Computers and Fluids 173 (2018) 195–205

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

High-performance multi-GPU solver for describing nonlinear acoustic waves in homogeneous thermoviscous media

Manuel A. Diaz[a], Maxim A. Solovchuk[a,b,*], Tony W.H. Sheu[b,c]

[a] Institute of Biomedical Engineering and Nanomedicine, National Health Research Institutes 35053, Taiwan
[b] Engineering Science & Ocean Engineering Department, National Taiwan University 10617, Taiwan
[c] Center of Advanced Study in Theoretical Sciences, National Taiwan University 10617, Taiwan

ARTICLE INFO

Article history:
Received 1 October 2017
Revised 6 February 2018
Accepted 1 March 2018
Available online 5 March 2018

Keywords:
Nonlinear acoustics
Finite difference methods
WENO–Z methods
Thermoviscous media
Perfectly matched layers
GPUs

ABSTRACT

A double-precision numerical solver to describe the propagation of high-intensity ultrasound fluctuations using a novel finite-amplitude compressible acoustic model working in multiple processing units (GPUs) is presented. The present solver is based on a conservative hyperbolic formulation derived from a variational analysis of the compressible Navier–Stokes equations and is implemented using an explicit high-order finite difference strategy. In this work, a WENO–Z reconstruction scheme along with a high-order finite-difference stencil are used to approximate the contributions of convective and diffusive spatial operators, respectively. The spatial operators are then associated to a low–storage Runge–Kutta scheme to integrate the system explicitly in time. The present multi-GPU implementation aims to make the best use of every single GPU and gain optimal performance of the algorithm on the per-node basis. To assess the performance of the present solver, a typical mini-server computer with 4 Tesla K80 dual GPU accelerators is used. The results show that the present formulation scales linearly for large domain problems. Moreover, when compared to an OpenMP implementation running with an i7 processor of 4.2 GHz, this is outperformed by our MPI-GPU implementation by a factor of 99. In this work, the present multi-GPU solver is illustrated with a three-dimensional simulation of a highly-intense focused ultrasound propagation.

1. Introduction

The numerical description of the nonlinear behavior of ultrasound propagation plays an important role in the development of biomedical applications of high-intensity focused ultrasound (HIFU) such as thermotherapy and shock wave lithotripsy [1,2, and references therein]. In this applications, small (acoustic) disturbances in the pressure field -typically generated at frequencies greater than 1 kHz- propagate through heat-conducting and viscous (*thermoviscous*) fluid media–such as water or tissue–and transport energy from a piezoelectric device into a target region inside the media. The propagation of this acoustic disturbances is assumed to be *irrotational* and models for describing it can be built from classical fluid dynamics equations, namely:

the equation of mass conservation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0, \tag{1}$$

the equation of motion

$$\rho\left(\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v}\cdot\nabla)\boldsymbol{v}\right) = -\nabla p + \mu\nabla^2\boldsymbol{v} + \left(\mu_B + \frac{1}{3}\mu\right)\nabla(\nabla\cdot\boldsymbol{v}), \tag{2}$$

and *the equation of heat transfer*

$$\rho T\left(\frac{\partial s}{\partial t} + (\boldsymbol{v}\cdot\nabla)s\right) = \frac{\mu}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial v_k}{\partial x_k}\right)^2 + \mu_B(\nabla\cdot\boldsymbol{v})^2 + \kappa\nabla^2 T, \tag{3}$$

where $\boldsymbol{v} : \boldsymbol{v}(\boldsymbol{x},t) = (u(\boldsymbol{x},t), v(\boldsymbol{x},t), w(\boldsymbol{x},t))$ is the fluid particle velocity, ρ: $\rho(\boldsymbol{x}, t)$ is the density of the media, s: $s(\boldsymbol{x}, t)$, the entropy, p and T are the pressure and temperature, respectively. $\boldsymbol{x} = (x,y,z) \in \mathbf{R}^3$ are the spatial (Cartesian) coordinates and $t > 0 \in \mathbf{R}$ denotes time. $\mu \geq 0$ and $\mu_B \geq 0$ are the coefficients of shear and bulk viscosity, and $\kappa \geq 0$ is the heat conductivity coefficient. The above system of equation is considered to be closed upon defining

[*] Corresponding author at: Institute of Biomedical Engineering and Nanomedicine, No. 35, Keyan Road, Zhunan Town 35053, Taiwan.

E-mail addresses: manuel.ade@nhri.org.tw (M.A. Diaz), solovchuk@nhri.org.tw (M.A. Solovchuk), twhsheu@ntu.edu.tw (T.W.H. Sheu).

https://doi.org/10.1016/j.compfluid.2018.03.008
0045-7930/© 2018 Elsevier Ltd. All rights reserved.

a suitable *equation of state* (EoS) and the *temperature relation* of the form

$$p = p(\rho, s) \qquad \text{and} \qquad T = T(\rho, s), \tag{4}$$

respectively. Let the variables $\rho_a = \rho - \rho_0$, $p_a = p - p_0$, $s_a = s - s_0$, $T_a = T - T_0$ and $\boldsymbol{v}_a = \boldsymbol{v}$ denote small acoustic disturbances relative to the state of the media, which is here assumed to be uniform and at rest ($\boldsymbol{v}_0 \equiv 0$). Introducing these acoustic variables into Eqs. (1)–(3) results in a set of *fluctuation equations*. Normalizing these resulting equations reveals that (ρ_a/ρ_0, p_a/p_0) are order ε and (T_a/T_0, s_a/s_0) are order $\varepsilon\delta$, where

$$\begin{cases} \varepsilon = \frac{v_a}{c_0} \ll 1: & \text{is a measurement of the acoustic amplitude,} \\ \delta = \frac{\mu}{\rho_0 c_0 l_c} \ll 1: & \text{is a measurement of the viscous dissipation.} \end{cases} \tag{5}$$

Therefore, each term in the *fluctuation equations* can be identified to be either order ε, ε^2, $\varepsilon\delta$ or $\varepsilon^2\delta$. In Eq. (5), c_0 denotes the speed of sound of the media and l_c is the characteristic length defined as $l_c = c_0/t_c$.

Hamilton and Blackstock in [1] argue that consistent acoustic equations can be obtained by only retaining terms of order ε, ε^2 and $\varepsilon\delta$ from the *fluctuation equations*. So that the fluctuation equations of mass and momentum simply become

$$\frac{\partial \rho_a}{\partial t} + \rho_0 \nabla \cdot \boldsymbol{v}_a = \frac{1}{\rho_0 c_0^4}\frac{\partial (p_a)^2}{\partial t} + \frac{1}{c_0^2}\frac{\partial \mathscr{L}}{\partial t}, \tag{6}$$

$$\rho_0 \frac{\partial \boldsymbol{v}_a}{\partial t} + \nabla p_a = (\mu_B + \mu)\nabla^2 \boldsymbol{v}_a - \nabla \mathscr{L}, \tag{7}$$

where

$$\mathscr{L} = \frac{1}{2}\rho_0 u_a^2 - \frac{p_a}{2\rho_0 c_0^2} \tag{8}$$

is the second-order Lagrangian density. The Lagrangian terms in above equations are found by using *the method of successive approximations* [1, Chap. 7.3]. Lastly, by using Taylor expansions up to the order $\varepsilon\delta$ for the EoS and the temperature relation, and using the fluctuation equation of heat-transfer together with thermodynamic relations, the classical quadratic EoS for thermoviscous fluids can be obtained [1], namely

$$p - p_0 = c_0^2 \rho_a + \frac{c_0^2}{\rho_0}\frac{B/A}{2}\rho_a^2 + \kappa\left(\frac{1}{c_v} - \frac{1}{c_p}\right)\nabla \cdot \boldsymbol{v}_a, \tag{9}$$

where B/A is Beyer's parameter, c_v and c_p are the heat capacities at constant volume and constant pressure, respectively.

In [3], it is argued that in most practical applications of acoustics –such as HIFU ultrasound propagation–only *cumulative nonlinear effects* are required. These effects are mainly accounted by the quadratic EoS and the first term in left hand side of Eq. (6). On the other hand the Lagrangian terms only account for *local nonlinear effects* and therefore their contribution can be safely neglected by setting $\mathscr{L} = 0$ [3]. Moreover, upon neglecting the contribution of these terms, it is shown that the classical acoustic wave models of Westervelt and the KZK equation can be recovered [1].

In [4], using successive approximation principles and the assumption $\mathscr{L} = 0$, it is found that the first term on the left hand side of Eq. (6) can be treated as $\frac{1}{\rho_0 c_0^4}\frac{\partial (p_a)^2}{\partial t} = -\nabla \cdot (p_a \boldsymbol{v}_a) + O(\varepsilon^3)$. Using this approximation, Eqs. (5), (6) and (8) can be written as the convective–diffusive system of the form

$$\frac{\partial p_a}{\partial t} + \nabla \cdot \left((c_0^2 \rho_0 + \beta p_a)\boldsymbol{v}_a\right) = d_1 \nabla^2 p_a, \tag{10a}$$

$$\frac{\partial \boldsymbol{v}_a}{\partial t} + \frac{1}{\rho_0}\nabla (p_a) = d_2 \nabla^2 \boldsymbol{v}_a, \tag{10b}$$

Table 1
Acoustic parameters of the media: density, ρ_0, speed of sound, c_0, Beyer's parameter of nonlinearity, B/A, and the acoustic absorption coefficient, α, for standard air, distilled water and human liver tissue.

Media	ρ_0 [kg/m^3]	c_0 [m/s]	B/A [–]	α [Np/m at 1 Mhz]
Air	1.205[a]	343[a]	0.4[b,d]	18.9[f]
Water	998.3[a]	1448[a]	5.0[b]	0.025[e]
Liver tissue	1055[c]	1550[c]	6.5[b]	5.0[g]

[a] Properties at 20 °C obtained from Ref. [10].
[b] Parameter A/B obtained from Ref. [11].
[c] Human liver at 30 °C obtained from Ref. [2].
[d] For γ–law gases : $B/A = \gamma - 1$, where γ=1.4 for air.[a]
[e] Ultrasound attenuation coefs. obtained from Ref. [12].
[f] For air at 20 °C, 1 atm with 60% of relative humidity. Ref. [13].
[g] Obtained from Ref. [14].

Table 2
Thermoviscous parameters of the media: shear viscosity, μ, bulk viscosity, μ_B, heat conductivity, κ, specific heat capacities at constant volume and pressure, $c_v\,|\,c_p$, for standard air, distilled water and human liver tissue.

Media	μ [Pa · s]	μ_B [Pa · s]	κ [W/(m · K)]	$c_v\,|\,c_p$ [kJ/(kg · K)]
Air	1.80E–5[a]		0.0257[a]	1.01 | 0.718[a]
Water	1.01E–3[b]	2.92E–3[b]	0.598[b]	4.182[b,e]
Liver tissue			0.469[d]	3.540[c,e]

[a] Properties at 20 °C obtained from Ref. [10].
[b] Properties at 20 °C obtained from Ref. [12].
[c] Human liver at 30 °C obtained from IT'IS Foundation: www.itis.ethz.ch.
[d] Human liver at 30 °C obtained from Ref. [2].
[e] For liquid fluids $c_v \simeq c_p$.

where p_a is the acoustic pressure field, $\boldsymbol{v}_a$ is the velocity field and $\beta = 1 + \frac{B/A}{2}$ is the parameter of nonlinearity of the media, $d_1 = \frac{\kappa}{\rho_0}\left(\frac{1}{c_v} - \frac{1}{c_p}\right)$ and $d_2 = \frac{1}{\rho_0}\left(\mu_B + \frac{4}{3}\mu\right)$ are the heat conductive and the viscous coefficients, respectively. Both coefficients are defined through thermoviscous properties of the media (i.e. μ, μ_B, κ, c_v and c_p). In Tables 1 and 2, some of the well-established acoustic and thermoviscous parameters for standard air, distilled water and human liver tissue are summarized.

In [4], the system of Eqs. (10) has been numerically explored using traditional finite-difference WENO methods [5] associated with the third-order low-storage Runge–Kutta scheme [6] and perfect matching layers (PML) [7] for one- and two-dimensional initial value problems. There, it is found that this system is suitable for the direct description of the nonlinear propagation of ultrasound for applications where *local nonlinear effects* are negligible.

In this work, we extend the work of Diaz et al. [4] by formulating a high-performing and parallel numerical solver using CUDA C [8] and MPI to solve the Eqs. (10) in three-dimensions. As a step forward from preceding works, our solver is based on a newer formulation of the finite-difference weighted essentially non-oscillatory (WENO) methods, namely, the WENO–Z formulation introduced in [9] which is shown to outperform the original formulation of Jiang and Shu [5] in the vicinity of discontinuous extrema.

Under this framework, a highly-performing direct acoustic pressure flow solver is obtained and can be used to study weakly- and highly-nonlinear acoustic wave propagation phenomena in homogeneous thermoviscous media. This solver is implemented to work on single and on multiple K80 Nvidia Tesla GPU accelerators. When the performance of the solver is compared with that of an OpenMP implementation, working on i7 CPU with 8 threads, it shows 99 × factor of improvement or 486 × factor with respect to the single-threaded solution. The formulation of the present solver is non-trivial. The details of its design and implementation are reported in the following sections.

This work is organized as follows. The discretization of the thermoviscous acoustic model in Eqs. (10) is briefly presented in Section 2. In Section 3, the basic algorithm, as well as the allocation, communication and kernel optimization strategies that led to the present numerical implementation are described. The performance of the proposed solver is then reported in Section 4. To illustrate the capabilities of the present solver formulation, the acoustic propagations produced by a typical HIFU device are simulated using a three-dimensional model in Section 5. Lastly, concluding remarks are addressed in Section 6.

2. Discretization

Under the assumption of homogeneous media, the density, ρ_0, and speed of sound, c_0, are constant parameters. Therefore, in three-dimensions, the system in Eqs. (10) can be recognized as a convective–diffusive system (or balance law) of the form

$$\boldsymbol{q}_t + \boldsymbol{f}_x + \boldsymbol{g}_y + \boldsymbol{h}_z = \mathcal{D}\,\nabla^2 \boldsymbol{q}, \tag{11}$$

where $\boldsymbol{q} = \{q_1, q_2, q_3, q_4\}^T = \{u_a, v_a, w_a, p_a\}^T$, is the vector of conserved acoustic quantities, $\mathcal{D} = diag\{d_1, d_1, d_1, d_2\}$ is square diagonal matrix that contains diffusive coefficients of the medium, and $\boldsymbol{f}$, $\boldsymbol{g}$ and $\boldsymbol{h}$ are flux functions defined as

$$\boldsymbol{f} = \begin{bmatrix} k_a u_a \\ p_a/\rho_0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{g} = \begin{bmatrix} k_a v_a \\ 0 \\ p_a/\rho_0 \\ 0 \end{bmatrix}, \quad \boldsymbol{h} = \begin{bmatrix} k_a w_a \\ 0 \\ 0 \\ p_a/\rho_0 \end{bmatrix}, \tag{12}$$

where $k_a = c_0^2 \rho_0 + \beta p_a$. We are interested in weak solutions of this system. To maintain the correct physical behavior in discontinuous flow regions, a stable approximation is obtained by using a shock capturing scheme. The discretization is based on a semi-discrete approach for a structured three-dimensional (3-d) grid. For the present work, we use a Cartesian grid with a total of $N_x \times N_y \times N_z$ uniformly distributed cells. A conservative finite-difference method is used to discretize the convective terms and a traditional finite-difference stencil operator is selected for the diffusive part. By approximating all terms by explicit spatial operators it is evident that the system can be written as a continuous system of differential equations as

$$\frac{\partial \boldsymbol{q}(t)}{\partial t} = \mathcal{L}(\boldsymbol{q}(t)), \tag{13}$$

where the vector $\boldsymbol{q}(t) \in \mathbb{R}^N$ is the vector of the acoustic conserved quantities in the computational domain. The system of differential Eqs. (13) can be integrated by using an explicit third-order low-storage Runge–Kutta scheme [15], namely

$$\begin{aligned} \boldsymbol{q}^{(1)} &= \boldsymbol{q}^n + \Delta t\,\mathcal{L}(\boldsymbol{q}^n), \\ \boldsymbol{q}^{(2)} &= \frac{3}{4}\boldsymbol{q}^n + \frac{1}{4}\boldsymbol{q}^{(1)} + \frac{1}{4}\Delta t\,\mathcal{L}(\boldsymbol{q}^{(1)}), \\ \boldsymbol{q}^{n+1} &= \frac{1}{3}\boldsymbol{q}^n + \frac{2}{3}\boldsymbol{q}^{(2)} + \frac{2}{3}\Delta t\,\mathcal{L}(\boldsymbol{q}^{(2)}). \end{aligned} \tag{14}$$

The numerical spatial operator $\mathcal{L}(\cdot) : \mathbb{R}^N \to \mathbb{R}^N$ returns the right-hand side (RHS) of Eq. (13). It provides an approximation to the convective and diffusive terms in (11) and is computationally the most intensive part when integrating the system (13). Furthermore, three evaluations are required for every time step due to the three stages of the numerical integrator. Given the Cartesian mesh, we apply a dimensional splitting as

$$\mathcal{L} = \mathcal{L}^{\boldsymbol{f}} + \mathcal{L}^{\boldsymbol{g}} + \mathcal{L}^{\boldsymbol{h}} + \mathcal{L}^{\nabla^2} \tag{15}$$

for the individual contributions of the operators. The contribution of convective operators for cell *i*, *j*, *k* reads

$$\begin{aligned} \mathcal{L}^{\boldsymbol{f}}_{i,j,k} &= -\frac{1}{\Delta x}\left(\hat{\boldsymbol{f}}^n_{i+1/2,j,k} - \hat{\boldsymbol{f}}^n_{i-1/2,j,k}\right), \\ \mathcal{L}^{\boldsymbol{g}}_{i,j,k} &= -\frac{1}{\Delta y}\left(\hat{\boldsymbol{g}}^n_{i,j+1/2,k} - \hat{\boldsymbol{g}}^n_{i,j-1/2,k}\right), \\ \mathcal{L}^{\boldsymbol{h}}_{i,j,k} &= -\frac{1}{\Delta z}\left(\hat{\boldsymbol{h}}^n_{i,j,k+1/2} - \hat{\boldsymbol{h}}^n_{i,j,k-1/2}\right). \end{aligned} \tag{16}$$

For the $\mathcal{L}^{\boldsymbol{f}}$, Δx denotes a uniform spacing along the x-direction and $\hat{\boldsymbol{f}}^n_{i+1/2}$ is a numerical approximation to the flux $\boldsymbol{f}$ at the cell interface $i + 1/2$ for the time interval $[t^n, t^{n+1})$. Here, the discrete time is given as $t^n = n\Delta t$ with fixed time step Δt. The numerical flux at the cell face $i + 1/2$ is obtained as

$$\hat{\boldsymbol{f}}_{i+1/2} = \hat{\boldsymbol{f}}^+_{i+1/2} + \hat{\boldsymbol{f}}^-_{i+1/2} \tag{17}$$

where $\hat{\boldsymbol{f}}^\pm_{i+1/2}$ are 5th-order ($r = 3$) WENO–Z reconstructions [9] of the left-going ($\boldsymbol{f}^-$) and right-going ($\boldsymbol{f}^+$) parts of the flux $\boldsymbol{f}$. In this work, the left- and right-going parts of the flux are obtained by means of the Lax–Friedrichs flux splitting, namely

$$\boldsymbol{f}^\pm(\boldsymbol{q}) = \frac{1}{2}(\boldsymbol{f}(\boldsymbol{q}) \pm c\boldsymbol{q}), \text{ with } c = c_0 + \frac{\beta}{\rho_0 c_0}\max_{i,j,k}|p_a|. \tag{18}$$

The definitions for $\mathcal{L}^{\boldsymbol{g}}$ and $\mathcal{L}^{\boldsymbol{h}}$ for the cell *i*, *j*, *k* are analogous to $\mathcal{L}^{\boldsymbol{f}}$. Notice that, unlike the Navier–Stokes system of equations, the nonlinear acoustic model in (10) allows us to directly evolve the primitive variables. Therefore, the flux splitting is performed on the vector of conserved variables, $\boldsymbol{q}$, without any special transformation. Lastly, the contribution of $\mathcal{L}^{\nabla^2}$ for cell *i*, *j*, *k* is given by a high-order 13-point finite-difference stencil that reads

$$\begin{aligned} \mathcal{L}^{\nabla^2}_{i,j,k} = \; & \frac{\mathcal{D}}{12\,\Delta x^2}\left(-\boldsymbol{q}^n_{i-2,j,k} + 16\boldsymbol{q}^n_{i-1,j,k} - 30\boldsymbol{q}^n_{i,j,k}\right. \\ & \left. + 16\boldsymbol{q}^n_{i+1,j,k} - \boldsymbol{q}^n_{i+2,j,k}\right) \\ & + \frac{\mathcal{D}}{12\,\Delta y^2}\left(-\boldsymbol{q}^n_{i,j-2,k} + 16\boldsymbol{q}^n_{i,j-1,k} - 30\boldsymbol{q}^n_{i,j,k}\right. \\ & \left. + 16\boldsymbol{q}^n_{i,j+1,k} - \boldsymbol{q}^n_{i,j+2,k}\right) \\ & + \frac{\mathcal{D}}{12\,\Delta z^2}\left(-\boldsymbol{q}^n_{i,j,k-2} + 16\boldsymbol{q}^n_{i,j,k-1} - 30\boldsymbol{q}^n_{i,j,k}\right. \\ & \left. + 16\boldsymbol{q}^n_{i,j,k+1} - \boldsymbol{q}^n_{i,j,k+2}\right), \end{aligned} \tag{19}$$

where $\mathcal{D}$ is a diagonal matrix with constant diffusive coefficients. Let us remark that the contribution of the flux and diffusive approximations depend on the cell average information of $\boldsymbol{q}^n$: $\boldsymbol{q}(t^n)$ in the complete domain. Therefore, to ensure a stable evolution of the information, we request that the size of the timestep be set as

$$\Delta t = \min(\Delta t^{\mathcal{C}}, \Delta t^{\mathcal{D}}), \tag{20}$$

where $\Delta t^{\mathcal{C}}$ and $\Delta t^{\mathcal{D}}$ represent the stable timestep size for the convection and the diffusion process respectively, defined as

$$\Delta t^{\mathcal{C}} \leq \frac{\text{CFL}}{c}\min(\Delta x, \Delta y, \Delta z), \tag{21}$$

$$\Delta t^{\mathcal{D}} \leq \frac{\text{CD}}{\max(d_1, d_2)\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)}. \tag{22}$$

where CFL < 1 is the Courant number and CD $< 1/2$ is the stability factor for the diffusion.

2.1. Brief on WENO formulations

Since the successful formulation of weighted essentially non-oscillatory (WENO) methods by Jiang and Shu [5], here denoted as WENO-JS, modifications on the nonlinear weights have been pursued to improve the accuracy of the method and/or reduce

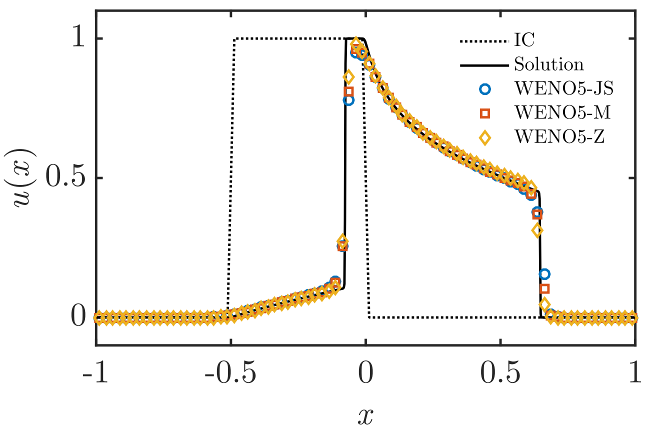

Fig. 1. Solutions for the nonlinear Buckley–Levertt problem using the fifth-order formulation of WENO–SJ [5], WENO–M [16] and WENO–Z [9] on an uniform grid of 80 nodes. The comparison of the numerical results shows that the solution of WENO–Z is the closest to the reference solution of this problem.

its computational cost. Arguably, the most popular reformulations in the literature are: the WENO formulation of Henrick et al. [16] called mapped-WENO or simply WENO–M; and the reformulation of Borges et al. [9], denominated WENO–Z. These new formulations are claimed to exhibit improved accuracy in the vicinity of discontinuous extrema.

To verify this claim, one can simply implement the fifth-order scheme of each of the above mentioned WENO formulations and proceed to compare numerically their accuracy. Here we have chosen to verify this claim by using the classical *Buckley–Leverett* nonlinear hyperbolic model [5]. A discrete domain using 80 uniformly distributed cells in $[-1, 1]$ with initial data: $u = 1$ in $[-0.5, 0]$ and u=0 elsewhere. The system is evolved up to a dimensionless time $t = 0.4$ under constant CFL $= 0.4$. A reference solution of this problem was obtained by re-running this problem with WENO-JS in 10,000 cells. In Fig. 1, one can observe that all fifth-order WENO formulations managed to correctly evolve the nonlinear hyperbolic model in the presence of discontinuities in the solution, however, one will also notice that WENO5–Z solution is closer to the reference solution (in black continuous line) than those of WENO5-JS and WENO5-M.

From the computational perspective, we observe that a WENO-M implementation is on average 20% and 25% more expensive than WENO–JS and WENO–Z respectively. These observations are consistent with those reported in [17,18]. A Matlab implementation to support these observations of this comparison is freely accessible from the first author's Github repository.[1]

The above observations have led us to consider the use of WENO–Z reconstructions in the formulation of the convective terms in Eqs. (10), so that the resulting numerical solver can mainly benefit from improved shock-capture capabilities.

3. Software design

In this section, the design of a multi-GPU solver based on Eq. (11) is presented. We first describe the steps required to evolve the acoustic system in time, as stated in Eq. (13), to later address the implementation of each spatial operator in Eq. (15) as single GPU kernels. Our initial target architecture is a mini-server containing four NVidia Tesla K80 GPU accelerators. Each accelerator is a dual unit such that this single node machine contains in total 8 GPUs, each containing 12GBs of RAM. Table 3 shows the main

[1] https://github.com/wme7/WENO5-X.

Table 3
Hardware specificiations for the Nvidia Tesla K80 GPU.

Feature	Nominal value
Chip	2 × GK210B
Primary clock	562 MHz
Number of SP cores	2 × 2496
Number of DP cores	2 × 832
Peak SP performance	2 × 4.470 TFlop/s
Peak DP performance	2 × 1.455 TFlop/s
Memory clock	2.5 GHz
GDDR5 memory	2 × 12GB
Memory bandwidth	2 × 240GB/s
PCIe Bus	Gen3.0 × 16.

hardware characteristics of this GPU. The present algorithm was written in CUDA C language.

3.1. Data allocation

We start by organizing vector of fields $\boldsymbol{q}$ in a structure of arrays (SoA) format in the CPU memory. In this format, the initial condition of the problem is setup in the complete domain and afterwards each sub-array $q \in \mathbb{R}^N$, where $N = N_x \times N_y \times N_z$, is decomposed in smaller partitions such that they can fit into the global memory of a single GPU. Let us remark that this partition does not modify the data on memory. Fig. 2 depicts the reference of the domain decomposition used in this work. Evidently the number of cells per m-segment is $N_m = N_x \times N_y \times n_{z,m}$ so that $N_z = \sum_m n_{z,m}$. In this work, a one-dimensional domain decomposition is considered along the k-axis which is also the slowest moving index. In our design, we propose that the data inside the GPU memory should be stored as pitched memory arrays and it should be padded along the i-coordinate. The purpose of doing this will become more evident later on. Here, this is done by allocating each of the fields of $\boldsymbol{q}_m$ as pitched buffers with the `cudaMallocPitch()` instruction [8]. This work results in the data of the SoA to be organized as continuous array of 2-d padded slices inside the GPU memory as shown in Fig. 3. Therefore every i, j, k-cell of a single q_m field can be simply accessed as

$$q_{i,j,k} \equiv q[i + pitch * j + (pitch * N_y) * k]. \tag{23}$$

To produce each of the Runge–Kutta stages, it is also necessary to allocate extra buffers for $\boldsymbol{q}_{0,\,m}$ and $\mathcal{L}_m$ in the same manner. By considering the data exchange between GPUs, it is evident that special regions are needed to be appended and prepended to the buffer arrays. Fig. 4 depicts how each of the individual fields in $\boldsymbol{q}_m$, $\boldsymbol{q}_{0,\,m}$ and $\mathcal{L}_m$ are extended to be of size $N_m = N_x \times N_y \times (n_{z,m} + 2r)$. Here r denotes the number of slices that will function as a ghost region for the communication and four-extra buffers of size $N_{exchange} = N_x \times N_y \times r$ needed for the communication strategy introduced in the next subsection.

3.2. MPI and the communication scheme

To achieve computations in multiple GPU architectures, we chose to rely on a *CUDA-aware* [8] Message Passage Interface (MPI) to directly transfer data between GPUs. Moreover, to overlap the computation process with the data exchange communications, we also rely on the concept of CUDA streams. A state-of-the-art strategy for achieving this communication exchange has been already reported in [19] in the context of a forward Euler time stepping. In this work, we extend this idea so that it can be used within a Runge–Kutta time integration scheme in an economic manner.

3.2.1. Convention of regions in the data allocation

As introduced in the previous section, $\boldsymbol{q}_m$, $\boldsymbol{q}_{0,\,m}$ and $\mathcal{L}_m$ buffers have been allocated in the GPU's memory and are slightly larger

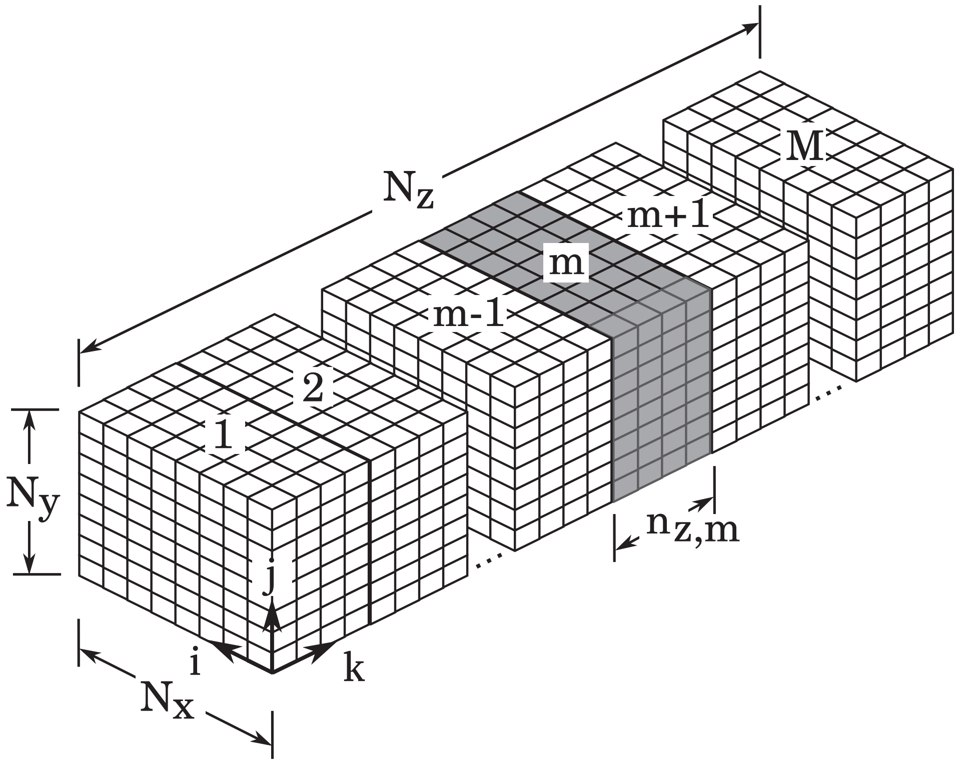

Fig. 2. Decomposition of a single domain array into small partitions to be submitted to the GPU memory.

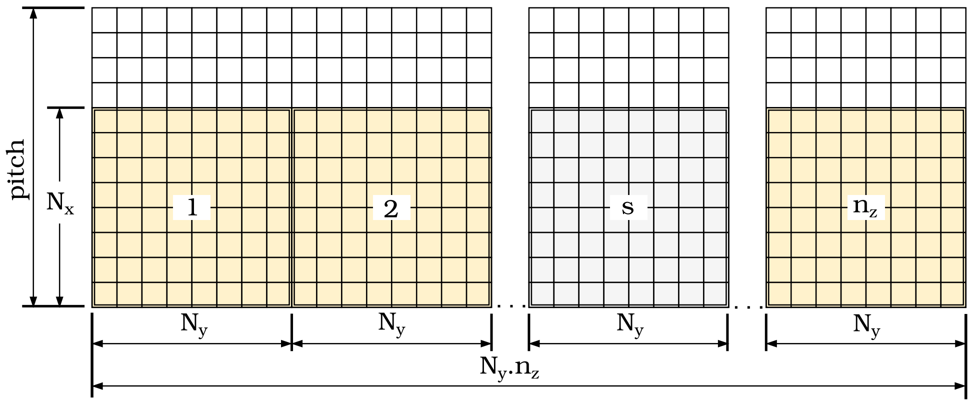

Fig. 3. Storing and padding the data of a single q_m field in the GPU global memory. Notice that this allocation technique can led a suitable for binding of the data with a single or a multiple array of 2-d textures.

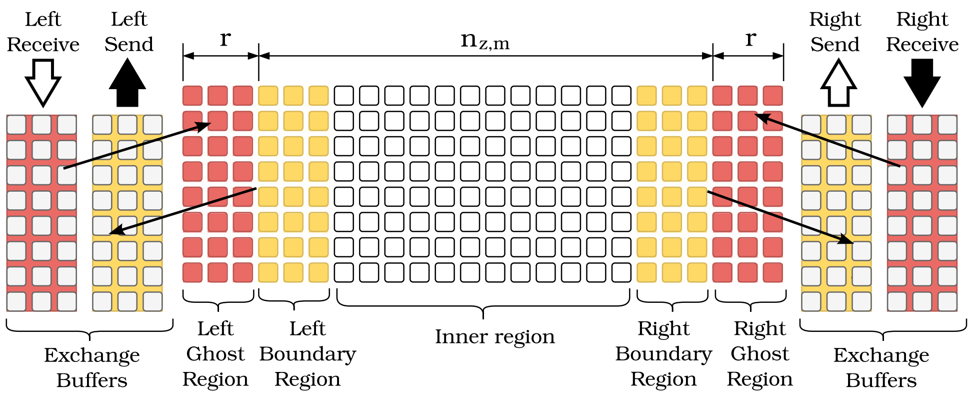

Fig. 4. Convention for the allocation each field in $\boldsymbol{q}_m$, $\boldsymbol{q}_{0,\,m}$ and $\mathcal{L}_m$ of the partition m inside the GPU global memory.

than the q buffers by $2r$ slices that were appended along the k-index. Fig. 4, depicts the regions in which this buffer is *virtually* subdivided. Namely, the *inner region*, located at the center, is where the GPU main kernels will store and accumulate the contributions of the convective and diffusive operators. The data values needed by the neighbor will constitute the *boundary region* (Fig. 4), whereas the data values that are provided by the neighbors constitute the *ghost region*. These regions exist at both ends of the domain and are simply differentiated by the labels *left* and *right*.

3.2.2. A modified communication strategy for the RK scheme

In [19] and other preceding works, data exchange is performed on the buffers containing the updated data, therefore requiring the allocation of different buffers for the old and updated data, namely $\boldsymbol{q}_m^n$ and $\boldsymbol{q}_m^{n+1}$ buffer fields and the swapping of their pointers every timestep. However, by realizing that the communication processes should take place at the level of $\mathcal{L}_m$ buffers instead, we can avoid the need of extra buffers, so we can keep to a minimum the memory requirements of the implementation.

Let us illustrate the communication strategy used in this work for the left-end of the $\mathcal{L}_m$ buffer in rank node m. The data exchange is a three-step procedure that starts with computing the contribution of the operators inside the *left boundary region* so that the updated info is then copied into the *left send buffer*. Secondly, the content of this buffer is transmitted directly to the GPU in the neighbor rank node $m-1$ by using a MPI send instruction. Thirdly, in the neighboring rank, a MPI receive instruction is already waiting to collect the transmitted data into a *right receive buffer* on the memory of the GPU. If the transmission is successful, this data is then copied into the *right ghost region* of the $\mathcal{L}_{m-1}$ buffer. Note that communication process at the right-end of $\mathcal{L}_m$ is mirror symmetric. Once the computation process and data exchange processes are complete at both sides, we will require the Runge–Kutta stage to be evaluated for all the data in $\mathcal{L}_m$ buffers so that the updated data could be directly stored in the $\boldsymbol{q}_m^n$ buffers, ready for the evaluation of the next time step.

From Fig. 4, it is evident that the communications at both ends of the buffer are independent processes, therefore these tasks can be scheduled in different CUDA streams and asynchronous MPI instructions can be used to overlap and ensure that the transmission occurs simultaneously on both sides of the buffer. We can also schedule the computation process in an independent stream so that the computations in the inner region are overlapped with the communication processes. At each end, one stream is used to initialize the computation to the boundary region and to send the data to the neighbor rank node, and another stream is used to start the MPI receive instruction to collect the data from a neighbor rank and copy it into a ghost region. This strategy requires five CUDA streams per rank, whose scheduling is depicted in Fig. 5. Let us remark that the evaluation of a Runge–Kutta stage should only take place until communications and the inner region computations are completed.

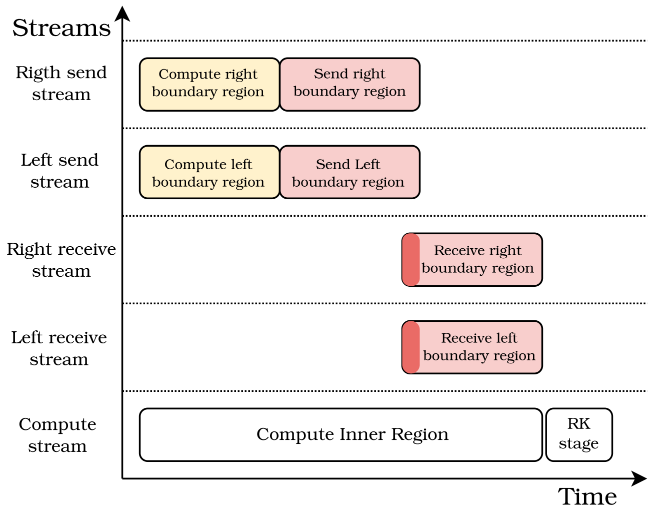
Streams
Rigth send stream
Compute right boundary region
Send right boundary region
Left send stream
Compute left boundary region
Send Left boundary region
Right receive stream
Receive right boundary region
Left receive stream
Receive left boundary region
Compute stream
Compute Inner Region
RK stage
Time

Fig. 5. Streams schedule for overlapping the computation and communications processes during the evaluation of a single Runge–Kutta stage.

Algorithm 1: Processing of Runge–Kutta stages.

```
 1: Make copy of $q^n$ to auxiliary buffer $q^0$ (compute stream)
 2: Set $q_0 \leftarrow q^n$ (virtual step)
 3: for $s = \{0, 1, 2\}$ do
 4:   Compute $\mathcal{L}(q_s)$ on boundary regions (comm. streams)
 5:   Exchange $\mathcal{L}(q_s)$ of boundary regions (comm. streams)
 6:   Compute $\mathcal{L}(q_s)$ on inner region (compute stream)
 7:   Synchronize all CUDA streams
 8:   Compute Runge–Kutta stage (compute stream)
          $\mathrm{RK}_s(q_s, q^0, \mathcal{L}(q_s)) \to q_{s+1}$
 9:   Synchronize compute stream
10: end do
11: Set $q^{n+1} \leftarrow q_3$ (virtual step)
```

3.3. Algorithm

The numerical implementation of Eq. (13) yields a fully explicit algorithm for the evolution of the data in every partition m. By using a Runge–Kutta time integration algorithm, the most expensive part of such algorithm is the evaluation of the spatial operator $\mathcal{L}_m$. Algorithm 1 describes the main steps required to evolve information in partition m from the present time step t^n to the next t^{n+1} using a third-order low-storage Runge–Kutta scheme.

To reach a target output time, t^{final}, Algorithm 1 must be executed n-times, where n denotes the number of timesteps, Δt. For simplicity we only consider a fixed timestep Δt, which is established from a conservative CFL condition at the beginning of the computation based on an estimation of the maximum acoustic pressure in the initial condition of the system. The implementation of a dynamic time step Δt typically requires a reduction process to find the maximum value of pressure inside the domain. Here, we postpone the implementation of this feature to an extension of the present work.

3.4. Computing kernels

In this section, the procedure for developing and optimizing the spatial GPU kernels needed to approximate the $\mathcal{L}_m$ operator on partition m, is described. As mentioned in Section 2, the convective operators are based on a 5th-order WENO–Z reconstruction while the diffusive operators are based on the 4th-order finite-difference stencil of Eq. (19). For the sake of clarity the implementations of the convective operators in CUDA C language are denoted in the figures as the WENO5$_Z$ $\boldsymbol{f}_x$, WENO5$_Z$ $\boldsymbol{g}_y$, WENO5$_Z$ $\boldsymbol{h}_z$ kernels. Similarly, the diffusive contribution of $\mathcal{D}\nabla^2\boldsymbol{q}$ is denoted as the Laplacian (∇^2) kernel. Lastly, we also consider a CUDA C implementation of the Runge–Kutta stages as presented in Eq. (14). This implementation will be referred as the *RK stage* kernel.

3.5. Optimization approach

To optimize the performance of the spatial and RK stage kernels we require firstly, to define a baseline implementation for all kernels, secondly, to build a roofline model [6] of the target architecture, and lastly, to set the thermoviscous acoustic model

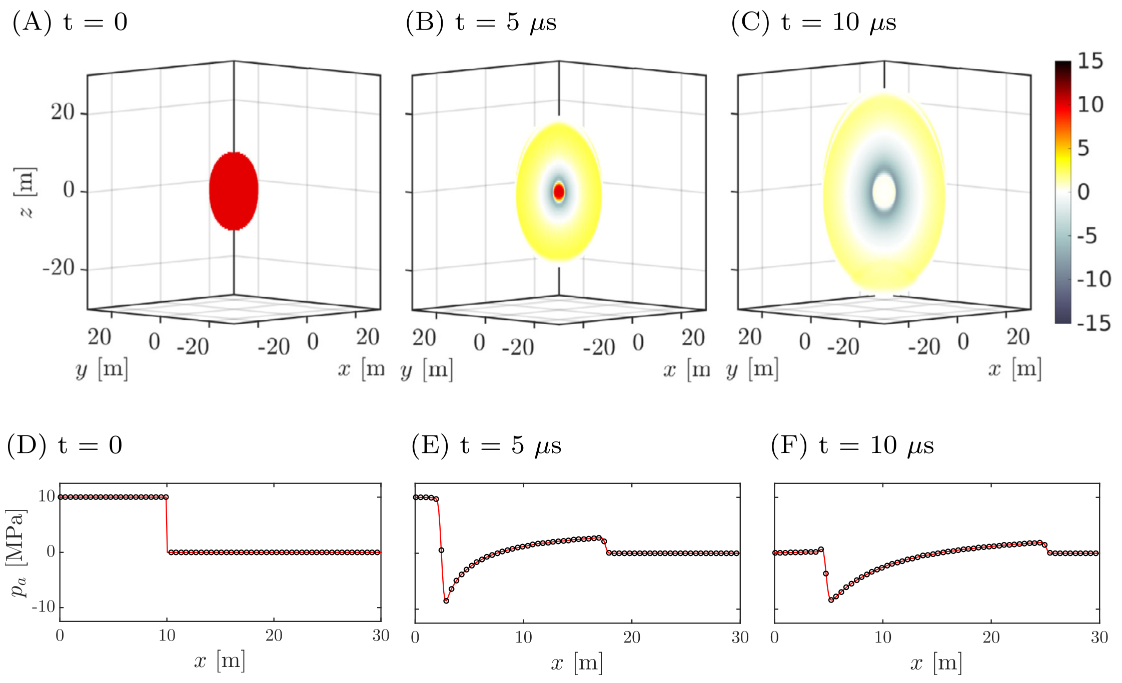

Fig. 6. Collapse of a high-pressure region modeled under the thermoviscous acoustic model in Eqs. (10) is depicted for at time $t = 0$, 5 and 10 μs. Color scale provided in MPa. In sub-figures (A), (B) and (C), the pressure field is depicted using a slice cut on the volume of data. Given the symmetry of the solution, in sub-figures (D), (E) and (F), the profile in the x-direction is presented. The reference solution, in continuous red line, is obtained by solving the same IVP in a domain of $[512]^3$ with a multi-GPU solver. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

in Eqs. (10) as an initial value problem (IVP). Any new strategy, memory optimization and/or reformulation that is introduced to the baseline kernels is measured and contrasted with that of the baseline. By using the roofline as a map of the performance (in GigaFLOP/s) and operational intensity (in FLOPs/byte), the overall implementation of acoustic solver can then be gradually optimized. As in the implementation of [20], we initially concentrate on the optimization of the solver for a single node. Once we are satisfied with the performance of the implementation in a single node, the solver is then extended to the multiple node context.

In this work, the baseline implementation for all the kernels is based on Micikevicius' single-pass strategy [21] working on coalesced memory, as introduced earlier in Section 3.1. The roofline model of a single GPU is built and used as a reference for the maximum performance achievable in a single node. Lastly, a symmetrical implementation of thermoviscous acoustic model in Eqs. (10) is formulated as an initial value problem (IVP). Here, a domain where $(x, y, z) \in \Omega \in [-30, 30]^3$ mm^3 filled with degassed water is assumed. Inside it, a high pressure region given by

$$\begin{cases} u(x,y,z) = v(x,y,z) = w(x,y,z) = 0, \\ p(x,y,z) = \begin{cases} 10 \text{ MPA} & \text{if } \sqrt{x^2+y^2+z^2} \le 10 \text{ mm} \\ 0 & otherwise, \end{cases} \end{cases} \tag{24}$$

it set. For simplicity, Dirichlet homogeneous conditions are set along the boundary, $\partial\Omega$. To provide enough work for the measurements, the domain is discretized with a grid of $[256]^3$ nodes. The expert reader would recognize that the IVP in Eq. (24) is the analog of a Riemann problem formulated for the present 3-d acoustic model. In Fig. 6, the solution of the high-pressure region collapse is depicted for times $t = 0$, 5 and 10 μs.

3.6. Optimization process

We aim for optimal kernels that approximate the spatial contributions $\mathcal{L}_m$ in partition m using *double precision* computations. Based on the traditional approach, implementations based on *spatial blocking* [22], *shared memory* and *texture memory* were developed.

For each kernel implementation, the total bandwidth (Q) consumed by an individual kernel, the total number of operations performed (W) and the average execution time (T) are measured. Therefore the performance (W/T) and the operational intensity (Q/W) can be established. For instance, the performances of each of the baseline kernels are depicted with black markers in Fig. 7. The performance parameters are then used for comparison and guidance criteria for developing improved versions from the baseline kernels. As a result, the best performing kernels are simply selected for the final implementation.

In this work, optimizations based on texture memory techniques were pursued. However, it is found that 2D texture arrays are not yet enabled to `fetch` natively double precision data. In [23], a workaround transformation has been presented via the `__hiloint2double()`[8] function. However, our measurements indicate that any improvement obtained by using texture memory to read double precision data is overshadowed by having to transform the data from `int2` into `doubles` as proposed by the workaround strategy. Therefore, we simply concentrate on blocked (denoted by upper index b and red markers in Fig. 7) and shared memory (upper index sh and orange markers in Fig. 7) implementations of the GPU kernels.

The result of this optimization process, as depicted in Fig. 7, shows that kernels based on the single-pass associated with blocked strategy performs better for the convective operators in the j- and k-direction, while a simple shared memory strategy is optimum for the convective operator in the i-direction. A careful analysis reveals that this is due to the fact that single-pass reduces the need of caching non-coalesced data from our pitched allocation. However, if the data is perfectly coalesced, it can be loaded faster into the a shared memory cache, so that the WENO–Z reconstructions can benefit from faster access to the data. For the diffusive operator, we notice that a single-pass with a blocking strategy

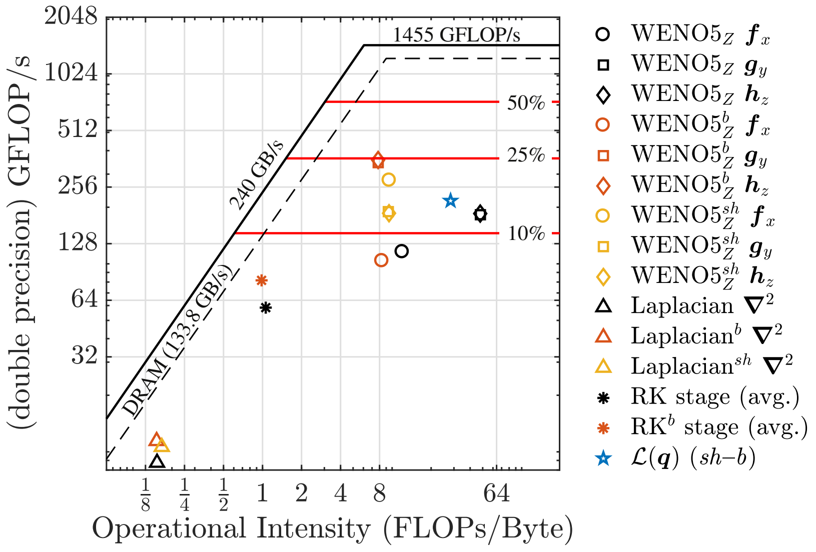

Fig. 7. Roofline for the Tesla K80 GPU with measured performances of each individual kernel. Black color markers correspond to the baseline implementation, red color, shows the optimized performance based on spatial blocking, orange color, shows the optimized performance using both spatial blocking and shared memory. Circles (◯), squares (□) and rhombus (◇) represent convective operators in x, y and z directions correspondingly (operators $\boldsymbol{f}_x$, $\boldsymbol{g}_y$, $\boldsymbol{h}_z$ introduced in Eqs. (2) and (3). L(q) is the total spatial operator given in Eq. (4), which includes both convective and diffusive parts. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

provides the best performance. Similarly, a pure blocked implementation of the RK stages kernel yields an improved performance. We observe that the best performance for all of the kernels is obtained using thread blocks that correspond to 4 warps. Moreover, we test for any improvement that results from limiting the number of registers per block at compile time and forcing the spilled register to be stored in L1 cache, e.g. providing the argument `-Xptxas -maxrregcount=32 -dlcm=cg` to the compiler. However, in our measurements the best performance was attained without allowing any register spill.

Table 4 shows the number of registers per thread and the level of occupancy for each of the final kernel obtained from the optimization process in the present work. In the following sections the final measurements of the best performing kernels and the overall performance of the implementation is reported.

4. Implementation performance

The performance measurements of the kernels for the multi-GPU acoustic solver were evaluated on a mini-server machine that contains two Xeon E5-2630v3 CPUs and four Nvidia Tesla K80 GPUs. Here, each K80 unit is in reality a dual unit, therefore this machine contains in total 8 GK210 chips to our disposal (see Table 3). In the present implementation, we rely on GCC 4.8.5 and CUDA 7.5 toolkit. The Nvidia profiler, `nvprof`, provided in the CUDA toolkit was used to capture the performance data necessary to get Fig. 7. Let us remark that all the present measurements were obtained using double precision computations.

4.1. Single node performance

In this section, we report the performance of the implemented GPU kernels by means of the roofline model [6]. In this work, the empirical roofline benchmark suite[2] is used to build the roofline ceiling of the memory-bound region. We measured a memory (DRAM) bandwidth of 133GB/s (55% of the nominal peak) and a double precision top performance of 1.23 TFLOP/s (85% of the nominal peak). ECC was enabled for the memory bandwidth measurements.

Fig. 7 shows the roofline for a single chip of the K80 GPU and the performances of the kernels obtained on the K80 accelerator are depicted. The black ceilings correspond to the measurements based on the benchmark suite mentioned above, while the references for 10%, 25% and 50% of the nominal peak performance are indicated by the red rooflines. We use the Nvidia profiler to count the number of floating point instructions and measure the runtime of the Laplacian^b kernel. From these number we determine a performance of 10.6 GFlop/s. Similarly, it is found that the Runge–Kutta (RK^b) stage kernel has an average performance of 85.7 GFlop/s. Given that the WENO5$_Z$ reconstruction kernels have a high floating point density, therefore it is not surprising that they exhibit a higher performance among other kernels. The profiler shows that the WENO5^b_Z $\boldsymbol{g}_y$ and WENO5^b_Z $\boldsymbol{h}_z$ kernels achieve 365.7 GFlop/s while the WENO5$^{sh}_Z$ $\boldsymbol{f}_x$ achieves 277.9 GFlop/s. All selected WENO5$_Z$ kernels exhibit an operational intensity of $\sim$8 Flop/Byte. Let us remark that the convective kernels include the computation flux splitting by the Lax–Friedrich method. Based on these measurements, an overall performance for the single partition computation of $\mathcal{L}_m$ is estimated to be 201 GFLOP/s with an operational intensity of 28 FLOPs/byte. The roofline model evidently shows that there is still plenty of room for improvements, however, these improvements require more involving strategies. Our future work will focus on this aspect. Lastly, we notice that the performance values of convective kernels also reflects performance of a single WENO–Z reconstruction. This is to be expected, as each kernel performs the reconstruction of a velocity and pressure field in a serial manner, as defined in Eqs. (2) and (3). Moreover, our measurements verify that each WENO–Z reconstruction exhibits an operational intensity of 5.2 Flop/Byte (including the flux splitting). This value contrasts well with the 6 Flop/Byte reported for the WENO-JS reconstruction in [20] and indicates that WENO–Z formulation is computationally cheaper than the classical WENO-JS formulation. Here, to the best of the authors' knowledge, this is the first time that measurements of performance and numerical intensity of WENO–Z implementa-

[2] https://bitbucket.org/berkeleylab/cs-roofline-toolkit.

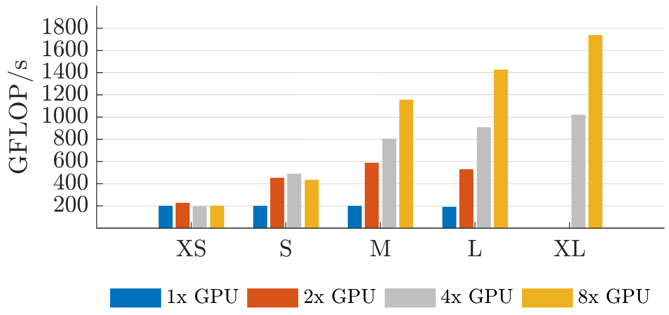

Fig. 8. Performance measurements of a multi-GPU solver based on model Eqs. (10). Here, the initial condition in Eq. (24) is evolved on uniformly distributed domains of different sizes. For simplicity they are denominated extra-small (SX), small (S), medium (M), large (L) and extra-large (XL). The sizes of these grids are defined in Table 4.

Table 4
Register usage per thread and occupancy for thread blocks.

Kernel	Warps	Registers per thread	Occupancy
WENO5$^{sh}_{Z}$ ($\boldsymbol{f}_x$)	4	60	72.3%
WENO5$^{b}_{Z}$ ($\boldsymbol{g}_y$)	4	78	66.9%
WENO5$^{b}_{Z}$ ($\boldsymbol{h}_z$)	4	78	67.1%
Laplacian^b (∇^2)	4	35	89.7%
RK^b stage (avg.)	4	28	99.5%

tions on a GPU device have been explored and reported in the literature.

4.2. Multiple node performance

For simplicity reasons the domain has been decomposed along the k-direction, resulting in a one-dimensional (1-d) decomposition. One benefit of a 1-d decomposition is that kernels developed for a single-GPU can be used without additional modification. Nevertheless, this 1-d decomposition can be suboptimal for extremely large problems size that span across many nodes. In this work, given the chosen target machine, the present 1-d decomposition will suffice. Fig. 8, depicts the performance of the acoustic solver using 1, 2, 4 and 8 GPUs for evolving the initial condition (24) in different domain sizes. Firstly, we notice that the single-GPU tests, as expected, exhibit a computational performance of ∼200 GFLOP/s. Secondly, given the size of the tests, it is noticed that as long as the computation part is big enough, the communication part can be entirely hidden. So that the computation only depends on how fast each GPU can compute for $\mathcal{L}_m$. This also proves that with the increase of the problem size the performance improves. Lastly, it is interesting to notice that for small sized problems, using two GPUs exhibit an improved performance. Here, we simply speculate that it must be related to the internal communication between the two GPU chips contained in every K80 unit.

To get a better feeling of the meaning of these measurements we compare them with those obtained using a single CPU-OpenMP implementation of the same algorithm. A gaming PC loaded with an Intel Core i7-7700K of 4.20 GHz, Socket 1151, 8MB Cache with 8 threads is used for this test. The computation time for evolving the IVP up to time 10 µs under different domain sizes is recorded, so that the average computation time of a single timestep can be compared in Tables 5 and 6. The comparison of computation time of a single timestep shows that the multi-GPU implementation can perform up to 99× faster than that of an OpenMP single CPU implementation using all 8 threads, and 486× faster than that of a single-threaded CPU implementation.

Table 5
Computation time per timestep obtained for an OpenMP solver of thermoviscous acoustic model in Eqs. (10). Computations were performed on a Intel Core i7-7700K CPU working at a frequency of 4.20 GHz.

		Time per iteration (s)	
Test	No. cells	1 thread	8 threads
XS	$32\times32\times64$	0.0596	0.0107
S	$64\times64\times128$	0.6441	0.1029
M	$128\times128\times256$	2.9990	0.9366
L	$256\times256\times512$	35.5667	9.2204
XL	$512\times512\times1024$	375.9982	76.9223

5. Numerical simulation

We are interested in modeling the wave propagation generated by focusing piston transducer typical for highly-intense focused ultrasound (HIFU) surgery [25,26]. The device is assumed to have a radius of curvature, A, of 62.6 mm, and aperture length, a, of 32 mm working. Moreover we assume that it generates a continuous sinusoidal signal at a constant frequency, f, of 1.06 MHz, with amplitude, P_0, of 1 MPa that propagates through a media of degassed water: see Fig. 9(a). The degassed water is assumed to be at 20 °C and its acoustic properties have been summarized in Table 1.

The proposed multi-GPU solver for Eqs. (10) is used to simulate the nonlinear propagation of ultrasound in a thermoviscous media. Here, to represent the geometry of the focused piston, a collection of point sources is arranged in a manner to represent the profile of the transducer: see Fig. 9(b). Moreover, in liquid fluids the ratio $c_p/c_v \approx 1$. Therefore, the contribution of the diffusion coefficient in the pressure equation can be neglected. As a result, these assumptions yield a thermoviscous acoustic system of form

$$\begin{cases} \dfrac{\partial p_a}{\partial t} + \dfrac{\partial (k_a u_a)}{\partial x} + \dfrac{\partial (k_a v_a)}{\partial y} + \dfrac{\partial (k_a w_a)}{\partial z} = S, \\ \dfrac{\partial u_a}{\partial t} + \dfrac{1}{\rho_0}\dfrac{\partial p_a}{\partial x} = d_2\left(\dfrac{\partial^2 u_a}{\partial x^2} + \dfrac{\partial^2 u_a}{\partial y^2} + \dfrac{\partial^2 u_a}{\partial z^2}\right), \\ \dfrac{\partial v_a}{\partial t} + \dfrac{1}{\rho_0}\dfrac{\partial p_a}{\partial y} = d_2\left(\dfrac{\partial^2 v_a}{\partial x^2} + \dfrac{\partial^2 v_a}{\partial y^2} + \dfrac{\partial^2 v_a}{\partial z^2}\right), \\ \dfrac{\partial w_a}{\partial t} + \dfrac{1}{\rho_0}\dfrac{\partial p_a}{\partial z} = d_2\left(\dfrac{\partial^2 w_a}{\partial x^2} + \dfrac{\partial^2 w_a}{\partial y^2} + \dfrac{\partial^2 w_a}{\partial z^2}\right), \end{cases} \tag{25}$$

where k_a is the corrected bulk modulus and d_2 is the viscous coefficient introduced in Section 2. However, for ultrasound signals, it is well known that the viscous coefficient d_2 can be alternatively obtained as

$$d_2 = \frac{2c_0^3\alpha}{\omega^2}, \tag{26}$$

where α is the absorption coefficient of the media in Np/m, c_0 is the speed of sound in the media and ω is the angular frequency of the signal. Lastly, S: $S(x, y, x, t)$ is a source term that represents a scalar field where singular discrete points introduce an acoustic perturbation into the system. This scalar field is then defined as

$$S(x,y,z,t) = s(t)\,\boldsymbol{\delta}(x-x_l^*)\,\boldsymbol{\delta}(y-y_l^*)\,\boldsymbol{\delta}(z-z_l^*). \tag{27}$$

where (27), $s(t) = P_0\sin(\omega t)$ is the sinusoidal signal to be introduced into the system which is characterized by its amplitude, P_0, and angular frequency, $\omega = 2\pi f$. The set $\{(x_l^*, y_l^*, z_l^*)\}$, l=1,2,...,$L$; represents a list of the spatial coordinates of discrete points that correspond to the radial profile of the transducer. Consequently $\boldsymbol{\delta}(x)\boldsymbol{\delta}(y)\boldsymbol{\delta}(z)$ represents a scalar field where the few non-zero elements corresponds to those cells whose coordinates are in the list of discrete points. For simplicity, we will refer to the formulation

Table 6
Comparison between the multi-GPU (MPI+CUDA C) solver and the OpenMP solver.

Test	Time per iteration (s)				Speed up w.r.t. OpenMP solver			
	1 GPU	2 GPUs	4 GPUs	8 GPUs	1 GPU	2 GPUs	4 GPUs	8 GPUs
XS	0.0043	0.0032	0.0038	0.0037	1.22	1.64	1.39	1.44
S	0.0156	0.0059	0.0055	0.0061	3.29	8.75	9.41	8.38
M	0.1349	0.0390	0.0285	0.0198	3.47	12.01	16.45	23.71
L	1.1590	0.3581	0.2079	0.1327	6.63	21.45	36.95	57.91
XL			1.3118	0.7721			58.64	99.62

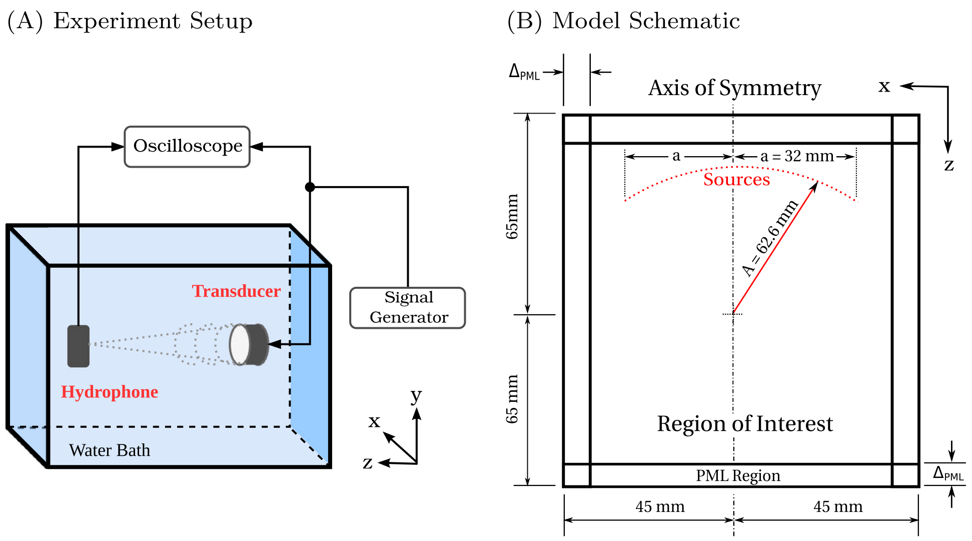

Fig. 9. Acoustic propagation in homogeneous media. In (a), Experimental setup for measuring the high-intensity ultrasound propagation in degassed water using hydrophone. In (b), a top view of the 3-d domain formulation. Here, the distribution of PML layers (or regions) and the arrangement of the point sources to represent the focused device inside the *region of interest* are depicted.

in Eq. (25) as the thermo-acoustic system (TAS) model for degassed water.

For this simulation the propagation of ultrasound waves is assumed to take place in an open domain. Numerically, this done by implementing perfectly matching layers (PML), in the sense of [7]. PML regions are considered for surrounding the entire domain to avoid unwanted reflection from happening at the boundaries of the discrete domain. In the TAS model, this is reflected by introducing absorption terms as

$$\begin{cases} \dfrac{\partial p_a}{\partial t} + \dfrac{\partial (k_a u_a)}{\partial x} + \dfrac{\partial (k_a v_a)}{\partial y} + \dfrac{\partial (k_a w_a)}{\partial z} = S - (\sigma_x + \sigma_y + \sigma_z) p_a, \\ \dfrac{\partial u_a}{\partial t} + \dfrac{1}{\rho_0}\dfrac{\partial p_a}{\partial x} = d_2\left(\dfrac{\partial^2 u_a}{\partial x^2} + \dfrac{\partial^2 u_a}{\partial y^2} + \dfrac{\partial^2 u_a}{\partial z^2}\right) - \sigma_x u_a, \\ \dfrac{\partial v_a}{\partial t} + \dfrac{1}{\rho_0}\dfrac{\partial p_a}{\partial y} = d_2\left(\dfrac{\partial^2 v_a}{\partial x^2} + \dfrac{\partial^2 v_a}{\partial y^2} + \dfrac{\partial^2 v_a}{\partial z^2}\right) - \sigma_y v_a, \\ \dfrac{\partial w_a}{\partial t} + \dfrac{1}{\rho_0}\dfrac{\partial p_a}{\partial z} = d_2\left(\dfrac{\partial^2 w_a}{\partial x^2} + \dfrac{\partial^2 w_a}{\partial y^2} + \dfrac{\partial^2 w_a}{\partial z^2}\right) - \sigma_z w_a, \end{cases} \tag{28}$$

where $\sigma_x(x)$ is an absorption coefficient that is non-zero only inside a sub-region (layer) in the vicinity of the boundary of the domain. See Fig. 9(b). The thickness of this layer is $\Delta_{PML} = N\Delta x$, where N is the PML thickness in terms of nodes and Δx is the spacing of the uniform grid in the x-direction. For a smooth transition for the region of interest to the boundary layer $\sigma_x(x)$ is defined as the parabolic profile as

$$\sigma_x(x) = \sigma_0\left(\frac{x}{\Delta_{PML}}\right)^2, \quad \sigma_0 = \log\left(\frac{1}{R}\right)\frac{3c_0}{2\Delta_{PML}}, \tag{29}$$

where σ_0 is the maximum damping parameter which is function of the theoretical reflection coefficient R. From [24], the recommended values of R are

$$R = \begin{cases} 0.01 & \text{if } N = 5, \\ 0.001 & \text{if } N = 10, \\ 0.0001 & \text{if } N = 20. \end{cases} \tag{30}$$

The definition of $\sigma_y(y)$ and $\sigma_z(z)$ are analogous to $\sigma_x(x)$.

Strictly speaking, given the nonlinear nature of the TAS model, it is not expected that use of PML layers will help to perfectly avoid small reflections from incoming waves with high amplitude. However, by considering slightly thicker PML regions it is possible to reduce those reflection to almost zero. In this work, PML layers with 20 nodes in width are considered at all faces of the Cartesian domain.

The domain for the simulation is assumed to span an area of $90 \times 90 \times 130$ mm^3. Moreover, it is assumed to be discretized with the uniform distribution of $832 \times 832 \times$ 1300 cells in the x, y and z direction, respectively. The number of cells used in this discrete domain is approximately 0.9 billion, and leads to a resolution of 16 cells per wave length. Moreover, a total of 63,553 discrete point sources are used to represent the focused device.

The TAS model is evolved up to time $t = 70$ µs. The timestep is set to $\Delta t = 12.5$ ns (or $\times 10{-}9$ s), as a result the simulation requires 5600 iterations to reach the output time. Using the multi-GPU acoustic solver with double precision computations on 8 GPU nodes, the entire computation takes 2.52 h. The unsteady acoustic propagation generated by a HIFU transducer is simply depicted in Fig. 10 for time $t = 70$ µs. Here, any acoustic reflection at the boundary has been dampened by the PML boundaries allowing the model to evolve under free-flow conditions.

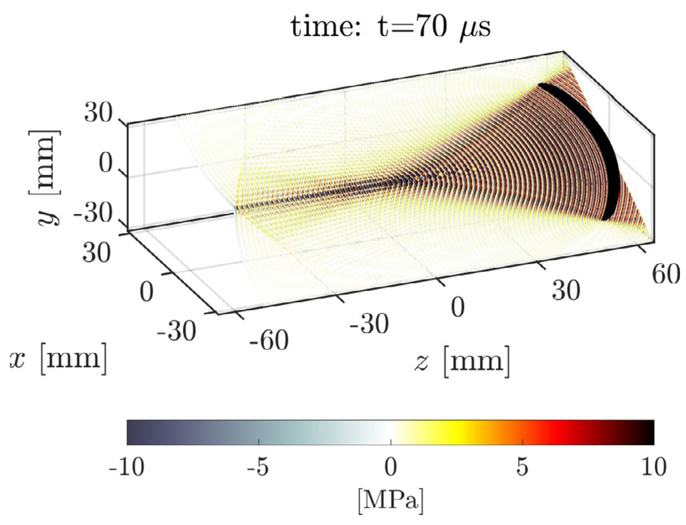

Fig. 10. Simulation of highly-intense acoustic propagation generated by a focused device represented by point sources in degassed water. The solution of the unsteady acoustic pressure field is interpolated from the full 3-d domain on a inclined plane along the domain and illustrated for time $t = 70$ μs.

6. Concluding remarks

We have presented a double-precision explicit solver for modeling the propagation of acoustic fluctuations in thermoviscous media. The present solver is based on a *shared memory–blocking* strategy that improves the performance of stencil computations in GPUs. Moreover, by using a fifth-order WENO–Z scheme the present solver provides improved shock capturing capabilities over preceding works. In this work, GPU implementations of WENO–Z reconstructions were studied, and their performance and operational intensity on a GPU are reported for first time. Lastly, an extension of the multi-stream communication strategy in [19] to be used with Runge–Kutta schemes, was presented. Although, in this work we targeted the acoustic system presented in [4], the approach used is rather general and can be straightforwardly applied to other convective–diffusive problems such as the Navier–Stokes equations. The overall floating point performance of the right-side computation is 201 GFLOP/s (in double precision) on a single node associated with a single chip of the dual K80 GPU. The observed performance corresponds approximately to 13.8% of the nominal peak performance (in double-precision) of a single K80 GPU chip. When solving on four K80 dual units (8 GPUs in total), a performance of 1730 GFLOP/s was achieved.

Acknowledgments

We express our gratitude to Dr. Lusher from University of Southampton for recommending a comparison of the traditional WENO methods with the new WENO–Z formulation during the ParCFD conference of 2017. This research was supported by the Ministry of Science and Technology, R.O.C., under the grant MOST-105-2221-E-400-005 and by National Health Research Institutes's project BN-106-PP-08.

References

[1] Hamilton MF, Blackstock DT, et al. Nonlinear acoustics, 1. San Diego: Academic press; 1998.
[2] Solovchuk M, Sheu TW-H, Thiriet M. Multiphysics modeling of liver tumor ablation by high intensity focused ultrasound. Commun Comput Phys 2015;18(04):1050–71.
[3] Aanonsen SI, Barkve T, TjØtta JN, et al. Distortion and harmonic generation in the nearfield of a finite amplitude sound beam. J Acoust Soc Am 1984;75(3):749–68.
[4] Diaz MA, Solovchuk MA, Shue TWH. A conservative numerical scheme for modeling nonlinear acoustic propagations in thermoviscous homogeneous media. J Comput Phys 2018. https://doi.org/10.1016/j.jcp.2018.02.005.
[5] Jiang G-S, Shu C-W. Efficient implementation of weighted eno schemes. J Comput Phys 1996;126(1):202–28.
[6] Williams S, Waterman A, Patterson D. Roofline: an insightful visual performance model for multicore architectures. Commun ACM 2009;52(4):65–76.
[7] Berenger J-P. A perfectly matched layer for the absorption of electromagnetic waves. J Comput Phys 1994;114(2):185–200.
[8] Nvidia C. Cuda c programming guide v8.0. Nvidia Corporation; 2017.
[9] Borges R, Carmona M, Costa B, Don WS. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. J Comput Phys 2008;227(6):3191–211.
[10] Haynes WM. CRC handbook of chemistry and physics. CRC press; 2014.
[11] Beyer R. Nonlinear acoustics: acoustical society of America. NY: American Institute of Physics; 1997.
[12] Holmes M, Parker N, Povey M. Temperature dependence of bulk viscosity in water using acoustic spectroscopy. In: Journal of physics: conference series, 269. IOP Publishing; 2011. p. 012011.
[13] Jakevicius L, Demcenko A. Ultrasound attenuation dependence on air temperature in closed chambers. Ultragarsas 2003;63(1):18–22.
[14] Parker KJ. Ultrasonic attenuation and absorption in liver tissue. Ultrasound Med Biol 1983;9(4):363–9.
[15] Williamson J. Low-storage Runge–Kutta schemes. J Comput Phys 1980;35(1):48–56.
[16] Henrick AK, Aslam TD, Powers JM. Mapped weighted essentially non-oscillatory schemes: achieving optimal order near critical points. J Comput Phys 2005;207(2):542–67.
[17] Castro M, Costa B, Don WS. High order weighted essentially non-oscillatory weno-z schemes for hyperbolic conservation laws. J Comput Phys 2011;230(5):1766–92.
[18] Brehm C, Barad MF, Housman JA, Kiris CC. A comparison of higher-order finite-difference shock capturing schemes. Comput Fluids 2015;122:184–208.
[19] Sourouri M, Gillberg T, Baden SB, Cai X. Effective multi-gpu communication using multiple cuda streams and threads. In: 2014 20th IEEE international conference on parallel and distributed systems (ICPADS). IEEE; 2014. p. 981–6.
[20] Wermelinger F, Hejazialhosseini B, Hadjidoukas P, Rossinelli D, Koumoutsakos P. An efficient compressible multicomponent flow solver for heterogeneous cpu/gpu architectures. In: Proceedings of the platform for advanced scientific computing conference. ACM; 2016. p. 8.
[21] Micikevicius P. 3d finite difference computation on gpus using cuda. In: Proceedings of 2nd workshop on general purpose processing on graphics processing units. ACM; 2009. p. 79–84.
[22] Maruyama N, Aoki T. Optimizing stencil computations for nvidia kepler gpus. In: Proceedings of the 1st international workshop on high-performance stencil computations, Vienna; 2014. p. 89–95.
[23] Meng C, Wang L, Cao Z, Ye X, Feng L-L. Acceleration of a high order finite-difference weno scheme for large-scale cosmological simulations on gpu. In: Parallel and distributed processing symposium workshops & PhD forum (IPDPSW), 2013 IEEE 27th international. IEEE; 2013. p. 2071–8.
[24] Collino F, Tsogka C. Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media. Geophysics 2001;66(1):294–307.
[25] Solovchuk M, Sheu TW, Thiriet M. Simulation of nonlinear Westervelt equation for the investigation of acoustic streaming and nonlinear propagation effects. J Acoust Soc Am 2013;134(5):3931–42.
[26] Solovchuk MA, Thiriet M, Sheu TW. Computational study of acoustic streaming and heating during acoustic hemostasis. Appl Ther Eng 2017;124:1112–22.