pressure statistics in turbulence simulations at high Reynolds numbers. *Journal of Fluid Mechanics*, 700:5–15.

Yuldashev PV and Khokhlova VA (2011) Simulation of three-dimensional nonlinear fields of ultrasound therapeutic arrays. *Acoust. Phys.,* 57(3): 334–343.

Yuldashev PV, Krutyansky LM, Khokhlova VA, et al. (2010) Distortion of the focused finite amplitude ultrasound beam behind the random phase layer. *Acoust. Phys*., 56(4): 467–474.

Zhang L and Wang ZB (2010) High-intensity focused ultrasound tumor ablation: review of ten years of clinical experience. *Front. Med. China,* 4(3): 294–302.

## Author biography

*Jiri Jaros* is currently a Marie Curie Fellow and an assistant professor at the Faculty of Information Technology, Brno University of Technology. He received his MSc and PhD in Computer Science from the Brno University of Technology in 2003 and 2010 respectively. He worked for two years as a postdoctoral researcher at the Australian National University in the Computer Systems group under the supervision of Prof Alistair Rendell, and 6 months as a postdoctoral researcher in the Centre for Computational Science, University College London under Prof Peter Coveney. His research interests include high performance computing, scientific computation, parallel programming, many-core accelerator and GPU architecture and programming. He is an active developer of the k-Wave project responsible for large-scale code development, validation and optimisation.

*Alistair Rendell* is a professor and currently the Director of the Research School of Computer Science at the Australian National University. He received a BSc in chemistry from Durham University, UK in 1983 and a PhD in theoretical chemistry from Sydney University, Australia in 1988. His research interests include computational science, high performance computing, parallel and distributed programming and computer architecture.

*Bradley Treeby* is currently an EPSRC Early Career Fellow and leads the Biomedical Ultrasound Group in the Department of Medical Physics and Biomedical Engineering at University College London, UK. He received a BE degree with first class honors in Mechatronics Engineering in 2003, and a PhD in acoustics in 2007, both from the University of Western Australia, Australia. His research interests include biomedical ultrasound, numerical methods and high performance computing. He has published more than 40 scientific papers, and is the author of an open-source acoustics toolbox for MATLAB called k-Wave.