MODELING OF THE 3D UNSTEADY MULTISCALE MULTIPHASE FLUID FLOW WITH SHOCKS: NUMERICAL METHODS AND IMPLEMENTATION ALGORITHMS

Korneev B., Levchenko V. PhD.
Keldysh Institute of Applied Mathematics RAS, Moscow, Russian Federation
boris.korneev@phystech.edu, lev@keldysh.ru

Abstract: *Numerical simulation is widely applied in the fluid dynamic research. Unsteady multiphase flow with high Mach and Reynolds numbers has high physical and mathematical fidelity and a special approach of its correct modeling is required. In this paper the appropriate numerical methods and special algorithms of implementation are developed for high-performance numerical modeling using the modern computational systems. The results of validation tasks and problems of practical importance are presented.*

Keywords: COMPUTATIONAL FLUID DYNAMICS, GPU COMPUTING, LRnLA ALGORITHMS, SHOCK WAVES, DROPLET BREAKUP

Introduction

Since the second half of the twentieth century, high-performance computer technologies have been widely used to solve problems in computational fluid dynamics [1]. At the moment, computational fluid dynamics (CFD) is one of the main methods for solving engineering problems of designing parts, devices and other objects with the required fluid dynamic properties.

For high-precision modeling "from the first principles," meaning without specifying and narrowing the mathematical formulation of the problem, it may be necessary (but not sufficient) to use detailed grids and a large number of iterations in the calculation. Let some estimates be given. The numerical region is characterized by some size range, which is usually determined by the characteristic spatial scale for the simulated process. The number of discrete elements along one axis is usually has to be taken in the range of 10^2 - 10^3 or more. Consequently, the number of cells used for a three-dimensional numerical experiment is estimated from 10^6 and can reach 10^9 - 10^{12} and more. The number of iterations in time is determined by the characteristic time scale for the considered process, taking into account the restrictions on the time step, for example, the Courant condition. The number of time steps can be up to 10^6 or more for actual tasks. Thus, the number of operations for calculating a single cell of a numerical scheme is 10^{12}-10^{18} for one numerical experiment. The operation of calculating a single cell of a numerical scheme may require dozens of elementary operations of addition-multiplication for the simplest explicit schemes up to tens of thousands, depending on the complexity of the numerical scheme.

To solve problems of this scale [2], it is necessary to use modern computer hardware with a complex hierarchical structure of computing units and memory system. From set of modern computing systems the ones equipped with graphical accelerators are outstanding due to their high performance at a relatively low cost [3; 4]. For the effective use of such computing systems it is necessary to develop and use suitable calculation algorithms that take into account their features. So, effective implementation must be adapted to the hierarchical memory structure from the CPU memory to the registers CPU/GPU without creating bottlenecks that slow down the speed of calculation [5]. In addition, the problem of efficient parallelization of the algorithm, taking into account the massive parallelism of the GPU, remains relevant. A separate issue remains the requirement of saving memory when building an implementation algorithm, with the help of which it is possible to simulate a domain with as large as possible number of cells within the available memory of the given computer. Due to the fact that the video card's memory is less than the RAM of the CPU, the video card's memory is often not enough to store the entire computing area, which makes the problem to exchange between the GPU memory and the CPU one. To build an effective implementation, it is advisable to take into account these issues.

The content of the paper is the following. In the next section the mathematical and numerical model is set. Then the proper implementation algorithm is described. Third section is devoted to the validation of the program based upon the algorithm and numerical scheme. In the following section the simulation results of some real tasks of practical importance is considered. In the conclusion we summarize the research.

1. Governing equations and numerical scheme

1.1 Equations of gas dynamics

The system of Navier-Stokes equations describing the dynamics of a viscous compressible fluid or gas in three-dimensional space [6], is represented as

$$\frac{\partial \mathrm{U}}{\partial \mathrm{t}} + \frac{\partial \mathrm{F_i}}{\partial \mathrm{x_i}} = 0,$$

where $\mathrm{U} = (\rho, \rho \mathrm{u_j}, \mathrm{E})$ are the gas variables, $\left[F(U)_i = \left[\rho\, u_i, \rho\, u_i u_j + P_{ij}, u_i E + P_{ij}\, u^j - \kappa \frac{\partial T}{\partial x^i} \right) \right]$ are fluxes, pressure tensor $\mathrm{P_{ij}} = \mathrm{p}\delta_{\mathrm{ij}} - \mu \left(\frac{\partial \mathrm{u_i}}{\partial \mathrm{x^j}} + \frac{\partial \mathrm{u_j}}{\partial \mathrm{x^i}} \right) - \frac{\partial \mathrm{u_k}}{\partial \mathrm{x_k}} \left(\zeta - \frac{2}{3}\mu \right) \delta_{\mathrm{ij}}$ taking the viscous forces into the account. The system is extended by the equation of state $\mathrm{p} = \mathrm{p}(\rho, \varepsilon)$. More complex equations of state are considered for the fluid mixtures [7].

We rewrite the fluxes to separate pure convective (Euler) and viscous (Navier-Stokes) parts as follows:

$$\frac{\partial U}{\partial t} + \frac{F_i^{Eu}}{\partial x_i} + \frac{F_i^{NS}}{\partial x_i} = 0,$$

where $F_i^{Eu} = [\, \rho\, u_i, \rho\, u_i u_j + p\delta_{ij}, u_i(E + p)\,]$ and

$F_i^{NS} = [0, -\mu \left(\frac{\partial \mathrm{u_i}}{\partial \mathrm{x_j}} + \frac{\partial \mathrm{u_j}}{\partial \mathrm{x_i}} - \frac{\partial \mathrm{u_k}}{\partial \mathrm{x_k}} \left(\zeta - \frac{2}{3}\mu \right) \delta_{\mathrm{ij}} \right), -\mu \left(\frac{\partial \mathrm{u_i}}{\partial \mathrm{x_j}} + \frac{\partial \mathrm{u_j}}{\partial \mathrm{x_i}} \right) \mathrm{u^j} - \frac{\partial \mathrm{u_k}}{\partial \mathrm{x_k}} \left(\zeta - \frac{2}{3}\mu \right) \mathrm{u_i} - \kappa \frac{\partial T}{\partial x^i}]$.

1.2 Runge-Kutta discontinuous Galerkin method

Suppose we construct a numerical solution of the fluid dynamics system in the domain $\mathrm{G} \in \mathrm{R}^3$, $\mathrm{t} > 0$ with boundary $\partial \mathrm{G}$ with some given initial conditions for $\mathrm{t} = 0$ and boundary conditions. We introduce the discretization of the space G using a grid of finite volumes.

In each cell L, we introduce a set of basis functions $\{\varphi^n\}$, and the solution will be sought as an expansion in this basis for each

component of the desired vector $U = u_s(t)\varphi^s(x)$, while the basis coefficients are functions of time.

In this paper, we use a generalization of this method to approximate dissipative terms in the Navier-Stokes equation, known as the LDG method (local discontinuous Galerkin method [8]).We introduce a set of new variables for spatial derivatives in the flux F_i^{NS}

$$W = \frac{\partial u_i}{\partial x_j}, \qquad S = \frac{\partial\, T}{\partial\, x^i};$$

and for them we apply the same expansion in the basis.

Obtained expressions are inserted into the main equations and the condition of the orthogonality of the residual to all basis functions is accepted to obtain the system of ODE on the basis coefficients:

$$\frac{du_m(t)}{dt} + \oiint_{\partial L} (F_i^{Eu} + F_i^{NS})\varphi_m n^i \, d\Sigma - \iiint_L F_i \frac{\partial \varphi_m}{\partial x_i} dV = 0;$$

$$w_m(t) = \oiint_{\partial L} u_i \, n^j \, \varphi_m d\Sigma - \iiint_L u_i \frac{\partial \varphi_m}{\partial x_j} \, dV;$$

$$s_m(t) = \oiint_{\partial L} T \, n^j \, \varphi_m d\Sigma - \iiint_L T \frac{\partial \varphi_m}{\partial x_j} \, dV;$$

Calculation of $\oiint_{\partial L} F_i^{Eu} \, \varphi_m n^i \, d\Sigma$ is made using the specified Riemann solver [9].

Numerical solution of the given ODE is performed using the explicit Runge-Kutta method with the limiter [10, 11].

2. DiamondTorre implementation algorithm

In the introduction of the paper the necessity of the proper algorithm of implementation for the constructed numerical scheme has been noticed. This paper is considered the DiamondTorre algorithm. Originally developed as an optimal locally-recursive non-locally asynchronous algorithm [12] for explicit numerical schemes with the stencil "cross", in this paper DiamondTorre algorithm is being developed the RKDG scheme of the method for solving gas dynamics problems. Despite the fact that the stencil of the RKDG method itself is not a "cross", each of its substage has a substencil "cross" [13].

The calculation of the problem using this algorithm is conveniently thought of as filling the computational area with towers. The essence of each brick of the tower with coordinates (x_i, y_j, t_k) is performing part of the calculation according to a numerical scheme over the entire segment (x_i, y_j, Nz, t_k), it can be execution stages of calculation of dissipative members, Runge-Kutta or limiters. We will assume that the coordinates of the tower are the coordinates of its lower left brick in the xt plane. The parameter $NT \geq 1$ determines how many time steps are calculated with one iteration of the algorithm. In the xt plane each tower is built bottom up on the t axis, and for x in the direction of its slope. At each iteration of the algorithm filling the area in the xy plane goes from right to left, starting from the right border at x, and on each x_i, $i \in 0, \ldots, Nx-1$ towers are built through one along the y axis, while if at $x = x_i$ the towers were built only on even y, then at $x = x_{i-1}$ towers are built only on odd y. We use the fact that the substage calculation algorithms have all the pattern of type "cross", with dependencies on the adjacent six cells.

The development of the fluid solver is made using the CUDA C++ technology. The DiamondTorre algorithm is friendly to CUDA programming tools as the Torres are evaluated by CUDA blocks and each z component inside the Torre is associated to each CUDA thread. Data copying between GPU and CPU is diminished.

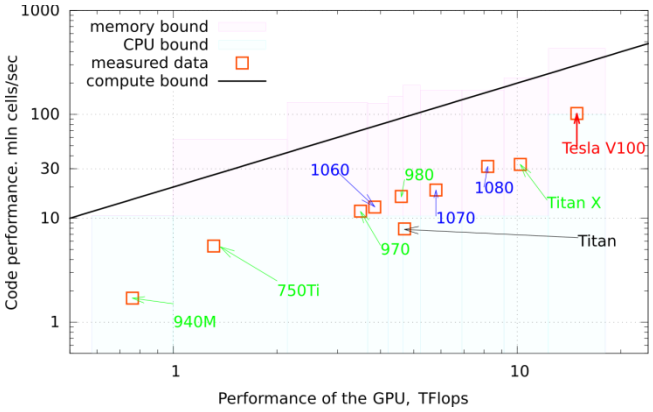
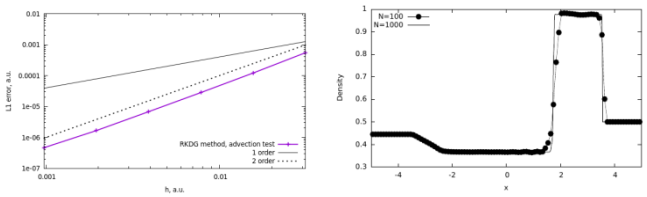


Fig. 1. *The performance test of the developed software based on the DiamondTorre algorithm for different GPUs.*

The performance tests show very good results. Gained speed is more than 10^8 cells per second on top-tier GPU. Full Navier-Stokes solver is considered in the examination. The results of testing on wide range of processors are shown in figure 1.

3. Validation

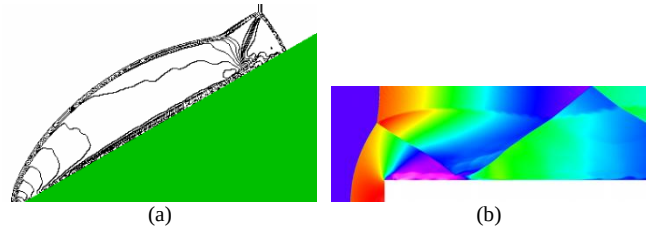
Several validation problems are considered: 1D, 2D and 3D.



(a) (b)

Fig 2. *(a) – the convergence of the solver for the 1D advection test; (b) – the results of density profile for 1D Sod problem.*

In figure 2 the results of 1D testing are shown. Fig. 2a describes the convergence order of the RKDG scheme based upon linear basis and 2-stage Runge-Kutta method investigated using the pure advection setting. In the 2b one can see the results of solving the Riemann problem of Sod [9], showing good shock capturing and low dissipation of the scheme.

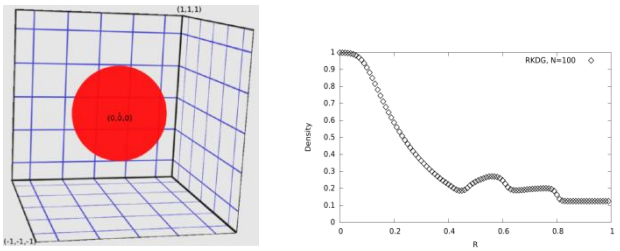


(a) (b)

Fig 3. *(a) – 2D supersonic M=10 flow past a wedge, density levels drawn; (b) – density field for the 2D flow past a step obstacle, M=3.*

In fig. 3 the widely used two-dimensional test problems of the flow past the wedge (fig. 3a) and the flow past the forward-facing step (fig. 3b). The good agreement with the known data is obtained.

Finally, the test of spherical Sod problem is considered [9]. The initial setting at the 3D Cartesian grid and the simulated density profile is shown in fig. 4.



(a) (b)

Fig 4. *(a) – setting for the 3D spherical Sod problem; (b) – density profile.*

Validation results show good quality of obtained solutions, allowing the application of the software for the practical problems.

4. Modeling results and discussion

The actual problems that researchers are usually faced with are complex flows of many components/phases and/or complex geometry. Due to the high performance of considered CFD solver, the "brute force" method of using extra-high grid resolution is possible to apply for these types of problems.

The first considered problem is the process of interaction between the shock wave and the heterogeneity. In fig. 5 the result of simulation of M=2 shock interacting with the high-density heterogeneity of complex initial shape (particularly inspired by letters K and L) is shown.

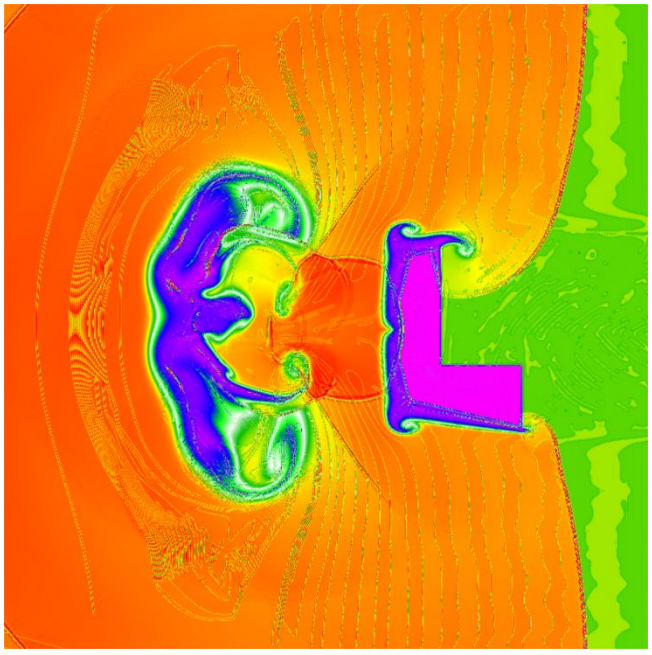


Fig 5. Interaction of a $M = 2$ shock with a high dense heterogeneity $At = 0.9$ of complex shape; density gradient field

From the knowledge of bubble-shock interaction problem, one can get that for the case of $At > 0$ the flow has unstable turbulent pattern [14], as proven by the current simulation.

The second problem concerns the secondary breakup of liquid droplets in the gas flow [15]. The following setup is used: $M = 0.3$ flow is given at initial time, the spherical droplet of incompressible fluid having surface tension force is put in the flow. The ratio between the kinetic energy of the flow and the Laplace energy of droplet (Atwood number At) describes the behavior of the droplet consistency. Experimental and numerical data shows that there are different regimes of the droplet breakup depending on the At number.

For this numerical experiment the current RKDG code for the gas is combined with the special GPU LBM solver for simulating non-compressible liquid. Gas simulation is much more computationally difficult, so the liquid solution is treated as the boundary condition for the gas at every time step. Simulation is performed using GeForce GTX 1080 GPU during several hours. Grid size is 256×128×128 for each simulation.

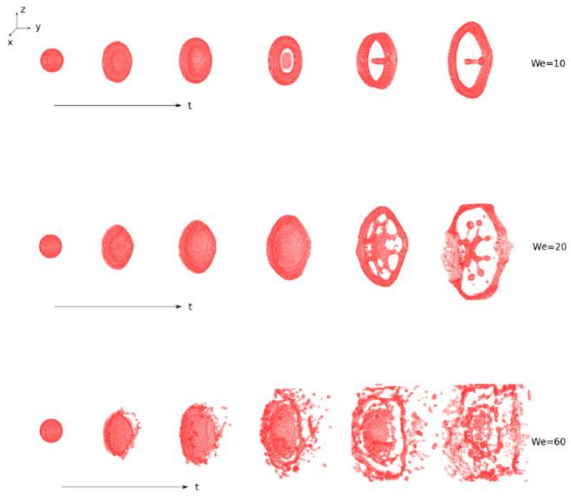


Fig 5. Combined RKDG and LBM simulation of liquid droplet breakup under the gas flow (Mach number M=0.3) at different Weber numbers: 10 (top line), 12 (middle), 60 (bottom). Droplet evolution at the time moments t=0 (left), t=10, ... , t=70 microseconds (right) is shown.

We obtain the qualitative agreement with the known pattern of the droplet breakup [15, 16, 17].

5. Conclusion

The main results of the paper are following. In the framework of the work, a numerical scheme for solving Navier-Stokes equations for a three-dimensional Cartesian grid is created. For the scheme, the GPU algorithm implementation of DiamondTorre is developed. A software package using CUDA C ++ is developed. Positive results of the program validation for the series 1D, 2D and 3D tests are obtained. Application of solver for solving complex engineering research problems of the interaction of the gas inhomogeneity with a shock and the disintegration of droplets in a gas stream is shown.

Acknowledgments

The work is supported by the Russian Science Foundation grant no. 18-71-10004. Also Kintech Lab Company is gratefully acknowledged.

References

1. Flow simulation and high performance computing / T. Tezduyar [et al.] // Computational Mechanics. — 1996. — Vol. 18, no. 6. — Pp. 397—412.

2. Geller T. Supercomputing's exaflop target // Communications of the ACM. — 2011. — Vol. 54, no. 8. — Pp. 16—18.

3. Nickolls J., Dally W. J. The GPU Computing Era // IEEE Micro. — 2010. — Vol. 30, no. 2. — Pp. 56—69.

4. Hagen T. R., Lie K.-A., Natvig J. R. Solving the Euler equations on graphics processing units // International Conference on Computational Science. — Springer. 2006. — Pp. 220—227.

5. Brodtkorb A. R., Hagen T. R., Sætra M. L. Graphics processing unit (GPU) programming strategies and trends in GPU computing // Journal of Parallel and Distributed Computing. — 2013. — Vol. 73, no. 1. — Pp. 4—13. Metaheuristics on GPUs.

6. Batchelor G. K. An introduction to fluid dynamics. – Cambridge university press, 2000.

7. Abgrall R. How to prevent pressure oscillations in multicomponent flow calculations: a quasi conservative approach // Journal of Computational Physics. – 1996. – V. 125. – No. 1. – P. 150-160.

8. Cockburn B., Shu C.-W. The local discontinuous Galerkin method for timedependent convection-diffusion systems // SIAM Journal on Numerical Analysis. — 1998. — Vol. 35, no. 6. — Pp. 2440—2463.

9. Toro E. Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction. — Springer, 2009.

10. Krivodonova L. Limiters for high-order discontinuous Galerkin methods // Journal of Computational Physics. — 2007. — Vol. 226, no. 1. — Pp. 879—896.

11. Moe S. A., Rossmanith J. A., Seal D. C. A simple and effective high-order shock-capturing limiter for discontinuous Galerkin methods // arXiv preprint arXiv:1507.03024. — 2015.

12. Levchenko V. D., Perepelkina A. Y. Locally recursive non-locally asynchronous algorithms for stencil computation // Lobachevskii Journal of Mathematics. – 2018. – V. 39. – No. 4. – P. 552-561.

13. Korneev B. A., Levchenko V. D. Simulating three-dimensional unsteady viscous compressible flow on GPU using the DiamondTorre algorithm // Preprints of the Keldysh Institute of Applied Mathematics. – 2018. – . 105-17.

14. Korneev B., Levchenko V. Numerical simulation of increasing initial perturbations of a bubble in the bubble–shock interaction problem // Fluid Dynamics Research. – 2016. – V. 48. – No. 6. – P. 061412.

15. Hinze J. Fundamentals of the hydrodynamic mechanism of splitting in dispersion processes // AIChE Journal. — 1955. — Vol. 1, no. 3. — Pp. 289—295.

16. Taylor G. The shape and acceleration of a drop in a high speed air stream // The scientific papers of GI Taylor. — 1963. — Vol. 3. — Pp. 457—464.

17. Secondary breakup of a drop at moderate Weber numbers / M. Jain [et al.] // Proc. R. Soc. A. — 2015. — Vol. 471, no. 2177. – – P. 20140930.