On-the-Fly Calculation of Time-Averaged Acoustic Intensity in Time-Domain Ultrasound Simulations Using a k-Space Pseudospectral Method

Petr Kleparnik, Pavel Zemcik, *Member, IEEE*, Bradley E. Treeby, *Member, IEEE*, and Jiri Jaros

***Abstract*— This article presents a method to calculate the average acoustic intensity during ultrasound simulation using a new approach that exploits compression of intermediate results. One of the applications of high-intensity focused ultrasound (HIFU) simulations is the calculation of the thermal dose, which indicates the amount of tissue destroyed using a state-of-the-art k-space pseudospectral method. The thermal simulation is preceded by the calculation of the average intensity within the acoustic simulation. Due to the time staggering between the particle velocity and the acoustic pressure used in such simulations, the average intensity calculation is typically executed offline after the acoustic simulation consuming both disk space and time (the data can spread over terabytes). Our new approach calculates the average intensity during the acoustic simulation using the output coefficients of a new compression method which enables resolving the time staggering on-the-fly with huge disk space savings. To reduce RAM requirements, the article also presents a new 40-bit method for encoding compression complex coefficients. Experimental numerical simulations with the proposed method have shown that disk space requirements are up to 99% lower. The simulation speed was not significantly affected by the approach and the compression error did not affect the prediction accuracy of the thermal dose. From the standpoint of supercomputers, the new approach is significantly more economical. Saving computing resources increases the chances of real use of acoustic simulations in practice. The method can be applied to signals of a similar character, e.g., for electromagnetic radio waves.**

Manuscript received 20 July 2022; accepted 9 August 2022. Date of publication 16 August 2022; date of current version 27 September 2022. This work was supported in part by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ under Grant 90140; in part by the Engineering and Physical Sciences Research Council (EPSRC), U.K., under Grant EP/S026371/1; and in part by the Brno University of Technology under Project FIT-S-20-6309. *(Corresponding author: Petr Kleparnik.)*

Petr Kleparnik and Pavel Zemcik are with the Department of Computer Graphics and Multimedia, Faculty of Information Technology, Brno University of Technology, 61200 Brno, Czech Republic (e-mail: ikleparnik@fit.vutbr.cz).

Bradley E. Treeby is with the Department of Medical Physics and Biomedical Engineering, University College London, London WC1E 6BT, U.K.

Jiri Jaros is with the Department of Computer Systems, Faculty of Information Technology, Brno University of Technology, 61200 Brno, Czech Republic.

Digital Object Identifier 10.1109/TUFFC.2022.3199173

***Index Terms*— Average acoustic intensity, compression, high-intensity focused ultrasound, k-Wave toolbox, ultrasound simulation.**

I. Introduction

HIGH-INTENSITY focused ultrasound (HIFU) is one of the modern technologies for cancer treatment. It is an emerging noninvasive therapeutic technique that uses ultrasound waves to destroy tissue, such as tumors inside the human body. A beam of ultrasound energy is sent into the tissue using a focused transducer. The focused region is rapidly heated, resulting in irreversible tissue damage while the surrounding tissue is not affected [1], [2], [3], [4], [5].

The purpose of HIFU simulations is to determine the exact location of the focus for a specific patient case. Within the calculations, an acoustic simulation is first performed, on the basis of which the average acoustic intensity in the steady part is calculated. From this quantity, the volume rate of heat deposition term is further calculated as the input quantity for thermal simulation, the result of which is thermal ablation in the tissue.

One of the issues that makes the whole simulation process computationally demanding when using a staggered-grid pseudospectral time domain (PSTD) method is that the calculation of the average intensity only after the acoustic simulation is completed. The reason is the temporal shift between the acoustic pressure and the particle velocity (the acoustic simulation outputs) can be calculated only from complete time series [2], [6]. Due to this fact, it is necessary to store and load a large amount of data from files at once, which leads to high demands on disk storage space (terabytes) and slows down the overall process.

This article presents a new approach for calculating the time-averaged vector of the acoustic intensity during the simulation. The method uses an on-the-fly compression for time-varying HIFU simulation data [7], [8]. During the simulation, the average intensity is calculated on-the-fly, directly from the pressure and particle velocity, including accounting for the temporal staggering of the particle velocity. Therefore, it is not necessary to save the time-varying simulation data to disk, which yields significant memory savings. The numerical

simulations show that despite the lossy compression algorithm used, the numerical errors are negligible. The rest of this article is organized as follows. Section II presents the current calculation approach and the description of the compression method (state-of-the-art). Section III contains a description of the new calculation method and Section IV discusses the performed numerical experiments and results. Finally, Section V concludes this article.

II. State-of-the-Art

A. Simulation Workflow

One of the most important things for the clinical use of HIFU is the precise placement of the focus and dosage assessment for specific patients. The most accurate parameters could be calculated using complex acoustic and thermal models and simulations [2], [4], [6]. The most accurate model possible is necessary to create due to the heterogeneous medium and nonlinear wave propagation. However, solving this problem is computationally very demanding. Due to the nonlinear propagation of ultrasound in a heterogeneous environment, many higher harmonic frequencies are generated. Because of the large distances covered by ultrasonic waves with respect to the wavelength of the highest harmonic frequency, and to obtain accurate results and applications in medical treatment, it is necessary to perform very large simulations. Currently, the resolution of simulations reaches the size of up to $4096 \times 2048 \times 2048$ grid points in 3-D space. During acoustic simulations, a huge amount of data on the order of hundreds of gigabytes is generated [2].

To calculate the required thermal ablation in the tissue, an acoustic simulation is first performed. When using a time-staggered PSTD method (using, e.g., software, such as k-Wave [6]), the results are the time-varying acoustic pressure and (spatially and temporally staggered) particle velocity (a vector field) which can be used to calculate the time-averaged vector intensity. This can then be used to calculate the volume rate of heat deposition (Q) from the divergence of the time-averaged intensity. Finally, the thermal simulation is executed to calculate the heat deposition. The result of the thermal simulation is the information about the temperature in the target region after heating and cooling, the thermal dose, and the lesion size [3], [4], [9]. The thermal dose is normally specified in cumulative equivalent minutes (CEM) relative to $T = 43$ °C (CEM43).

A key bottleneck in this procedure is the fact that the average intensity must be calculated after the end of the acoustic simulation from the stored time-varying pressure and velocity data. The reason is the time shift of the particle velocity with respect to the acoustic pressure, which results from the time grid staggering in the k-space pseudospectral simulation method [2], [6]. This procedure requires reading the large stored time-varying simulation data from the files, temporally shifting the velocity data by half a time step, e.g., using Fourier interpolation, and calculating the average intensity by multiplying the velocity and pressure, and averaging. For large simulations, this means a huge disk and memory consumption, in the order of terabytes, while the result should be a relatively small 3-D matrix with the average intensity [2], [4], [6], [10].

B. Current Method of Calculation

The typical simulation process that leads to the thermal dose begins with the creation of an acoustic simulation model. The simulation parameters are defined in the input simulation file and cover, e.g., the domain discretization based on the physical domain size and the maximum frequency of interest, the spatially varying material properties, the position and properties of the ultrasound transducer and drive signal, and the desired output data—in our case, the output should be the volume rate of heat deposition (Q) or also the average intensity [4], [6], [11]. The Q term calculation is performed as soon as the simulation reaches a steady state [11].

During the execution of the acoustic simulation, the acoustic pressure and time staggered particle velocity are often stored within the entire 3-D simulation domain. The reason why the whole domain is stored rather than a small area around the focus is the aliasing that arises when calculating the divergence of average intensity and the accuracy of the Q calculations is the critical parameter of usability/precision of the presented methods. To calculate the average intensity, it is necessary to sample a signal with a duration of at least one period (T), which is given by the fundamental frequency of the ultrasound signal.

In each simulation (sampling) step, we have the current acoustic pressure and the time staggered particle velocity. The use of the staggered temporal (and also spatial) grids in the simulation calculations is related to discretization. In the case of discretization, their use brings us additional accuracy and stability [12]. Importantly, Fourier interpolation, which is typically used to accurately recalculate the particle velocity time shift, requires entire time series. Thus, after the end of the simulation phase, the calculation of the average intensity vector $\mathbf{I}_{\text{avg}}$ is performed in the postprocessing phase according to

$$\mathbf{I}_{\text{avg}} = \frac{1}{T}\int_0^T p(t)\mathbf{u}(t)\mathrm{d}t \tag{1}$$

or

$$\mathbf{I}_{\text{avg}} = \frac{1}{N}\sum_{n=0}^{N-1} p(n)\mathbf{u}(n) \tag{2}$$

where

$$\mathbf{u}(n) = \mathbf{u}_{\text{staggered}}(n + 0.5) \tag{3}$$

and n or t is the simulation time step or time, respectively, $p(t)$ is acoustic pressure and $\mathbf{u}(t)$ is the vector acoustic particle velocity, T is the acoustic period of the fundamental frequency of the ultrasound signal. The evaluation of this equation is performed through numerical integration. N is the number of samples of discrete signal taken within the period of T (it is assumed that T can be divided exactly into N sampling periods ($1/f_s$), $\Delta t = T/N$ (so that $N\Delta t = T$), $\mathbf{u}_{\text{staggered}}(n)$ is the time staggered particle velocity output from the simulation, and $\mathbf{u}(n)$ is the velocity shifted by half a step forward in time,

typically using Fourier interpolation. For the average intensity calculation, the pressure and velocity data (vector field for the x-, y-, and z-axes) must be read from the output file so that they are continuous over time [6], [10], [13].

The next step is the calculation of the volume rate of heat deposition term Q according to

$$Q = -\mathrm{div}(\mathbf{I}_{\mathrm{avg}}) \tag{4}$$

where the divergence is calculated as the sum of the gradients for each axis. The Q term is one of the input parameters of the Pennes' bio-heat equation used for the subsequent thermal simulation [4], [10], [13], [14], [15].

The main problem of this calculation procedure is the time shift of the acoustic pressure with respect to the acoustic particle velocity, which requires a large amount of data to be stored during the acoustic simulation. In the approach described in Section II-C, the storage of acoustic simulation data and its time shift is replaced by an on-the-fly approximation that is implemented using a compression method. No additional shift of the velocity values over time is performed after the end of the simulation.

C. Compression Method

A compression method for time varying HIFU simulation data was described in [7] and [8]. The lossy compression method is especially focused on the on-the-fly data compression during simulations and it is intended for distributed computing environments even in situations where every grid point in the 3-D space is processed separately.

The method assumes that the time-varying quantities have a harmonic character with only a low amplitude and phase deviations. An output signal is modeled, such as the decomposition of a 1-D signal (one point in 3-D space), as a sum of half-overlapped exponential bases multiplied by a window function. Each base is defined by its complex coefficients (amplitude and phase) and a harmonic frequency (wavenumber). A shifted window function w is defined as

$$w(t,m,d) = \begin{cases} 0, & (m+2)dT \le t < mdT \\ w_0(t - mdT), & \text{otherwise} \end{cases} \tag{5}$$

or

$$w(n,m,d) = \begin{cases} 0, & (m+2)dN \le n < mdN \\ w_0(n - mdN), & \text{otherwise} \end{cases} \tag{6}$$

where w_0 is a window function (typically Hann or Triangular), N or T is the number of samples within the period or acoustic period, respectively, n or t is the simulation time step or time, resp., m is a window (the basis) index, and d is an integer multiple of overlap size. The length of the window is therefore $2dN$ or $2dT$, resp. We obtain complex exponential sliding-window basis vectors

$$b(t,m,h,d) = w(t,m,d)e^{-jh\omega t} \tag{7}$$

or

$$b(n,m,h,d) = w(n,m,d)e^{-jh\Omega n} \tag{8}$$

where

$$\omega = \frac{2\pi}{T} \quad \text{and} \quad \Omega = \frac{2\pi}{N} \tag{9}$$

with the number of the harmonic frequency (wavenumber) h and the known fundamental angular frequency ω (Ω).

Let M be the total number of periods of the fundamental frequency of the signal (let us assume that MN is the total number of samples taken, also MT is the total duration of the signal). The whole reconstructed signal s can then be expressed as

$$s(n) = \sum_{h=1}^{H} \frac{2}{dN} \sum_{m=0}^{M-1} b(n,m,h,d)\widehat{k}(m,h) \tag{10}$$

where H is the number of harmonics (1 to H), h is a harmonic index, and k are the resulting complex coefficients. The normalization factor $2/dN$ is based on the sum of the window function samples $dN/2$, i.e., the area $dT/2$ in continuous time.

The coefficients k for the harmonic frequency h used to model the output simulation signal x are approximately computed for every frame m (usually with a minimum length of two periods $2N$, which experimentally proved to be the most suitable) as a dot product of the simulation signal sample $x(n)$ and the windowed exponential basis vector b (the vinculum denotes complex conjugate of b)

$$\widehat{k}(m,h) = \sum_{n=0}^{MN-1} \overline{b(n,m,h,d)}x(n). \tag{11}$$

The bases of the individual harmonic components are independent/perpendicular to each other because

$$\sum_{m=0}^{M-1} b(n,m,g,d)b(n,m,h,d) = 0 \quad \text{whenever} \quad g \neq h. \tag{12}$$

The coefficients for other harmonic frequencies can be computed independently and are summed in the reconstruction phase. It is not necessary to have the entire signal x available to calculate one coefficient, because the sliding-window basis vectors b are zero for $(m+2)dN \le n < mdN$.

III. Proposed Approach

A. On-the-Fly Calculation of Intensity

Here, we describe how to calculate the time-averaged intensity vector during the simulation using on-the-fly data compression [7], [8]. This directly uses compression coefficients to calculate the average intensity, which are not stored in files during the simulation.

In case of the average intensity calculation, we are specifically interested in the coefficients of the acoustic pressure and the particle velocity. Let k_p and $\mathbf{k}_{u\_\mathrm{staggered}}$ be the computed compression coefficients for the pressure and the staggered velocity from the previous Section II-C. For simplicity, we consider the coefficients only for the one window base. The shift of the particle velocity in time by half the sampling

period ($\Delta t/2$, 1/2 sample, thus phase shift by $\Omega/2$) is being calculated by exploiting a shift of the phase, therefore

$$\mathbf{k}_u(h) = \mathbf{k}_{u\_\text{staggered}}(h)e^{jh\Omega/2} \tag{13}$$

where $\mathbf{k}_u$ is the particle velocity coefficient that is no longer shifted in time by $(\Delta t/2)$ relative to the pressure.

Equations (14)–(18) show only the derivation and only the last (19) or (20) are important for the calculations. To use the integral for derivation, continuous notation is used. Using complex magnitude and phase angle of the coefficients, the harmonic functions for the pressure p and the velocity $\mathbf{u}$ with the angular frequency w (first harmonics) and time t for the one frame can be expressed as

$$p(t) = |k_p|\sin\big(\omega t + \arg\big(k_p\big)\big) \tag{14}$$

$$\mathbf{u}(t) = |\mathbf{k}_u|\sin(\omega t + \arg(\mathbf{k}_u)). \tag{15}$$

The average intensity can be computed as the integral of product of pressure and particle velocity over time from 0 to T, dividing by T to take the average

$$\mathbf{I}_\text{avg} = \frac{1}{T}\int_0^T |k_p|\sin\big(\omega t + \arg\big(k_p\big)\big) \times |\mathbf{k}_u|\sin(\omega t + \arg(\mathbf{k}_u))\mathrm{d}t \tag{16}$$

$$\mathbf{I}_\text{avg} = |k_p||\mathbf{k}_u|\cos\big(\arg\big(k_p\big) - \arg(\mathbf{k}_u)\big)/2 \tag{17}$$

by modifying the expression using trigonometric functions, we achieve

$$\mathbf{I}_\text{avg} = |k_p||\mathbf{k}_u|\,\mathrm{Re}\big(\cos\big(\arg\big(k_p\big) - \arg(\mathbf{k}_u)\big) + j\sin\big(\arg\big(k_p\big) - \arg(\mathbf{k}_u)\big)\big)/2 \tag{18}$$

$$\mathbf{I}_\text{avg} = \mathrm{Re}\big(k_p\overline{\mathbf{k}_u}\big)/2. \tag{19}$$

The average intensity over multiple frames (M) including all harmonic frequencies H can be calculated using a simple principle of numerical integration with exploitation of non-staggered velocity as

$$\mathbf{I}_\text{avg\_all} = \frac{1}{M}\sum_{m=0}^{M-1}\sum_{h=1}^{H}\mathrm{Re}\big(k_p(m,h)\overline{\mathbf{k}_u(m,h)}\big)/2. \tag{20}$$

To obtain suitable results using the compression method, the half-width of the complex exponential window basis should be an integer multiple of $N = 2\pi/(\omega\Delta t)$ (i.e., the input period T), where ω is the known driving fundamental angular frequency and Δt is known time step. The minimum value of the half-width is equal to one period, and therefore, we need at least $2N$ of signal samples for the complete calculation of one complex coefficient. However, if the signal already contains steady-state amplitudes, we can calculate an equally accurate coefficient from the N samples of the signal by mirroring the envelope (window function). Thus, the window function has a constant value in the processed signal frame. This is illustrated in Fig. 1 (for the first harmonic frequency). The period is 106 time steps. The first coefficient is "mirrored" and calculated as the sum of even (2nd, 4th,…,) and odd (1st, 3rd,…,) coefficient for a signal length of one period. The second and third coefficients are computed from two periods. To reconstruct one point in time of the modeled signal, we need two coefficients whose weights are given by the overlapping envelopes. For special cases, thus for the first and last period, the first and last coefficients are duplicated.

The minimum number of memory cells c (single-precision floating-point numbers, 32 bits) required for computing intermediate results in one-time step for the stable parts of the signal is

$$c = 2H \tag{21}$$

as one complex number is needed per every harmonic frequency. Section III-C further describes the method of encoding a complex coefficient to 40 bits instead of 2×32 bits.

Compared to the original average intensity calculation procedure, the new approach does not need to save the pressure and velocity data to a file during the simulation, but needs more RAM. The calculation of the volume rate of heat deposition term Q is performed in the same way as in the case of nonuse of the compression method (offline).

B. Resource Consumption

The compression method described above is advantageous especially in terms of saving disk space, but also increases memory consumption. Consider the following several simulation scenarios representing clinical HIFU simulations.

Table I shows the basic simulation parameters and the comparison of the minimum file sizes required to calculate the Q term. The columns named N_x, N_y, and N_z are simulation domain sizes. The column named "Period" represents the number of simulation steps per period ($N = 1/(f\Delta t) = T/\Delta t$, where f is the known transducer driving frequency, and Δt is the known time step). The parameters (period, harmonics and number of simulation steps) are calculated as part of creating the input simulation file, using the domain sizes, transducer driving frequency, sound speed, real size in z-axis, and the Courant–Friedrichs–Lewy (CFL) number [6].

The transducer driving frequency f is 1 MHz. Due to the nature of the input simulation data (heterogeneous absorbing material properties) and to obtain reasonable accuracy and stability of the simulations, the CFL number is set to 0.1. The real size in the z-axis z_size is 22 cm. The aperture diameter of the transducer bowl is 12 cm, and the radius of curvature is 14 cm. The reference sound speed c_ref is 1524 ms^{-1}. The number of points per z-axis $\Delta z = z_\text{size}/N_z$ and the number of points per wavelength $\text{PPW} = c_\text{ref}/(f\Delta z)$. The time step $\Delta t = 1/(f\lfloor\text{PPW}/\text{CFL}\rfloor)$ and therefore the period $N = 1/(f\Delta t)$. End time is calculated as $t_\text{end} = 2\ z_\text{size}/c_\text{ref}$ and the number of simulation steps, i.e., the total number of simulation steps from the beginning to the end of the simulation $N_t = \lfloor(t_\text{end}/\Delta t)\rceil$. The number of harmonics supported by the spatial grid is given by $H = \lceil 1 \times 10^{-6}(c_\text{ref}/(2\Delta z))\rceil$.

The larger the grid size, the more accurate and usable results (more harmonics). As already mentioned in Section II, for planning HIFU therapy, we need a reasonably high number of harmonic frequencies and the reasonably high spatial resolution. A typical scenario using a single supercomputer node is the case 4. Case 9 is approaching the limits of available supercomputers, using multiple nodes, if we do not want to wait a few days for the result of the simulation.

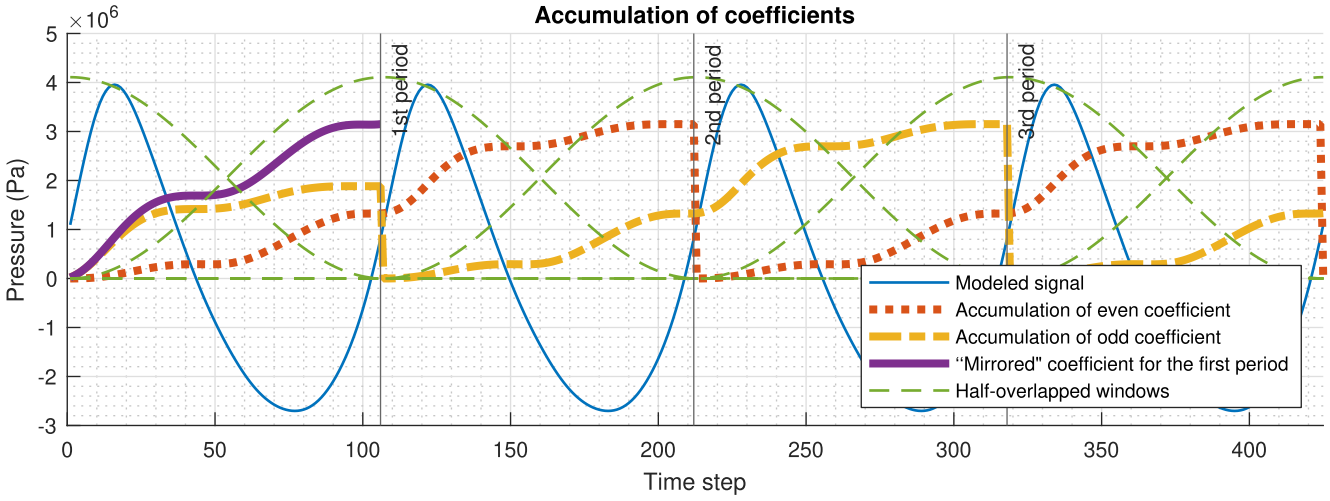


Fig. 1. Accumulation of compression coefficients for the first and more periods.

TABLE I
MINIMUM FILE SIZES FOR Q TERM CALCULATION

	Domain size						File sizes generated during simulation		
							without compression		with compression
Case	N_x	N_y	N_z	Period N	Harmonics H	Simulation steps (N_t)	1 period	3 periods	1 and more periods
1	256	256	350	24	2	6929	8.49 GB	25.3 GB	88 MB
2	384	384	512	35	2	10105	40.6 GB	121 GB	288 MB
3	576	576	768	53	3	15302	207 GB	619 GB	972 MB
4	768	768	1024	70	4	20210	647 GB	1.94 TB	2.30 GB
5	960	960	1280	88	5	25407	1.59 TB	4.76 TB	4.50 GB
6	1152	1152	1536	106	6	30604	3.30 TB	9.90 TB	7.78 GB
7	1344	1344	1792	124	7	35801	6.14 TB	18.4 TB	12.3 GB
8	1536	1536	2048	141	8	40709	10.4 TB	31.2 TB	18.4 GB
9	1728	1728	2304	159	8	45906	16.7 TB	50.1 TB	26.2 GB

In case of the proposed method that uses compression, it is enough to store only one the Q term (a single 3-D matrix). Without the compression, the time series of pressure and velocity data time series is necessary to store (leading into storage of four 4-D matrices). It is noted, please, the fundamental difference in the amount of disk space required.

The RAM memory required for the on-the-fly average intensity calculation is given in Table II. Here, we see that the amount of memory required depends on the number of harmonic frequencies. 40-bit compression refers to the reduction of memory (reduce format) used for complex coefficients from 64 to 40 bits, which is described in Section III-C. The memory calculation is performed according to

$$\text{memory[MB]} = \frac{4N_yN_z(4\lceil N_x m\rceil nH + 3\ N_x)}{1024^2} \tag{22}$$

where N_x, N_y, and N_z are simulation domain sizes, H is the number of harmonic frequencies, n is 1 for one period or 2 for any number of periods larger than 1, and complex size multiplier m is equal to 2 for compression and 1.25 for 40-bit compression. The first number 4 represents the number of bytes per float while the second number 4 represents the number of compressed 3-D matrices, i.e., pressure and velocity for the x-, y-, and z-axes. The number 3 represents uncompressed 3-D matrices for the time-averaged intensity in each Cartesian direction. The operating memory for the original pressure and velocity data is not included in (22) as it is part of the simulation itself (described in the following paragraph).

TABLE II
RAM USED FOR THE ON-THE-FLY AVERAGE INTENSITY CALCULATION

	with compression (2 × 32 bits)		with 40-bit compression (Section III-C)	
Case	1 period	2 and more periods	1 period	2 and more periods
1	1.66 GB	3.06 GB	1.14 GB	2.01 GB
2	5.47 GB	10.1 GB	3.74 GB	6.62 GB
3	26.2 GB	49.6 GB	17.5 GB	32.1 GB
4	80.6 GB	154 GB	53.0 GB	99.1 GB
5	194 GB	374 GB	126 GB	239 GB
6	397 GB	770 GB	257 GB	490 GB
7	729 GB	1.42 TB	469 GB	901 GB
8	1.23 TB	2.41 TB	793 GB	1.53 TB
9	1.76 TB	3.44 TB	1.13 TB	2.18 TB

Table III shows the common memory requirements for the remaining partial operations of the whole acoustic simulation process. They are the same with and without the compression.

TABLE III
OTHER COMMON RAM REQUIREMENTS

Case	RAM estimation used for simulation itself	RAM used for offline Q term calculation
1	3.50 GB	263 MB
2	11.0 GB	864 MB
3	36.5 GB	2.92 GB
4	87.0 GB	6.91 GB
5	170 GB	13.5 GB
6	294 GB	23.3 GB
7	467 GB	37.0 GB
8	697 GB	55.3 GB
9	992 GB	78.7 GB

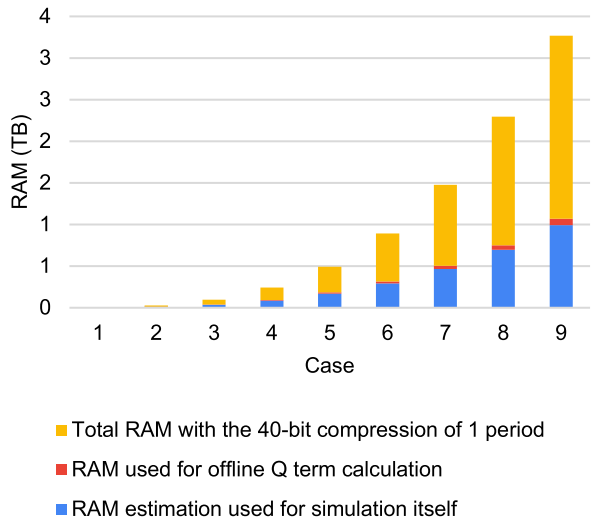


Fig. 2. RAM memory requirements of the proposed method in TB.

The first column is an estimate of the memory requirements of the simulation itself without other operations such as compression or postprocessing, calculated on the basis of simulation experiments. The second column contains the memory requirements for the calculation of the Q term, which is performed as part of postprocessing, and the size corresponds to the three auxiliary matrices that are needed to copy the average intensity matrices due to the sensor mask (the sensor mask is a defined set of locations that will be sampled. In our examples, all points in the domain are sampled, but in general the sensor mask can be an arbitrary and sparse set of locations [6], [16]).

A comparison of the total memory usage with partial operations of the whole acoustic simulation process using the new approach is shown in Fig. 2. The graph shows the case of using 40-bit compression over one period.

It is difficult to evaluate what memory requirements the offline process of calculating the average intensity without the compression has. In the presented case, the amount depends on how much free memory is available on a given computing resource, and more RAM means faster reading and computing. The ideal amount of RAM corresponds to the size of the entire time series of pressure and velocity in the output files. In addition, the size of the auxiliary matrix for the FFT is needed. It is important that due to the time shifts, it is necessary to read at least the whole time series in time, while we can load and process blocks of different sizes. The total offline intensity calculation time does not depend only on the disk speed. In terms of memory layout of the stored 4-D data, it is most advantageous to load as much data as possible at once.

From the point of view of today's clusters, one node contains up to hundreds of GB of RAM. An example is the Barbora supercomputer in Ostrava (IT4Innovations), where each standard computational node is equipped with 192 GB of RAM [17]. This memory therefore limits us to using the compression if it wants to use only one node, e.g., with OpenMP technology. In the case of simulation on multiple nodes using message passing interface (MPI), the operating memory is not such a problem. Conversely, the amount of free disk space and disk write speed can be a bigger complication, such as unavailability of disk space, its high price or disk space quota for the user (e.g., user space quota 10 TB on Barbora scratch filesystem).

If we consider, in the case of no compression, the possibility of storing the entire time series in RAM instead of in the file, in terms of the size of this data it will not be overall advantageous. This would be possible for smaller simulations, but for example already in case 3 we would need at least 246 GB RAM (207 + 36.5 + 2.92, according to Tables I–III), which is not realistic on one node of a common supercomputer. In addition, if we needed to calculate the average intensity from two or more periods. RAM requirements would multiply with each period. In the case of using the compression, the RAM requirements for calculations from two or more periods will be essentially the same.

C. Efficient Coefficient Encoding

To reduce the RAM memory required for temporary complex coefficients kept during accumulation (scalar product or computing intermediate results in a one-time step), we have proposed a method that uses 40 bits instead of 64 bits (2×32 bits) for the float complex number. There are many methods for lossy and lossless compression of float data, the best known of which are fpzip and zfp [18], [19]. These algorithms do not solve our problem because they are designed for single- or double-precision floating-point arrays. Furthermore, procedures for compressing blocks of complex numbers have been published. For example, an exponent is shared across the block of samples and the encoding box is used for the shared exponent to reduce quantization error [20]. Another approach is based on the principle that the number of bits per mantissa is determined by the maximum magnitude sample in the group and the exponent differences are encoded [21].

Our algorithm encodes one complex number independently of neighboring values and uses an approximate range of pressure and particle velocity values. The assumption is that we have at the input a complex number whose exponents of the imaginary and real components do not differ significantly. Thanks to this and the assumed maximum range of the values, only 4 bits are used to encode the larger exponent. The second exponent is stored as the difference in the shifted mantissa.

TABLE IV
40-BIT COMPLEX FLOATING POINT FORMAT

Type of data	real sign	imag. sign	real mantissa	imag. mantissa	shifted exponent
Number of bits	1	1	17	17	4

TABLE V
STANDARD 2 × 32-BIT COMPLEX FLOATING POINT FORMAT (IEEE-754)

Type of data	real sign	real exponent	real mantissa	imag. sign	imag. exponent	imag. mantissa
Number of bits	1	8	23	1	8	23

The format of the 40-bit encoded complex number is shown in Table IV. Mantissa is composed from: 0–16 zero bits, 1 flag bit, and 0–16 data (mantissa or fraction) bits, in total it consists of exactly 17 bits. Number of zero bits means exponent shift from the stored exponent. For comparison, the Table V shows the standard format (IEEE-754) for encoding 2 × 32-bit complex number [22].

The encoding procedure is illustrated by Algorithm 1. The decoding analogous procedure is illustrated by Algorithm 2.

The number of bits for the mantissa can potentially be further reduced, however, 16 + 1 bits, thus a total of 40 bits, is practical in terms of memory alignment to bytes and acceptable errors. Within this article, the relative normalized L-infinity error of the Q term calculation caused by compression up to about 1% is considered acceptable [23].

IV. EXPERIMENTAL EVALUATION

The goal of the experimental numerical simulations was to investigate how the compression method affects the simulation execution time, the computing resources consumption, and numerical accuracy for realistic HIFU simulations.

Within the time measurement of the experimental simulations, the most important time is the time of the simulation phase itself (iteration of simulation steps), which can range from a few minutes to days, depending on the size of the simulation domain. The purpose of this measurement is to show that applying compression does not slow down the simulation process. Furthermore, we are interested in the time of the postprocessing phase, where the calculation of the Q term and offline calculation of the average intensity takes place.

Considering the consumption of computing resources, we are mainly interested in the consumption of RAM and disk space. The aim is to confirm the assumption that despite the higher demands of the compression method on the operating memory, the total memory requirements, including disk space, are significantly smaller.

Finally, the evaluation of compression errors is performed, both for the Q term, the average intensity, and for the outputs of the thermal simulation. The purpose is to show that the number of different points of ablated tissue is ideally the same with and without the use of compression.

The proposed method was implemented within the k-Wave toolbox [6]. Simulations using the k-Wave (k-space pseudospectral methods) were experimentally verified with phantoms and biological tissues [24], [25], [26], [27]. The compression method was implemented in both C++ OpenMP and CUDA versions, but due to the extent of the measured data, this work contains detailed measured results of only the OpenMP version. The original version of the intensity and Q term calculation was implemented only in the MATLAB version. To compare the performance of both approaches, the calculation was ported to C++ to the postprocessing stage. Spatial gradients are computed using Fourier transform. The compression algorithm was implemented in a parallel environment and is performed during the simulation.

Acoustic simulations were performed on one node of the Barbora supercomputer cluster, where 36 processor nodes (2 × Intel Cascade Lake 6240, 2.6 GHz) and at least 192 GB of RAM are available. For reading and writing files, Barbora provides the Luster shared filesystems. On the positive side, it provides a theoretical maximum throughput of 5 GB/s (38 GB/s with burst mode) [17]. Unfortunately, the fact that the filesystem is shared does not guarantee this throughput. Experimental simulations have shown that the times of such calculation phases, in which large files were written or read, sometimes differed significantly (e.g., by a factor of 10).

Due to the available computing resources, four sizes of the simulation domain between 256 × 256 × 350 and 768 × 768 × 1024 were tested, corresponding to cases 1–4 presented in Section III-B. The average intensity was calculated only in the last simulation period. The input simulations material properties such as sound speed, attenuation, density and B/A (nonlinearity parameter) were generated from the AustinWoman Electromagnetic Voxels Model [28]. The heterogeneous parameters were specified for every grid point independently wherever enabled by the simulation tool. Individual book values for the material properties in the human body were used [29].

Table VI shows the measured performance data for each simulation case without the use of compression (N), with the use of compression (C) and with the use of compression using 40-bit coding (C 40-bit). The "file size" represents the size of the file that must be used during the simulation. The RAM memory is divided into two columns. The first is the memory needed for the simulation and sampling itself. The total RAM corresponds to the amount of memory used in the whole simulation case, including compression and postprocessing. In the case no compression is used, the effort is to use the maximum amount of free RAM so that the data for offline calculation of the average intensity within the postprocessing phase is read from the file as quickly as possible, to make the comparison as fair as possible. In the postprocessing phase, differences can be seen between the times when only the Q term calculation is performed and when the average intensity is

Algorithm 1 40-Bit Coefficient Encoding Procedure

1: Get the real and imaginary part of the input float complex number and their sign bits.
2: Get 8-bit exponents and subtract e constant from them which allow the exponent to be stored for only 4 bits.
In case of the acoustic pressure:
$e = 138$, max exponent is 2^{26} $(15 + 138 = 153, 153 - 127 = 26)$
maximal encoding value is $2^{26-16} \times 0 \times \texttt{1FFFF} = 134216704$
minimal encoding value is $2^{26-16-15} \times 0 \times 1 = 0.03125$
In case of the particle velocity:
$e = 114$, max exponent is 2^{2} $(15 + 114 = 129, 129 - 127 = 2)$
maximal encoding value is $2^{2-16} \times 0 \times \texttt{1FFFF} = 7.99993896484375$
minimal encoding value is $2^{2-16-15} \times 0 \times 1 = 0.000000001862645149230957031 25$
3: Get 23-bit mantissas and set their default shift to the right by 6 bits - these least significant bits will be discarded.
4: Find the higher exponent to be saved. Add the difference between the larger and smaller exponents to the shift to the right for the mantissa of a number with a smaller exponent.
5: Crop exponents less than zero and the right shifts greater than 23. Apply right shifts to the mantissas.
6: Round the least significant bits in the mantissas.
7: Set 1 flag bit for the shifted mantissa with a smaller exponent.
8: Check exponent overflow, and set maximum values if necessary.
9: Store the output data at 40 bits (using bitwise operators) as shown in Table IV.

Algorithm 2 40-Bit Coefficient Decoding Procedure

1: Get the mantissas, signs and exponent from the input 40 bits value (using bitwise operators).
2: Shift the mantissas 6 bits to the left (we now have 23-bit mantissas).
3: Add the e constant to the exponents ($e = 138$ for the acoustic pressure, $e = 114$ for the particle velocity).
4: For the both mantissas (`mR`, `mI`) and exponents (`eR`, `eI`):
5: **if** the mantissa is zero **then**
6: set the exponent zero (zero mantissa means zero float number),
7: **else**
8: find the index of the most left one bit in mantissa using the specialized `_BitScanReverse` or `_builtin_clz` function
9: and shift the mantissa according to the index value to the left (`mR <<= 23 - index`)
10: and recompute the final exponent by the index (`eR -= 22 - index`).
11: **end if**
12: Put together the output complex float numbers using bitwise operators at 2 $\times$ 32 bits from signs, mantissas and exponents.

also calculated. Given the overall simulation time, these values are negligible. To determine the variability of the total times, the simulation time was measured for every 5% of the total simulation steps. The coefficient of variation of simulation times $c_v = \sigma/\mu$, where σ is the standard deviation and μ is the mean, was about 9%. Based on the measurement results, we can say that the total simulation times with and without compression for the given domain sizes do not differ significantly (variability is about 3%). Due to the fact that only the last period was sampled for the calculation of the average intensity, which is approximately 0.35% of all simulation steps, the total times are not significantly affected by this sampling. However, we can also see the average iteration times in which the sampling takes place in the table, and we can see that compression is faster than writing to the files. The average nonsampling iteration time of a given simulation case is calculated as the ratio of the sum of individual iteration times to the number of iterations, within the simulation, when sampling was not performed. The average sampling iteration time is calculated as the ratio of the sum of individual iteration times to the number of iterations, within the simulation, when sampling was performed. In particular, the iteration times are 2–10 times faster with the compression than without the compression. In terms of the memory used—the sum of the file size and RAM, the new approach is considerably more economical. A disadvantage of the new approach may be the need for a minimum amount of free RAM depending on the number of coded harmonics.

The numerical error caused by the compression is expressed as a normalized L-infinity error, i.e., maximum absolute difference between noncompression (calculated without the use of the compression) and compression data (calculated using compression) divided by the absolute maximum value of noncompression data. The maximum values were calculated across the entire domain. So for Q term and the average intensity over the whole simulation 3-D space and for the pressure and the velocity in addition also over the sampling simulation time (4-D). Table VII shows the error values in the percent of the volume rate of heat deposition (Q term), the average intensity for the individual axes ($I_{x_{\text{avg}}}$, $I_{y_{\text{avg}}}$, and $I_{z_{\text{avg}}}$), the

TABLE VI
COMPARISON OF MEMORY USAGE AND COMPUTATIONAL TIMES IN INDIVIDUAL CASES OF SIMULATIONS ON ONE NODE OF THE BARBORA SUPERCOMPUTER CLUSTER, WITH 36 PROCESSOR CORES (2 × INTEL CASCADE LAKE 6240, 2.6 GHZ) AND AT LEAST 192 GB OF RAM

Case	Method	File size	Simulation + sampling RAM	Total RAM including post-processing	Post-processing time [seconds]	Simulation time [seconds]	Average non-sampling iteration time [seconds]	Average sampling iteration time [seconds]
1	N	8.54 GB	**3.44** GB	14.1 GB	11.4	906	0.11	0.82
1	C	**88** MB	5.08 GB	5.34 GB	1.05	814	0.11	**0.27**
1	C 40-bit	**88** MB	4.55 GB	**4.82** GB	**0.27**	**767**	0.11	0.34
2	N	40.6 GB	**11.0** GB	61.7 GB	57.4	3,834	0.36	3.63
2	C	**288** MB	16.5 GB	17.3 GB	**1.18**	**3,665**	0.36	**0.81**
2	C 40-bit	**288** MB	14.7 GB	**14.7** GB	2.73	3,816	0.36	1.11
3	N	207 GB	**36.9** GB	168 GB	529	20,456	1.33	45.5
3	C	**972** MB	63.2 GB	66.1 GB	6.64	21,383	1.33	**3.13**
3	C 40-bit	**972** MB	54.4 GB	**57.3** GB	**6.32**	**19,532**	1.33	4.54
4	N	648 GB	**87.1** GB	168 GB	1,880	**74,531**	3.51	53.5
4	C	**2.30** GB	168 GB	175 GB	6.56	78,232	3.51	**9.42**
4	C 40-bit	**2.30** GB	140 GB	**147** GB	**7.22**	74,849	3.51	12.4

TABLE VII
RELATIVE ERRORS CAUSED BY COMPRESSION. THE PARTICLE VELOCITIES (u_x, u_y, u_z) ARE NONSTAGGERED

Case	Q	$I_{x_{avg}}$	$I_{y_{avg}}$	$I_{z_{avg}}$	p	u_x	u_y	u_z
Compression L-infinity error in % (method C in Table VI)								
1	0.00030	0.000045	0.000040	0.000031	0.0068	0.042	0.040	0.012
2	0.0092	0.000066	0.000058	0.000041	0.018	0.083	0.084	0.010
3	0.0098	0.000066	0.000094	0.000071	0.023	0.065	0.061	0.013
4	0.016	0.000090	0.000089	0.000085	0.014	0.048	0.044	0.010
40-bit compression L-infinity error in % (method C 40-bit in Table VI)								
1	0.034	0.0045	0.0031	0.0038	0.0068	0.043	0.040	0.013
2	0.46	0.0046	0.0042	0.0043	0.020	0.083	0.085	0.013
3	0.85	0.0060	0.0049	0.0063	0.024	0.067	0.063	0.015
4	1.21	0.0093	0.0067	0.0068	0.017	0.051	0.046	0.013

TABLE VIII
THERMAL SIMULATION PARAMETERS

The initial temperature	37 °C
Density	1,020 kg m^{-3}
Thermal conductivity	0.5 W m^{-1} K
Specific heat capacity	3,600 J kg^{-1} K
Number of heating time steps	100
Number of cooling time steps	200
Size of the time step (dt)	0.1

acoustic pressure (p), and the nonstaggered particle velocity for the individual axes (u_x, u_y, u_z). If we take into account the accuracy of the float data type (∼7.2 decimal digits), then the intensity errors are very small. Higher error values for the Q term are most likely due to the type of gradient calculation, where many products are performed between FFT and IFFT. In the case of compression, especially with 40-bit coding, the error generally increases with the number of samples in the period.

The CUDA version is fundamentally limited especially by the amount of memory available on the GPU. The amount of memory listed in Table III, in the first column (approximately) should also be available on the GPU and this is quite a major and fundamental limitation. The compression with calculating the average intensity it is not performed on a GPU and uses a CPU and a RAM connected to it. The compression on the GPU does not make sense yet, as in one iteration its computational time is negligible compared to the simulation and in addition it would need the amount of RAM similar to the sizes available to CPUs on the GPUs, which is not yet true.

To be able to meaningfully evaluate the magnitudes of errors caused by the compression, the Q term is applied to the calculation of the thermal simulation. This will show how large the differences will be caused by compression in the heat applied to the tissue, and specifically how the ablated tissue will differ. Thermal simulations were performed in MATLAB using the `kWaveDiffusion` function for the time-domain solution of the diffusion equation or the Pennes' bioheat equation

$$\rho_0 C_0 \frac{\partial T}{\partial t} = K_t \nabla^2 T - \rho_b W_b C_b (T - T_b) + Q \qquad (23)$$

where ρ_0 is the tissue density in kg m^{-3}, C_0 is the tissue specific heat capacity in Jkg^{-1}K^{-1}, T is the total temperature in K, K_t is the tissue thermal conductivity in Wm^{-1}K^{-1}, ρ_b is the blood density in kgm^{-3}, W_b is the blood perfusion rate in s^{-1}, C_b is the blood-specific heat capacity in Jkg^{-1}K^{-1}, T_b is the blood arterial temperature in K, and Q is the volume rate of heat deposition in Wm^{-3}.

The input parameters of the thermal simulation are shown in Table VIII. The heating with the Q term calculated in the

TABLE IX
THERMAL SIMULATION ERRORS

Case	Maximum temperature after heating error [°C]	Maximum temperature after cooling error [°C]	Maximum thermal dose (CEM43) L-infinity error [%]	Absolute maximum thermal dose without the compression [CEM43]	Number of different points of ablated tissue (ablated volume in mm^3 in brackets)
Compression error					
1	0.0001	0.000019	0.0013	2.82×10^7	0 (0)
2	0.00023	0.000019	0.0082	8.06×10^9	0 (0)
3	0.00029	0.000027	0.0089	4.37×10^{12}	0 (0)
4	0.00028	0.000038	0.010	2.57×10^{13}	0 (0)
40-bit compression error					
1	0.0094	0.000088	0.049	2.82×10^7	0 (0)
2	0.016	0.00011	0.62	8.06×10^9	0 (0)
3	0.020	0.00019	0.48	4.37×10^{12}	1 (0.0235)
4	0.020	0.00015	0.42	2.57×10^{13}	0 (0)

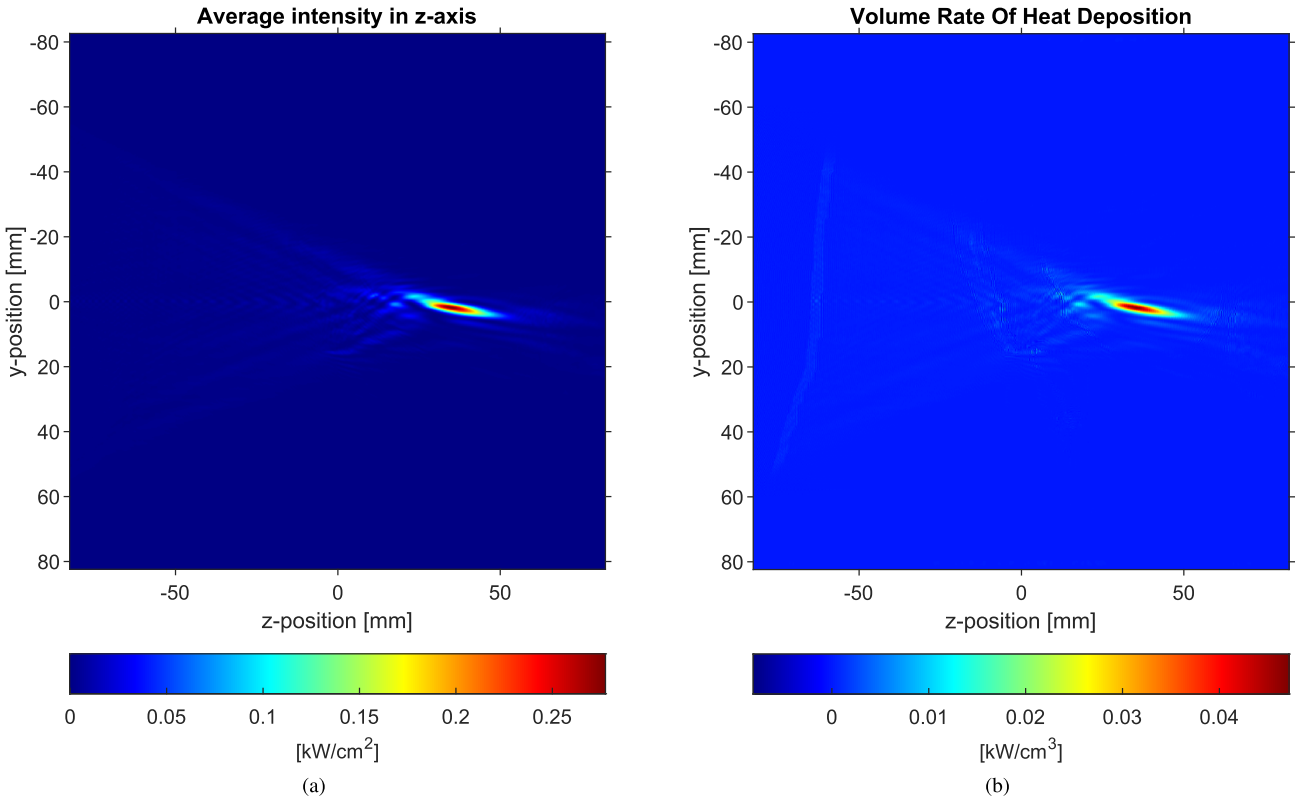


Fig. 3. Visualization of (a) average intensity in z-axis and (b) volume rate of heat deposition without the compression.

acoustic simulation was set to 10 s, the cooling time with the $Q = 0$ was set to 20 s.

The results of the thermal simulation shown in Table IX are the temperature after heating, the temperature after cooling, CEM43 in %, maximum absolute value of CEM43 for cases without compression, and the number of different points (also expressed as ablated volume in mm^3) of binary matrix representing ablated tissue, where CEM43 $\geq$ 240 min.

A very important result of the thermal numerical simulations is the number of ablated tissue points. This value is essentially the same without and with the use of compression. The maximum thermal dose L-infinity errors are around 0.5%, which is negligible. The temperature differences are also minimal.

Figs. 3–7 show sections of the output 3-D data in the center of the x-axis for the case 4. Some of the figures also include a zoomed-in figure cutout from the focused region. Average intensity in z-axis and volume rate of heat deposition is shown in Fig. 3, errors caused by the compression in Figs. 4 and 5. The thermal dose in CEM43 units is shown in Fig. 6 on the left and the ablated tissue (CEM43 $\geq$ 240 min) is shown in red on the right, where shades of gray show the mass density derived from the AustinWoman voxel model. The thermal dose errors caused by compression can be seen

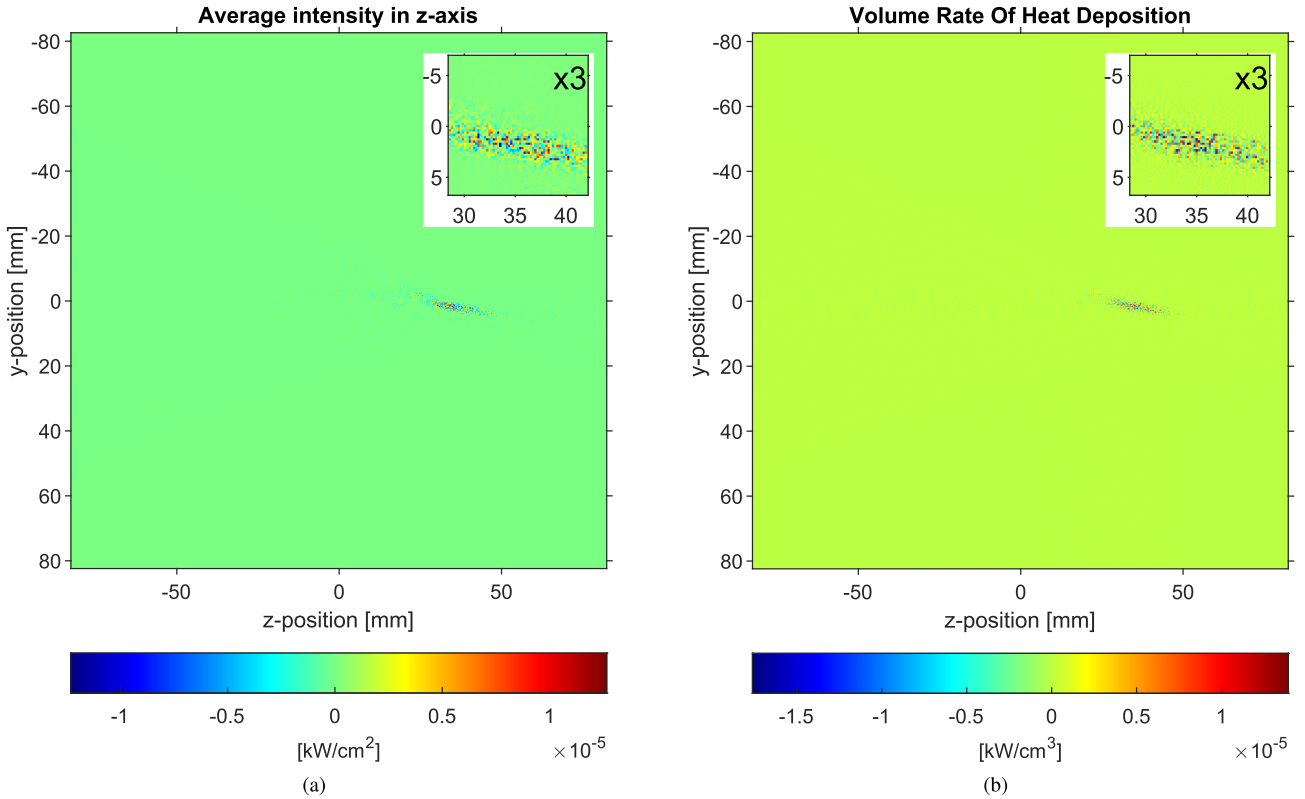


Fig. 4. Visualization of (a) average intensity error in *z*-axis and (b) volume rate of heat deposition with the compression.

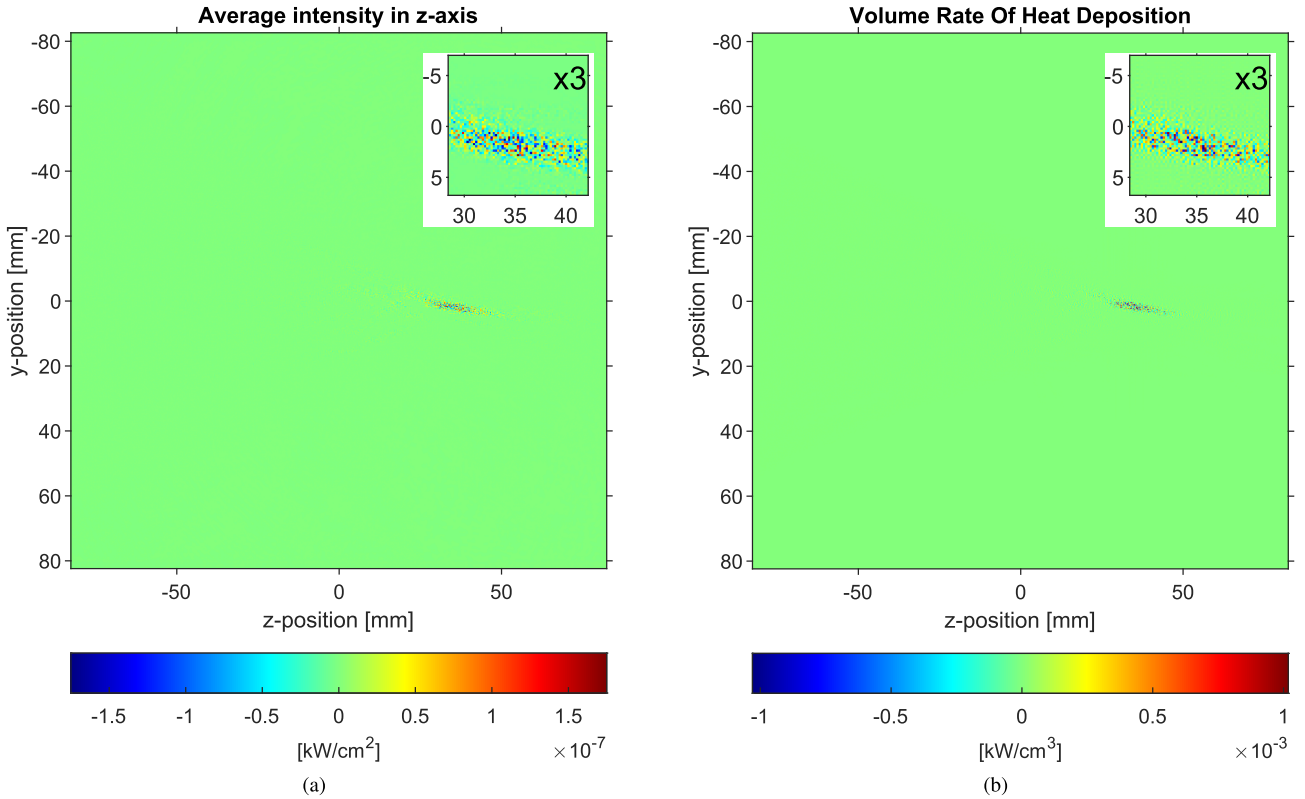
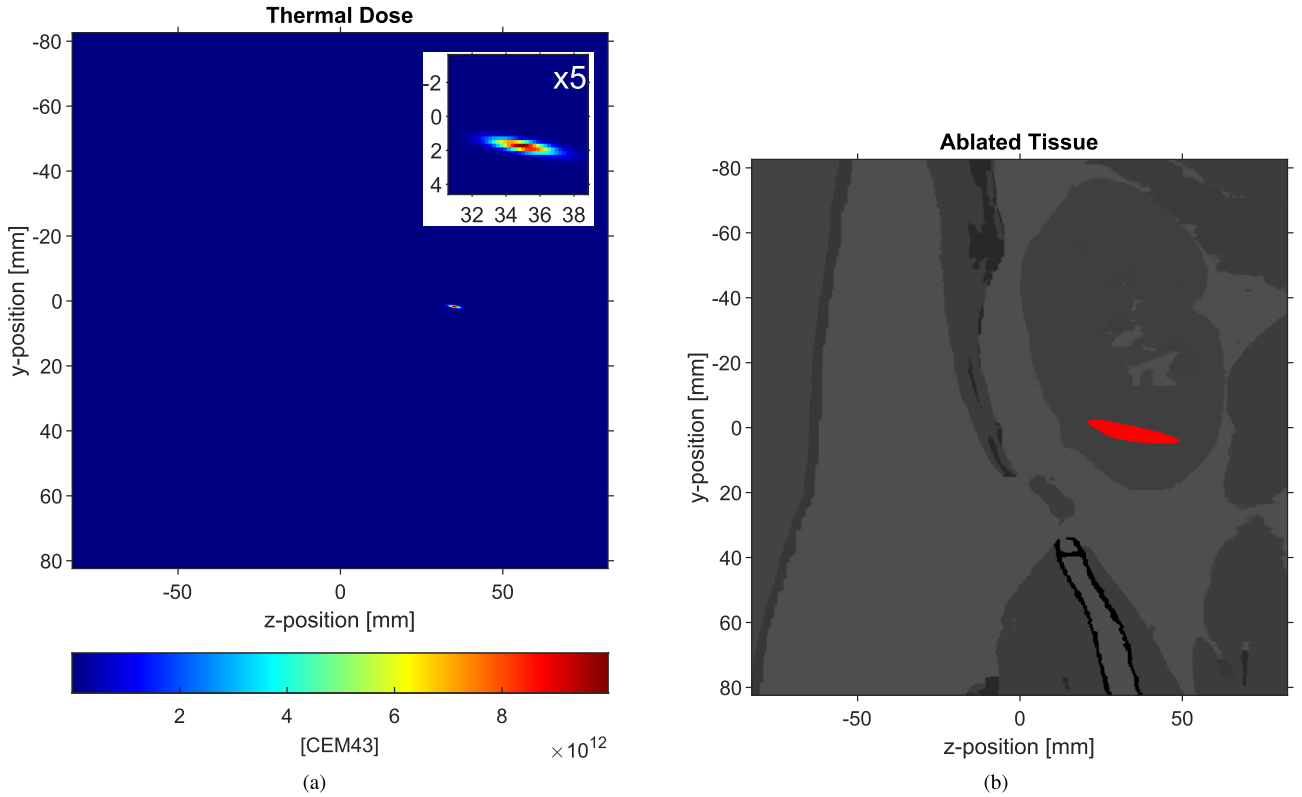


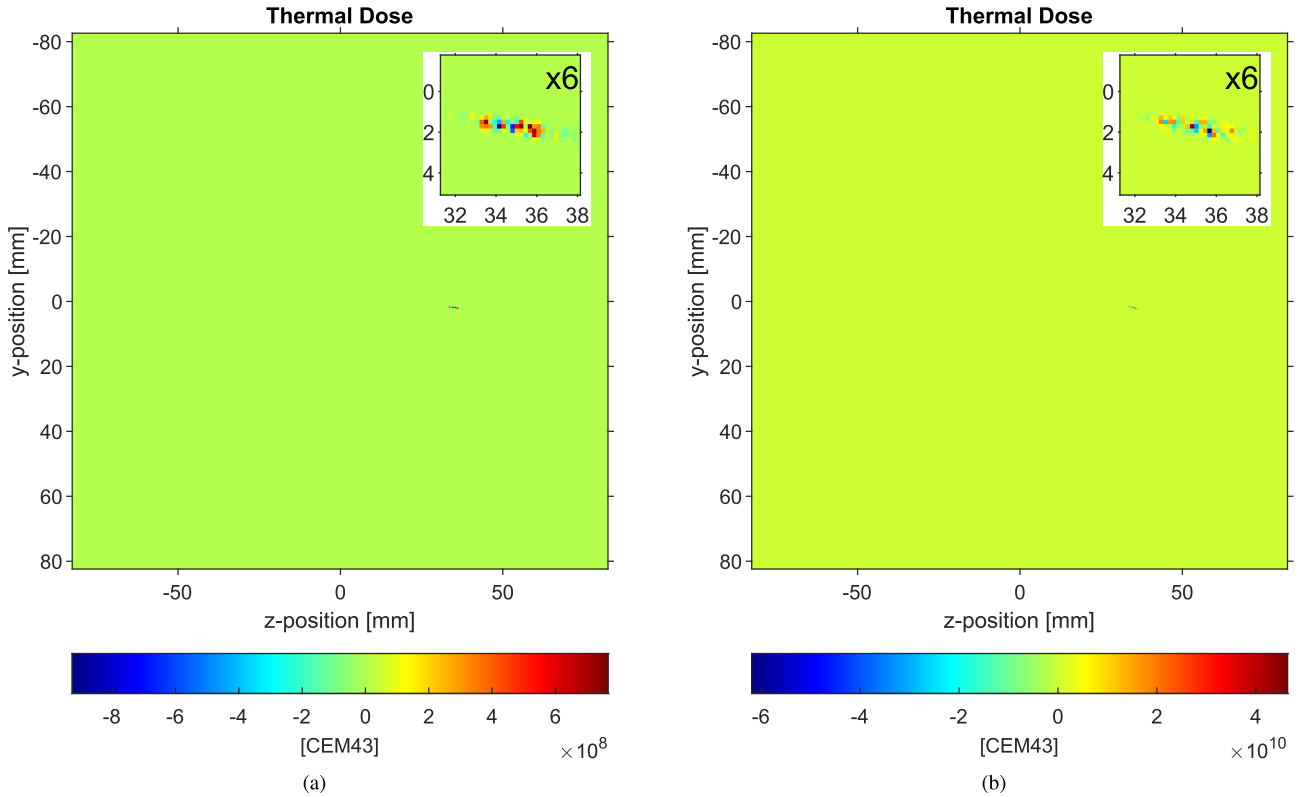
Fig. 5. Visualization of (a) average intensity error in *z*-axis and (b) volume rate of heat deposition with the 40-bit compression.

in Fig. 7, the compression error is on the left, the 40-bit compression error on the right. The absolute thermal dose errors caused by compression shown in Fig. 7 are, in fact, only very small relative errors (0.24% left and 0.036% right) due to the very large maximum value of thermal dose 2.57×10^{13} (see Table IX).



(a) (b)

Fig. 6. Visualization of (a) thermal dose and (b) ablated tissue without use of the compression.



(a) (b)

Fig. 7. Visualization of the thermal dose errors. (a) Compression error (left) and (b) 40-bit compression error (right).

V. Conclusion

This work introduces a new method for calculating the time-averaged acoustic intensity vector during ultrasound simulations performed using a staggered-grid PSTD method. The calculation is performed using a compression method. The method presented in this article has significant advantages over the state-of-the-art represented by the simulation with uncompressed output. The main advantage is largely (up to 99%) reduced consumption of precious disk space during the

simulation which may significantly reduce the price of the computational platform and in some existing configurations of such platforms, it can even present an enabling factor for execution of the simulations. At the same time, the presented method has approximately the same demand for RAM and in longer simulations, it can even reduce the computational time. Moreover, the compression errors in the proposed method are negligible.

The results of the proposed method are based on improvement of the important intermediate step in the acoustic simulations, calculation of the average intensity. The presented approach calculates it using the compression coefficients obtained on-the-fly during the simulation, avoiding saving of the intermediate results of acoustic pressure and particle velocity to the disk during the simulation, as used in state-of-the-art approaches.

In terms of disk space requirements, the new method is significantly more economical. While it has higher RAM memory requirements, it brings significant disk space savings. Please, mind that from the standpoint of supercomputers, the extensive fast I/O disk storage space consumption is much bigger problem than a need for RAM.

Through experimental numerical simulations, it has been shown that the compression does not adversely affect the overall simulation time. Moreover, the average iteration time during sampling is 2–10 times shorter which can reduce the simulation time in some cases.

The accuracy of the new method was evaluated using thermal simulations. Using the new method, we achieved essentially the same results in the determination of ablated tissue as with other approaches. The maximum errors are around 0.5% for thermal dose and 0.02 °C for temperature after heating, which are minimal or even negligible.

The future work may shows that the new method could be applicable for signals of a similar nature, e.g., for electromagnetic radio waves, where the problem of immediate calculation of intensity is the mutual time shift of the signals in time. Future work could focus, e.g., on MPI implementation or saving RAM.

REFERENCES

[1] V. S. Dogra, M. Zhang, and S. Bhatt, "High-intensity focused ultrasound (HIFU) therapy applications," *Ultrasound Clinics*, vol. 4, no. 3, pp. 307–321, Jul. 2009.

[2] J. Jaros, A. P. Rendell, and B. E. Treeby, "Full-wave nonlinear ultrasound simulation on distributed clusters with applications in high-intensity focused ultrasound," *Int. J. High Perform. Comput. Appl.*, vol. 30, no. 2, pp. 137–155, 2016.

[3] E. Cetin, B. Karaboce, H. O. Durmus, O. Kilinc, and O. Orun, "Temperature effect of HIFU with thermal dose estimation," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, May 2020, pp. 1–6.

[4] V. Suomi, J. Jaros, B. Treeby, and R. O. Cleveland, "Full modeling of high-intensity focused ultrasound and thermal heating in the kidney using realistic patient models," *IEEE Trans. Biomed. Eng.*, vol. 65, no. 11, pp. 2660–2670, Nov. 2018.

[5] Y.-F. Zhou, "High intensity focused ultrasound in clinical tumor ablation," *World J. Clin. Oncol.*, vol. 2, no. 1, pp. 8–27, Jan. 2011.

[6] B. E. Treeby and B. T. Cox, "k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields," *J. Biomed. Opt.*, vol. 15, no. 2, 2010, Art. no. 021314.

[7] P. Kleparnik, D. Barina, P. Zemcik, and J. Jaros, "Efficient low-resource compression of HIFU data," *Information*, vol. 9, no. 7, p. 155, Jun. 2018.

[8] P. Kleparnik, P. Zemcik, and J. Jaros, "Efficient lossy compression of ultrasound data," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec. 2017, pp. 232–237.

[9] F. Kuklis, M. Jaros, and J. Jaros, "Accelerated design of HIFU treatment plans using island-based evolutionary strategy," in *Applications of Evolutionary Computation*. Cham, Switzerland: Springer, 2020, pp. 463–478.

[10] A. Grisey, S. Yon, V. Letort, and P. Lafitte, "Simulation of high-intensity focused ultrasound lesions in presence of boiling," *J. Therapeutic Ultrasound*, vol. 4, no. 1, pp. 1–14, Dec. 2016.

[11] B. E. Treeby and T. Saratoon, "The contribution of shear wave absorption to ultrasound heating in bones: Coupled elastic and thermal modeling," in *Proc. IEEE Int. Ultrason. Symp. (IUS)*, Oct. 2015, pp. 1–4.

[12] B. Fornberg, "High-order finite differences and the pseudospectral method on staggered grids," *SIAM J. Numer. Anal.*, vol. 27, no. 4, pp. 904–918, Aug. 1990. [Online]. Available: http://www.jstor.org/stable/2157688

[13] E. G. Williams, "Chapter 2—Plane waves," in *Fourier Acoustics: Sound Radiation and Nearfield Acoustical Holography*. London, U.K.: Academic, 1999, pp. 15–87.

[14] H. H. Pennes, "Analysis of tissue and arterial blood temperatures in the resting human forearm," *J. Appl. Physiol.*, vol. 1, no. 2, pp. 93–122, 1948.

[15] S.-M.-P. Fletcher, M. Choi, N. Ogrodnik, and M. A. O'Reilly, "A porcine model of transvertebral ultrasound and microbubble-mediated blood-spinal cord barrier opening," *Theranostics*, vol. 10, no. 17, pp. 7758–7774, 2020.

[16] J. L. Robertson, B. T. Cox, J. Jaros, and B. E. Treeby, "Accurate simulation of transcranial ultrasound propagation for ultrasonic neuromodulation and stimulation," *J. Acoust. Soc. Amer.*, vol. 141, no. 3, pp. 1726–1738, 2017.

[17] (2021). *IT4Innovations Documentation*. Ostrava, Czech Republic. [Online]. Available: https://docs.it4i.cz/

[18] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE Trans. Vis. Comput. Graphics*, vol. 12, no. 5, pp. 1245–1250, Nov. 2006.

[19] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 12, pp. 2674–2683, Dec. 2014.

[20] Y. F. Choo, B. L. Evans, and A. Gatherer, "Complex block floating-point format with box encoding for wordlength reduction in communication systems," in *Proc. 51st Asilomar Conf. Signals, Syst., Comput.*, Oct. 2017, pp. 1023–1028.

[21] A. Wegener. (2014). *Block Floating Point Compression With Exponent Difference and Mantissa Coding*. [Online]. Available: https://www.freepatentsonline.com/8874794.html

[22] *IEEE Standard for Floating-Point Arithmetic*, IEEE Standard 754-2019, 2019, pp. 1–84.

[23] J.-F. Aubry *et al.*, "Benchmark problems for transcranial ultrasound simulation: Intercomparison of compressional wave models," 2022, *arXiv:2202.04552*.

[24] B. T. Cox, J. G. Laufer, K. P. Kostli, and P. C. Beard, "Experimental validation of photoacoustic k-space propagation models," *Proc. SPIE*, vol. 5320, pp. 238–248, Jul. 2004, doi: 10.1117/12.531178.

[25] K. Wang, E. Teoh, J. Jaros, and B. E. Treeby, "Modelling nonlinear ultrasound propagation in absorbing media using the k-Wave toolbox: Experimental validation," in *Proc. IEEE Int. Ultrason. Symp.*, Oct. 2012, pp. 523–526.

[26] E. Martin, J. Jaros, and B. E. Treeby, "Experimental validation of k-Wave: Nonlinear wave propagation in layered, absorbing fluid media," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 67, no. 1, pp. 81–91, Jan. 2020.

[27] E. Martin and B. E. Treeby, "Experimental validation of computational models for large-scale nonlinear ultrasound simulations in heterogeneous, absorbing fluid media," in *Proc. AIP Conf.*, 2015, vol. 1685, no. 1, Art. no. 070007, doi: 10.1063/1.4934444.

[28] J. W. Massey and A. E. Yilmaz, "AustinMan and AustinWoman: High-fidelity, anatomical voxel models developed from the VHP color images," in *Proc. 38th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2016, pp. 3346–3349.

[29] P. A. Hasgall *et al.*, "IT'IS database for thermal and electromagnetic parameters of biological tissues," Version 3.0, Found. Res. Inf. Technol. Soc. (IT'IS), Zurich, Switzerland, Feb. 2022. [Online]. Available: https://itis.swiss/virtual-population/tissue-properties/overview/ and https://itis.swiss/virtual-population/tissue-properties/downloads/database-v3-0/, doi: 10.13099/VIP21000-04-1.