DOI: 10.14529/jsfi220408

A Numerical Code for a Wide Range of Compressible Flows on Hybrid Computational Architectures

***Anton A. Shershnev*[1], *Alexey N. Kudryavtsev*[1], *Alexander V. Kashkovsky*[1], *Georgy V. Shoev*[1], *Semyon P. Borisov*[1], *Timofey Yu. Shkredov*[1], *Danila P. Polevshchikov*[1], *Alexey A. Korolev*[1], *Dmitry V. Khotyanovsky*[1], *Yulia V. Kratova*[1]**

The major points in the development of the parallel multiplatform multipurpose numerical code solving the full unsteady Navier–Stokes equations are presented. The developed code is primarily designed for running on multi-GPU computational devices but can also be used on traditional multicore CPUs and even on manycore processors such as Intel Xeon Phi. Physical models include calorically perfect inert gas, single- and multi-temperature approaches for chemically reactive flows and an Euler–Euler model for gas-particle suspensions. Main details of the implementation are described. Shock capturing TVD and WENO schemes in general curvilinear coordinates are used for spatial approximation. Explicit, semi-implicit and fully implicit schemes are employed for advancing solution in time. The code is written in C++ with CUDA API and opensource libraries, such as MPI, zlib and VTK. A few examples of numerical simulations are briefly described to provide general idea of the numerical code capabilities. They include a supersonic flow past a wedge, a jet exhausting from a square nozzle, a heavy gas bubble descending in a lighter medium and a heterogeneous detonation in gas-particle suspension.

Keywords: Navier–Stokes equations, numerical simulation, compressible flows, DNS, thermochemical non-equilibrium, GPGPU, CUDA.

Introduction

Currently numerical simulation has become one of the main tools for conducting scientific research. Fluid dynamics of compressible flows is one of the areas where numerical tools are particularly ubiquitous. However, the compressibility of the flow is associated with the presence of shock waves and other hydrodynamic discontinuities and, therefore, requires rather sophisticated and computationally expensive shock-capturing numerical schemes. Additionally, modern fundamental problems of interest are associated with a wide range of considered scales, including transitional to turbulent and fully turbulent flows, chemically reactive flows, and multiphase flows. And despite the fact of extensive use of parallel computations, such simulations can take hundreds of hours of computational time even on high-performance clusters.

There is a large number of high-performance codes developed for numerical simulations of compressible flows. In particular, in Russia well-known research codes are NERAT [1], Noisette [2], EWT-TsAGI [3], NTS [4], Flowmodellium [5], VP2/3 [6], Jet3D [7], SINF [8] as well as more industry-oriented codes LOGOS [9] and FlowVision [10].

These and many other codes implement different physical models for laminar, turbulent, inert and chemically reactive flows, use different computational meshes and numerical schemes and parallelization techniques. The latter include traditional MPI-based approach, more recent OpenMP application programming interface and even rather exotic OpenCL framework. For any of the numerical tools mentioned above, physical models, numerical techniques, parallelization

[1]Khristianovich Institute of Theoretical and Applied Mechanics, Siberian Branch of Russian Academy of Sciences, Novosibirsk, Russian Federation

methods and programming solutions used in a code reflect the research fields of interest, personal experience and specific needs of its developers and users.

One of the reasonable ways to increase efficiency and significantly reduce computational wall-clock time is to employ modern general-purpose graphics processing units (GPGPU) with high computational efficiency and data throughput (see, e.g., [11–13]). The present paper summarizes the main details of the HyCFS-R numerical code [14–17] developed at the Laboratory of Computational Aerodynamics of Khristianovich Institute of Theoretical and Applied Mechanics and designed to run on hybrid CPU/GPU computational systems. The code solves the full unsteady Navier–Stokes equations using modern shock capturing high-order schemes. In this paper we describe governing equations and discuss some details of the implementation including the problems specific for hybrid and heterogeneous computations.

The rest of the paper is organized as follows: Section 1 describes governing equations and numerical techniques of their solution, in Section 2 the details of the program implementation are given, including general architecture, data structures etc., and in the last section examples of performed numerical simulations are presented.

1. Governing Equations

The code solves the full 3D unsteady Navier–Stokes equations for inert and chemically reactive multispecies mixtures on a structured mesh in general curvilinear coordinates:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}_x}{\partial x} + \frac{\partial \mathbf{F}_y}{\partial y} + \frac{\partial \mathbf{F}_z}{\partial z} = \frac{\partial \mathbf{F}_{v,x}}{\partial x} + \frac{\partial \mathbf{F}_{v,y}}{\partial y} + \frac{\partial \mathbf{F}_{v,z}}{\partial z} + \mathbf{S}, \tag{1}$$

$$\mathbf{Q} = (\rho u, \rho v, \rho w, E, \rho, \rho_1, \ldots, \rho_N,\ \rho_{i_1} E_{v_1}, \ldots, \rho_{i_M} E_{v_M},\ \rho\tilde{\nu},\ \rho_p u_p, \rho_p v_p, \rho_p w_p, E_p, \rho_p)^T, \tag{2}$$

where $\mathbf{Q}$ is the pseudo-vector of conservative variables, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$ are convective fluxes, $\mathbf{F}_v$, $\mathbf{G}_v$, $\mathbf{H}_v$ are viscous fluxes, and $\mathbf{S}$ denotes source terms.

Options for physical models in numerical simulation include:

- A mixture of N calorically perfect gases for laminar and turbulent regimes. Specific heats for each species are constant, i.e., do not depend on the temperature and are calculated in accordance with the number of the molecular degrees of freedom.
- A thermally perfect gas mixture with the single-temperature model for finite-rate chemical kinetics. In this case gas mixture thermodynamic properties are calculated using widely employed approach based on the polynomial approximations with coefficients taken from one of the available sources: the database by Alexander Burcat [18], the NASA database [19] or approximations used by Gupta [20].
- A multi-temperature model with equations for the vibrational energies of molecular species and finite-rate chemical kinetics. In this case, we assume that thermodynamic properties of the translational-rotational mode of the species are given by the relations for the calorically perfect gas, so the specific heats are constant and depend only on the number of degrees of freedom. The rate of translational-vibrational exchange is calculated using either the classical Landau–Teller equation [21], or new model [22], which was obtained using the rigorous methods of the kinetic theory of gases.
- An Euler–Euler model for the chemically reactive gas-particle suspension. The model is based on the concept of interpenetrating continua. The governing equations that describe

the detonation flow in polydisperse gas suspensions of aluminum particles are based on the works of Fedorov and Khmel [23–25].

The pressure of the gas mixture in all cases is calculated using Dalton's law from the partial pressures of components. The transport coefficients of the gas mixture are calculated from those of the individual species using the Wilke mixing rule. Dynamic viscosities of individual species can be calculated using the power law, the Sutherland law or via Ω-integral approximations of the kinetic theory. The thermal conductivities are given by the Eucken formula with Hirschfelder correction [26]. Finally, the option of adding an external field of gravitational force is also available in the code.

2. Numerical Techniques

All spatial approximations in the code are based on the shock capturing TVD (total variation diminishing) and WENO (weighted essentially non-oscillatory) schemes formulated in general curvilinear coordinates. The high-order TVD schemes are implemented using MUSCL (Monotonic Upstream-centered Scheme for Conservation Laws) approach initiated in [28, 29]. More specifically we use the 2nd/3rd order MUSCL formula proposed in [30] along with the **minmod** slope limiter to reconstruct flow variables on cell boundaries from their cell-centered values. Numerical fluxes are calculated using one of the approximate Riemann solvers implemented in the code. The list of available Riemann solvers include HLLE [31] (Harten–Lax–van Leer–Einfeldt), HLLC (Harten–Lax–van Leer–Contact) [32] with an estimate for the contact-wave speed from [33], Roe [34], and a few solvers from the AUSM (Advection Upstream Splitting Method) family: AUSM-Van Leer [35, 36], AUSM-up [39], AUSM+(P) [37], and AUSMPW+ [38]. Alternatively, for the simulation of a calorically perfect gas the WENO scheme of the 5th order by Jiang and Shu [27] with either local or global Lax–Friedrichs flux splitting can be used. Convective terms for the disperse phase continuum are calculated using the MUSCL reconstruction combined with the dusty gas Riemann solver from [40]. Diffusive terms are calculated using the central differences of the 2nd order.

The solution is advanced in time using explicit, semi-implicit or implicit schemes. Explicit time integration is based on the so-called Runge–Kutta TVD schemes of the 1st through the 3rd order [41], or the low-storage Runge–Kutta–Gill scheme of the 4th order [42]. In the additive semi-implicit Runge–Kutta scheme of the 2nd order (ASIRK2C) [43] stiff chemical source terms are evaluated implicitly while for the convective terms more efficient and simpler explicit methods are used. The implicit DPLUR (Data Parallel Lower-Upper Relaxation) scheme [44] available for non-reacting flows allows one to integrate equations in fully implicit manner in parallel computations.

2.1. Boundary Conditions

The imposition of boundary conditions in the HyCFS-R code is implemented using the so-called ghost cells technique. The computational domain is surrounded by three rows of virtual cells, in which flowfield variables are calculated according to the type of boundary condition. Generally speaking, the number of the ghost-cell rows depends on the size of the stencil, but HyCFS-R always uses three layers, corresponding to the 7-point stencil in high-order schemes.

The following boundary types are implemented in HyCFS-R: supersonic inflow and outflow, pressure-constant inlet and outlet, periodic boundaries, and solid boundaries, such as inviscid,

no-slip isothermal and no-slip adiabatic walls. Additionally, special boundary conditions for superposing disturbances onto the flow are implemented: in the form of eigenfunctions of the linear stability problem, white gaussian noise and periodic thermal fluctuations. The last three conditions are typically used for flow forcing when simulating transition to turbulence in shear flows.

It should be mentioned, that the code employs specific scheme of boundary conditions imposition. In most papers this stage of numerical algorithm is described very briefly and it is usually implied that values in ghost cells are recalculated at the beginning of the new time step based on the values in the internal cells. In HyCFS-R boundary conditions for inviscid and viscous numerical fluxes are imposed separately, to ensure conservation of mass at the solid boundaries.

3. Details of the Program Implementation

The program is written in C++ and uses only open-source tools, libraries and technologies and in-house utilities, such as GNU/Linux, GCC, OpenMP, MPI, zlib, VTK, to avoid vendor-lock situations often encountered when using commercial solutions. The primary computational platform of the code are hybrid CPU/GPU clusters and all numerical procedures in the code are implemented with SIMD (single instruction multiple data) execution in mind. It should be noted that computations of numerical fluxes and source terms in the Navier–Stokes equations mostly consist of uniform-length loops with fixed-size stencils and small number of conditional operators. Such computations match the aforementioned SIMD architecture quite well and straightforward implementation of numerical routines provides sufficient efficiency.

For the maximum flexibility of the code it can be compiled to run on conventional multicore CPUs using a special compile-time wrapper for the C preprocessor, which converts GPU-specific entities to OpenMP syntax. In this approach each OpenMP thread is interpreted as a GPU thread. Just as GPU kernel function is executed by each CUDA thread, the converted CPU version of the kernel is executed on each OpenMP thread. The only difference is that OpenMP threads are not forced on a hardware level to be executed synchronously, i.e., in a SIMD manner. However, it is not important in our computations. This technique allows one to significantly increase the number of supported computational architectures and also simplifies debugging of the code when using conventional C++ development tools, such as **GDB debugger, Valgrind** framework for code dynamic analysis and so on.

3.1. Input and Output Files

Internal formats are used for all input data: mesh, flowfields, chemical mechanisms description and general configuration options. Mesh and flowfields are stored in simple ASCII or binary files, compressed with zlib library to reduce storage space. General configuration is processed using a special configuration reader, also developed at the Laboratory of Computational Aerodynamics at ITAM SB RAS. The reader is based on the "key-value" scheme. The parameters are divided into groups corresponding to its type and/or parent data structure. The reader also supports `#include` directives, allowing to directly insert contents of one file into another, like in a typical C/C++ code, for additional flexibility of the configuration. The reader stores both the list of all keys supported by the solver and the list of the keys actually read from the input files and checks if the parameter values fall out of the required range. Preprocessing tools for

importing files from third-party programs are also included in the code, e.g., tools for reading the description of chemical reaction mechanisms in CHEMKIN format, for the mesh in FlowLogic format and some others.

Postprocessing utilities implemented in the code allow one to export flowfields in ASCII and binary Tecplot format, and legacy ASCII VTK formats, calculate integrated and mean values of any variable, extract distributions from arbitrary line segment, and calculate integrated and distributed aerodynamic characteristics on the solid walls.

Most aerodynamic characteristics are based on the following normalized and integrated quantities at the solid wall surface: the pressure p_w, the frictional stress $\tau_w = \mu_w(\partial u_s/\partial n)$ and the heat flux $q_w = \kappa_w(\partial T/\partial n)$. Here $\mathbf{n}$ and $\mathbf{s}$ are the directions normal and tangent to the wall, respectively. Normal and tangent components of velocity are calculated from Cartesian velocity components stored in the flowfields. Then derivatives along the normal and tangent directions are transformed to the derivatives in general curvilinear coordinates. The quantities, such as velocity $\mathbf{u}$, dynamic viscosity μ, heat conductivity κ and Jacobian of transformation $\mathcal{J}$ between Cartesian and curvilinear coordinates are extrapolated to the wall with the 1st or 2nd order of accuracy. After calculation of the distributed characteristics the integrated characteristics are obtained by summation of local values multiplied by the surface element areas.

3.2. Data Structures

3.2.1. cContainer class

Because the HyCFS-R code contains various physical models with different number of equations, which can be switched at runtime, the data structures used for flowfields storage should also have an adjustable list of variables and means to access these variables in a convenient way. Conventional containers from C++ STL (Standard Template Library) cannot be used on GPU devices, so one has to implement custom container for the flowfields. The HyCFS-R uses a class **cContainer** with a simple array of floating point values at its core, extended with a few extra features, such as a "CPU/GPU" flag (indicating in which memory the data is allocated), a list of variables names, functions for accessing values by the number of cell and the name or the number variable, possibility to store in a cell not only a list, but also a full matrix of parameters. The latter is useful when dealing with various Jacobians and other matrices used in implicit methods.

3.2.2. Mem class

When developing large complex codes for the numerical simulation, the programmers have to change the list of parameters of various functions on a regular basis. Additional parameters are introduced when programming new numerical methods, physical models or new implementations of existing functions. Naturally, any change in the list will require revisions in every occurrence of the function call. To avoid or at least reduce the amount of this tedious work, in the HyCFS-R we store all variables and pointers to arrays in a single structure **Mem**, instances of which are stored in both the CPU and GPU memories. We avoid the word 'copies', because for each Mem class member we specify explicitly in which memory (CPU or GPU) it will be allocated. It allows us to minimize the array duplication.

A pointer to **Mem** structure passed to the functions as an argument provides the access to all data available in a computation. One of the advantages of this approach is that most of

the memory allocation routines are limited to methods of a single class, allowing to simplify the control of memory consumption and prevention of memory leaks and corruption.

When using domain decomposition, each partition has its own dedicated pair of CPU and GPU instance of Mem class.

3.2.3. BSegment and condition classes

When imposing boundary conditions, the boundary of the domain is divided into partitions we call *segments*. They are rectangular in the $\{\xi, \eta, \zeta\}$-space and, generally speaking, can overlap. Geometrical parameters of each segment and the type of boundary conditions assigned to this segment are stored as object of the dedicated class **BSegment**. Physical parameters of boundary conditions are stored as separate **Condition** objects, referenced by one of the segments. The conditions contain wall temperature, free-stream flow parameters and so on. This mechanism of splitting of geometrical and physical parameters of a boundary condition allows one to re-use the segments for organization of inter-block exchange in a multiblock structured mesh. For this case the segments also store a multiblock interface identificator and exchange buffers of an appropriate size. Figure 1 shows a schematical example of domain boundary divided into segments, including a multiblock partition.

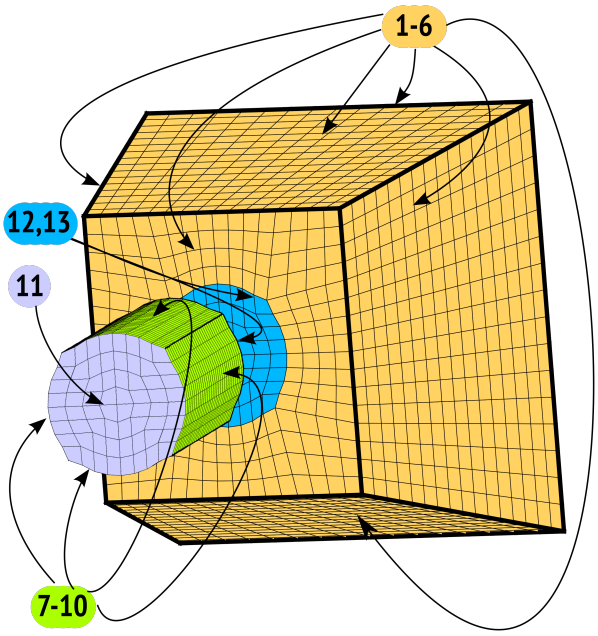


Figure 1. Schematic of boundary segments

3.2.4. Element and reaction classes

The physical and chemical properties of the species and descriptions of chemical reactions are stored in the dedicated structures. And since CUDA platform puts certain limitations on C++ features, inheritance is used mostly to separate subsets of parameters describing gas and condensed phases. So, the hierarchy of the classes is as follows: the classes **Gas** and **Condensed** are both inherited from the basic class **BaseElement** and the classes **GasReaction** and **HeterogenousReaction** are both inherited from the class **BaseReaction**.

3.3. Parallelization

In the most general case the HyCFS-R uses a multi-level parallelization based on the CUDA API, OpenMP threads and the MPI protocol. There are 2 variations of the numerical algorithm: one is used for domains with simple geometries and regular meshes and the other is used for complex domains with structured multiblock meshes. In both cases the whole computational domain is divided into a number of partitions, corresponding to the number of used GPUs. Each GPU performs computations in cells of its sub-domain and after each step exchanges data with neighboring sub-domains. GPUs on one computational node are controlled using OpenMP threads and exchanges are made via the CPU memory. Inter-nodal exchanges are performed using the MPI protocol.

The difference is that in the simple variant the domain is partitioned along the k-index (ζ-axis) as shown in Fig. 2. As a result, neighboring previous and subsequent partitions are unambiguously defined from the current partition index. Another benefit of this scheme is that cells with the same index on the ζ-axis are in a continuous memory segment and do not require extra buffers for exchange.

As for the multiblock exchange, it requires not only the description of every block-block connection via the segment mechanism, but also a special configuration file assigning each block to a GPU device and to an MPI node.

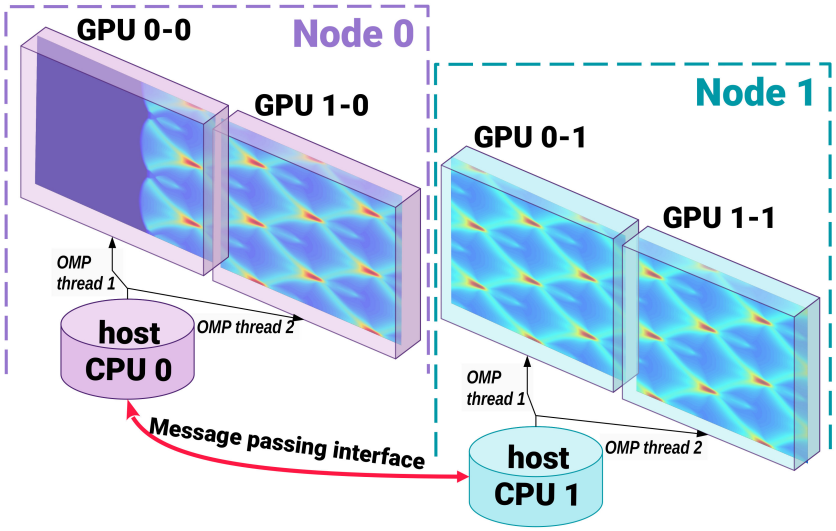


Figure 2. Scheme of multi-level parallelization

The multilevel approach described above allows one a hybrid parallelization when both GPUs and multicore CPUs are used as computational units. Hybridization is achieved by employing OpenMP for multicore CPUs in the same way as CUDA threads are used on GPUs. The source code is the same on all platforms, which, combined with a proper compile-time wrapper, ensures that all computations are performed in the same way. Execution and data exchange are done using the MPI library. Figure 3 shows an example of devices configuration in a hybrid computation. In this example the domain is divided into 8 partitions, represented by gold and green blocks at the bottom of the scheme, where the gold partitions are computed on GPU, and green are computed on CPU, respectively. The configuration consists of 2 nodes with single 10-core CPU and 2 GPUs on each node. The MPI processes with ranks 0 and 1 control the GPUs and the rest of processes control CPU computations with OpenMP threads. This example illustrates

a typical situation when using hybrid CPU/GPU clusters, where the number of CPU cores is significantly higher than the number of GPU devices.

In practice such hybrid computations are rarely performed because most of the popular job schedulers lack features to manage resources for them. And it also should be noted that complex computations require to balance the load of a computational devices, e.g., by matching the size of each partition to the performance of the device.

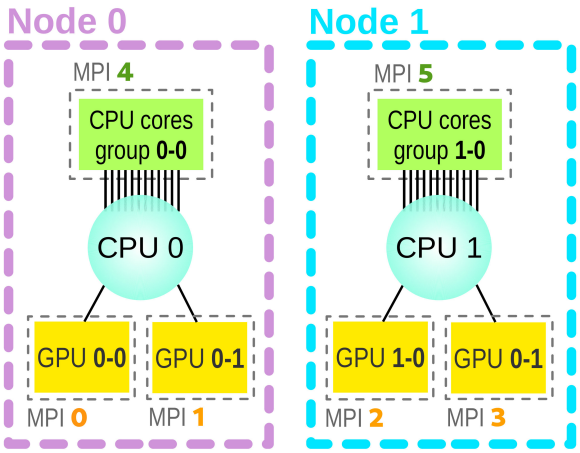


Figure 3. Example of hybrid CPU-GPU parallelization scheme. 6 MPI, 8 domains: 4 GPU + 4×4 OpenMP(CPU)

3.3.1. Service and debugging options

Development of any CFD program is associated with thorough testing for correctness, accuracy and performance of the code. For these purposes, all calls of computational routines are surrounded by wall-time measuring functions and followed by special inspecting routines, which allow one to check all affected flowfields for any changes.

The timers are implemented in a simple class based on the standard **gettimeofday** function and **std::map<string,double>** container. They provide simple statistics for the execution time between two given lines of code and can be switched off at runtime using the configuration file. This statistics is sufficient for initial assessment of hot spots and bottlenecks in the code. A more detailed analysis can be done using external tools such as **valgrind/kcachegrind**, **gperftools**, **Intel VTune** and so on.

As for the inspection, it is enabled at compile time and is only used in the debug-build of the code. This build uses not only inspection routines but also specific build flags for the detection of floating point exceptions, enabling faster search for the problematic fragments.

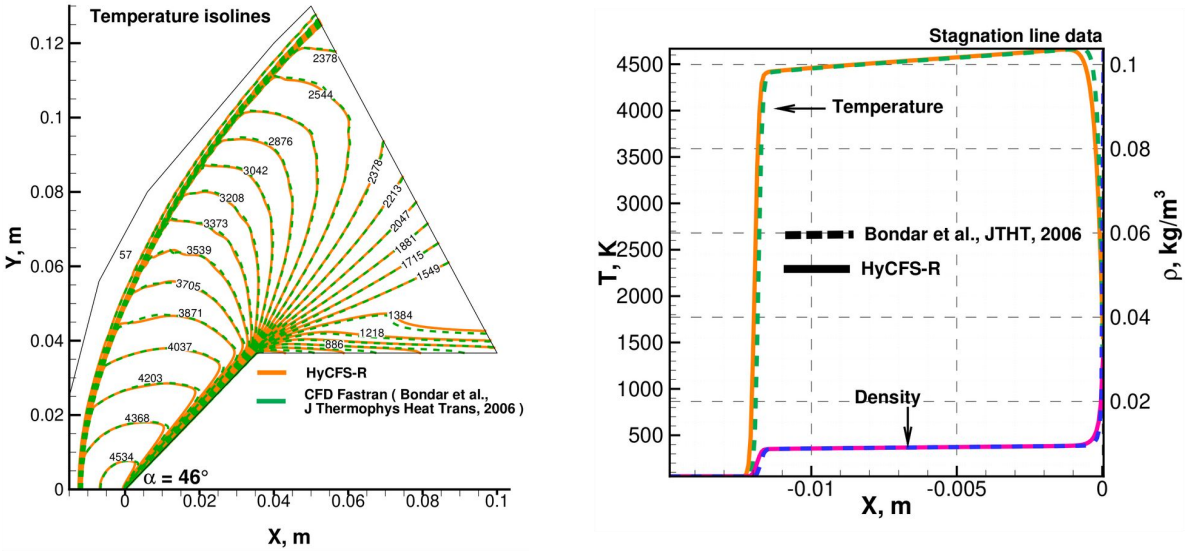
4. Some Examples of Numerical Simulations

This section contains a brief description of a few numerical studies performed using the developed numerical code, which illustrate some typical problems of the compressible flow gas dynamics and the capabilities of the code.

4.1. Supersonic Flow Past a Wedge

A supersonic flow of pure argon past a 46° wedge for the conditions of experiment by Hornung and Smith [47] was simulated numerically and compared with the data from [45]. The mesh size is identical to that used for Navier–Stokes computations in [45], namely 360×200 cells.

Results are shown in Fig. 4, where the comparison of temperature and density distributions are depicted. As can be seen, the HyCFS-R solution is in excellent agreement with the data from [45] in terms of the shock wave stand-off distance. This test verifies spatial approximation and boundary conditions at the solid walls.



(a) Temperature isolines

(b) Temperature and density distributions along the stagnation line

Figure 4. Supersonic argon flow past a 46 degree wedge

4.2. Jet Exhausting from a Square Nozzle

A 3D numerical simulation of the development of instabilities was carried out for a supersonic perfectly-expanded jet exhausting from a nozzle of the square cross-section into an ambient stream. The simulations were performed for the jet Mach number $M_j = 2.5$ and the co-flow Mach number $M_a = 1.5$ at the Reynolds number $Re = 5000$. The computational domain had the hexahedral shape. The computational grid was refined around the jet core and its near field. The computations were carried out on the grid $1152 \times 330 \times 330 \approx 125$ million cells. The jet instability was excited by superimposing disturbances in the form of a white Gaussian noise onto the main flow velocity components.

Figure 5 shows isosurfaces and contours of density in a central cross-section. As can be clearly seen, at the initial stage, near the inflow boundary, the instability develops in the form of a typical chain of the Kelvin–Helmholtz vortices. This development of instability is quite expected for the considered values of the ambient and jet flow Mach numbers. Downstream, the vortex motion starts to dominate the entire jet core. Overall, the results of these numerical simulations allowed us to identify specific features of the instability development in a supersonic rectangular jet with a supersonic co-flow.

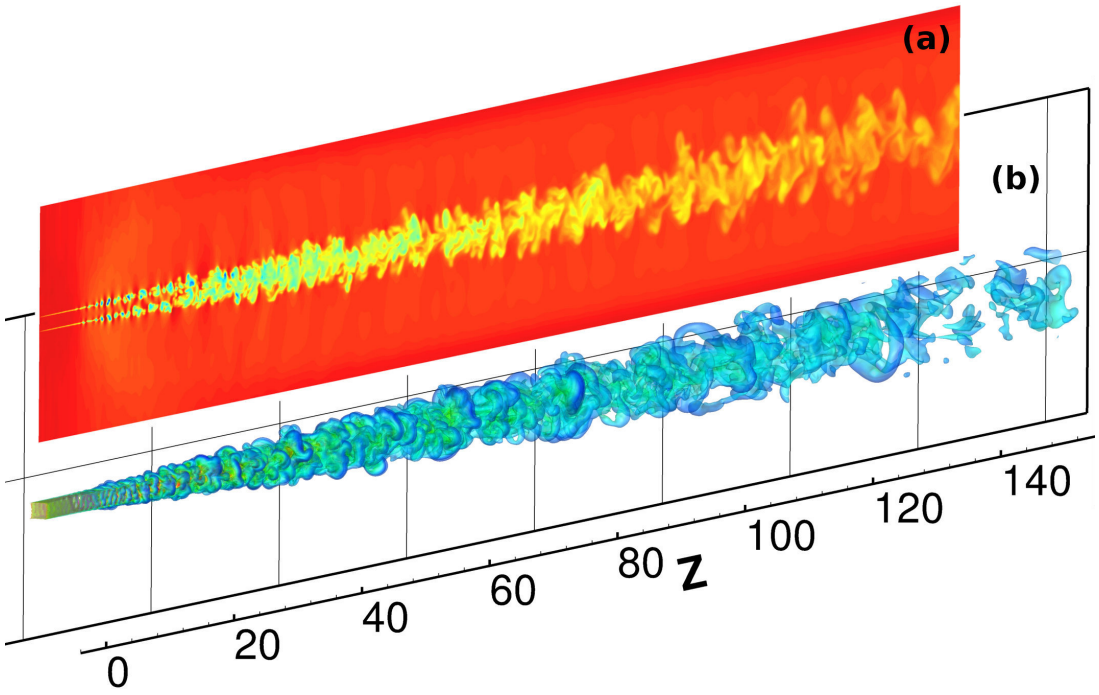


Figure 5. Rectangular jet with Mach number $M_j = 2.5$ exhausting into ambient coflow with Mach number $M_a = 1.5$. Density contours in middle cross-section (a) and density isolines colored with local Mach number values (b)

4.3. Instability of a Gas Bubble in a Gravitational Field

Numerical simulations of the instability of cylindrical and spherical volumes of argon surrounded by helium in the field of gravity were carried out. The volume diameter was set equal to 20 m, the initial temperature in all domain T = 300K. The gravitational acceleration was equal to $g = 1000$ m/s^2. The barotropic distribution of density and pressure was set both inside the volume and in the surrounding light gas. The pressure at the bottom wall was about 1 Pa.

Three cases were considered and 2D, axisymmetric, and 3D computations were carried out for the following domain and mesh sizes. The 100×200 m domain with 800×1600 grid was used for two-dimensional computation, the domain was halved along the axis of symmetry in axisymmetric computation down to 50×200 m with mesh containing 400×1600 cells. In 3D simulation 100×100×200 m domain with 280×280×560 ≈ 43.9 million cells was used.

The results of various computations were compared with each other (see Fig. 6). The volume in all simulations changed shape from round in cross section becoming close to semicircular, and then transforming into a crescent-like structure, with cusps pointing upwards. These cusps begin to twist, a secondary instability develops on them. At first the cusps and then entire bubble start to break down, mixing with the ambient gas. In all geometries the instability emergence and development as well as the breakdown process have their own characteristics. The fastest mixing and the breakdown of the heavy gas bubble was observed in the 3D simulation while the mixing process in the 2D simulation was the slowest, because of its cylindrical shape.

4.4. Heterogeneous Detonation in Al/O_2 Gas-particle Suspension

In this test case a detonation wave propagating through a channel filled with oxygen and 10 μm aluminum particles suspension is investigated numerically. Gas phase parameters corre-

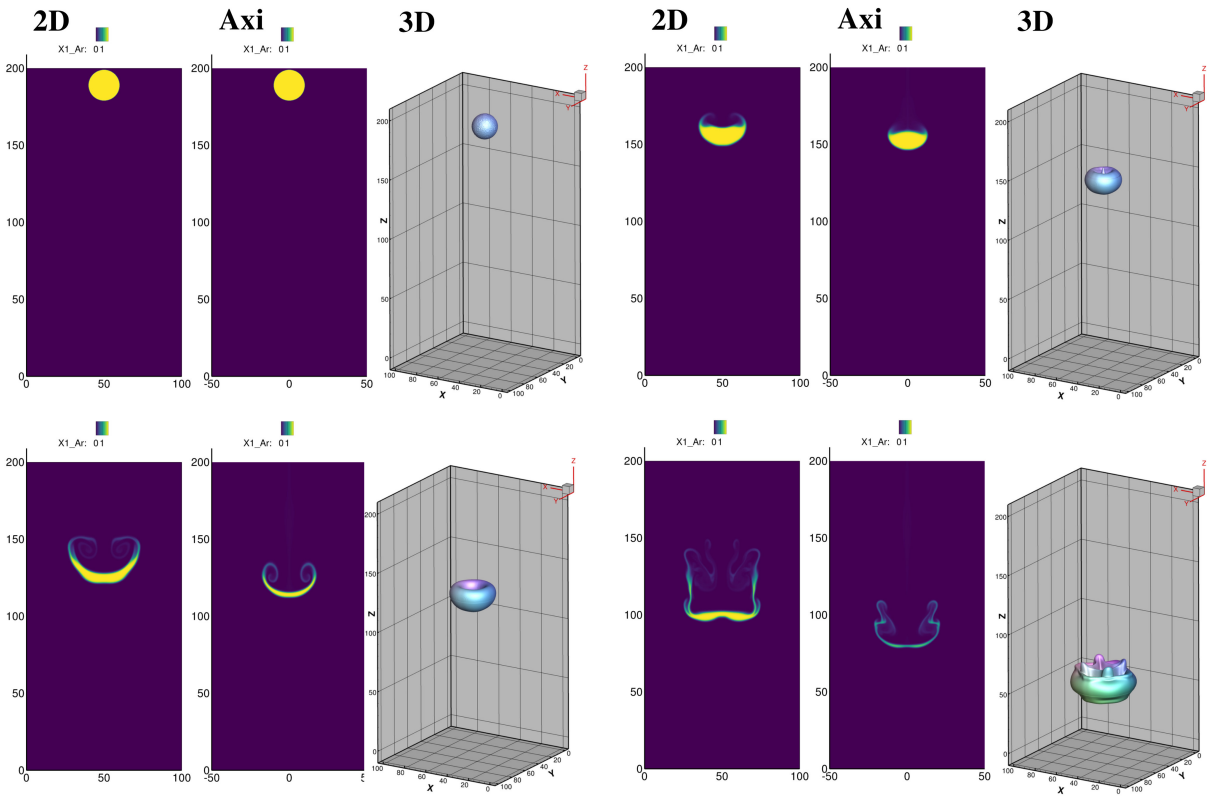
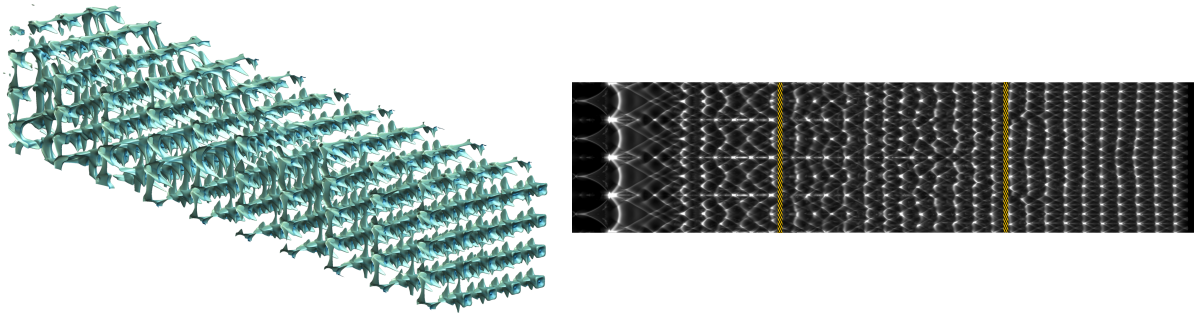


Figure 6. Instability of an argon bubble in helium. Argon molar fraction contours in 2D, axisymmetric and 3D NS computations shown at different time instants

spond to the standard atmospheric conditions with the pressure of 101325 Pa and the density of 1.28 kg/m^3. Detonation is initiated by a localized ignition at the left boundary of the computational domain. The 22.5 m $\times$ 1 m $\times$ 1 m channel was simulated as 3 separate 7.5 m long sections each with the spatial resolution of $2400 \times 350 \times 350 \approx 300$ million of cells. The computations were carried out using 8 Nvidia Tesla V100 GPUs and took 96 hours of wall-time.

Figure 7b shows the history of maximum pressure in the form of isosurfaces and distributions on the channel lateral sides, imitating the soot-covered foils from the real experiment. The latter also illustrate the process of transformation of the initial blast wave into a regular cellular structure. Additional code features were used to measure the speed of detonation wave front and also capture its shape.



(a) Isosurface in the flowfield of history of maximum pressure

(b) Numerical "soot-foil" records on the channel walls

Figure 7. Heterogeneous detonation in a rectangular channel filled with Al/O_2 gas-particle suspension

Conclusion

The HyCFS-R numerical code for solving the compressible Navier–Stokes equations on parallel hybrid computational systems equipped with multiple GPUs or manycore co-processors has been developed. The developed code can be used to simulate gaseous flows including flows of thermally non-equilibrium and chemically reacting gases and gas mixtures as well as multiphase gas-particle suspensions. Modern shock-capturing TVD and WENO schemes on structured muiltiblock meshes are used for spatial approximation, while the time stepping is performed with explicit RK TVD, semi-implicit ASIRK2c and implicit DPLUR schemes. The descriptions of flowfields, boundaries and boundary conditions, physical and thermodynamic properties of the gases, chemical kinetics, are stored as simple C++ classes and structures compatible with CUDA API. Multilevel parallelization of the HyCFS-R code is achieved via a set of techniques, namely MPI, OpenMP and CUDA technologies. Third-party software tools such as **GNU Data Debugger**, **valgrind/kcachegrind**, and **Intel VTune** are used for debugging and code optimization, **zlib** library is used for flowfields data compression and **VTK** library is used for data visualization.

The HyCFS-R code has been employed to simulate a number of subsonic and supersonic flows: a flow past a wedge, a jet exhausting from a square nozzle, a bubble of heavy gas descending in a lighter medium, a heterogeneous detonation in a rectangular channel. Numerical grids containing up to 300 million of cells has been used. The results of numerical simulations presented in the paper give evidence that the developed code is capable to efficiently simulate a wide range of compressible flows.

Acknowledgements

This work was supported by the Russian Science Foundation, grant No. 18-08-01442-П.

References

1. Surzhikov, S.T.: Radiation aerothermodynamics of the Stardust space vehicle. Journal of Applied Mathematics and Mechanics 80(1), 44–56 (2016). https://doi.org/10.1016/j.jappmathmech.2016.05.008

2. Abalakin, I.V., Bakhvalov, P.A., Gorobets, A.V., *et al.*: Parallel software package NOISETTE for large-scale computations in fluid dynamics and aeroacoustics. Num. Meth. Prog. 13(3), 110–125 (2012).

3. Neyland, V.Y., Bosnyakov, S.M., Glazkov, S.A., *et al.*: Conception of electronic wind tunnel and first results of its implementation. Progress in Aerospace Sciences 37(2), 121–145 (2001). https://doi.org/10.1016/S0376-0421(00)00013-0

4. Shur, M., Strelets, M., Travin, A.: High-order implicit multi-block Navier–Stokes code: Ten years experience of application to RANS/DES/LES/DNS of turbulent flows. https://cfd.spbstu.ru/agarbaruk/doc/NTS_code.pdf

5. Petrov, M.N., Tambova, A.A., Titarev, V.A., *et al.*: FlowModellium Software Package for Calculating High-Speed Flows of Compressible Fluid. Comput. Math. and Math. Phys. 58, 1865–1886 (2018). https://doi.org/10.1134/S0965542518110118

6. Isaev, S.A., Baranov, P.A., Usachov, A.E.: Multiblock computational technologies in the VP2/3 Package on aerothermodynamics. LAP LAMBERT Academic Publishing, Saarbrucken (2013).

7. Lebedev, A.B., Lyubimov, D.A., Maslov, V.P., *et al.*: The prediction of three-dimensional jet flows for noise applications. AIAA Paper no. 2002-2422 (2002). https://doi.org/10.2514/6.2002-2422

8. Smirnov, E.M., Zajtsev, D.K.: The finite-volume method in application to complex-geometry fluid dynamics and heat transfer problems. Scientific-Technical Bulletin of the St.-Petersburg State Technical University 2(36), 70–81 (2004). (in Russian)

9. Kozelkov, A.S., *et al.*: Multifunctional LOGOS software package for computing fluid dynamics and heat and mass transfer using multiprocessor computers: basic technologies and algorithms. Supercomputing and mathematical modeling: Proceedings of the XII International Workshop, Russia, Sarov, pp. 215–230 (2010). (in Russian)

10. Aksenov, A.A.: Flowvision: Industrial computational fluid dynamics. Computer Research and Modeling 9(1), 5–20 (2017). (in Russian) https://doi.org/10.20537/2076-7633-2017-9-5-20

11. Yao, Y., Yeo, K.-S.: An application of GPU acceleration in CFD simulation for insect flight. Supercomputing Frontiers and Innovations 4(2), 13–26 (2017). https://doi.org/10.14529/jsfi170202

12. Chaplygin, A.V., Gusev, A.V., Diansky, N.A.: High-performance shallow water model for use on massively parallel and heterogeneous computing systems. Supercomputing Frontiers and Innovations 8(4), 74–93 (2021). https://doi.org/10.14529/jsfi210407

13. Gorobets, A.V., Duben, A.P.: Technology for supercomputer simulation of turbulent flows in the good new days of exascale computing. Supercomputing Frontiers and Innovations 8(4), 4–10 (2022). https://doi.org/10.14529/jsfi210401

14. Shershnev, A.A., Kudryavtsev, A.N., Kashkovsky, A.V., Khotyanovsky, D.V.: HyCFS, a high-resolution shock capturing code for numerical simulation on hybrid computational clusters. AIP Conf. Proc. 1770, 030076 (2016). https://doi.org/10.1063/1.4964018

15. Kudryavtsev, A.N., Kashkovsky, A.V., Borisov, S.P., Shershnev, A.A.: A numerical code for the simulation of non-equilibrium chemically reacting flows on hybrid CPU-GPU clusters. AIP Conf. Proc. 1893, 030054 (2017). https://doi.org/10.1063/1.5007512

16. Borisov, S.P., Kudryavtsev, A.N., Shershnev, A.A.: Development and validation of the hybrid code for numerical simulation of detonations. J. Phys.: Conf. Ser. 1105, 012037 (2018). https://doi.org/10.1088/1742-6596/1105/1/012037

17. Borisov, S.P., Kudryavtsev, A.N., Shershnev, A.A.: Influence of detailed mechanisms of chemical kinetics on propagation and stability of detonation wave in H_2/O_2 mixture. J. Phys.: Conf. Ser. 1382, 012052 (2019). https://doi.org/10.1088/1742-6596/1382/1/012052

18. Alexander Burcat's Ideal Gas Thermodynamic Data in Polynomial form for Combustion and Air Pollution Use. https://garfield.chem.elte.hu/Burcat/burcat.html

19. McBride, B.J., Zehe, M.J., Gordon, S.: NASA Glenn coefficients for calculating thermodynamic properties of individual species. NASA/TP-2002-211556 (2002).

20. Gupta, R.N.: Viscous shock-layer study of thermochemical nonequilibrium. Journal of Thermophysics and Heat Transfer 10(2), 257–266 (1996). https://doi.org/10.2514/3.801

21. Landau, L., Teller, E.: Theory of sound dispersion. Phys. Z. Sowjetunion 10(1), 34–43 (1936).

22. Kustova, E., Oblapenko, G.: Reaction and internal energy relaxation rates in viscous thermochemically non-equilibrium gas flows. Phys. Fluids 27(1), 016102 (2015). https://doi.org/10.1063/1.4906317

23. Fedorov, A.V., Khmel, T.A. Fomin, V.M.: Non-equilibrium model of steady detonations in aluminum particles-oxygen suspensions. Shock Waves 9, 313–318 (1999). https://doi.org/10.1007/s001930050191

24. Fedorov, A.V., Khmel, T.A.: Numerical simulation of formation of cellular heterogeneous detonation of aluminum particles in oxygen. Combust. Expl. Shock Waves 41(4), 435–448 (2005). https://doi.org/10.1007/s10573-005-0054-7

25. Fedorov, A.V., Khmel, T.A.: Formation and degeneration of cellular detonation in bidisperse gas suspensions of aluminum particles. Combust. Expl. Shock Waves 44(3), 343–353 (2008). https://doi.org/10.1007/s10573-008-0042-9

26. Hirschfelder, J.O., Curtiss, C.F., Bird, R.B.: Molecular Theory of Gases and Liquids. Wiley, New York (1954).

27. Jiang, G.-S., Shu, C.-W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. 126, 202–228 (1996). https://doi.org/10.1006/jcph.1996.0130

28. Kolgan, V.P.: Application of the principle of minimal values of the derivative to construction of mesh schemes to calculation of discontinuous solutions of gas dynamics. Uch. Zap. TsAGI 3(6), 68–77 (1972). (in Russian). Translated to English and reprinted in J. Comput. Phys. 230, 2384–2390 (2011). 10.1016/j.jcp.2010.12.033

29. van Leer, B.: Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. J. Comput. Phys. 32(1), 101–136 (1979). https://doi.org/10.1016/0021-9991(79)90145-1

30. Anderson, W.K., Thomas, J.L., van Leer, B.: Comparison of finite volume flux vector splittings for the Euler equations. AIAA Journal 24(9), 1453–1460 (1986). https://doi.org/10.2514/3.9465

31. Einfeldt, B., Munz, C.D., Roe, P.L., Sjögren, B.: On Godunov-type methods near low densities. J. Comput. Phys. 92(2), 273–295 (1991). https://doi.org/10.1016/0021-9991(91)90211-3

32. Toro, E.F., Spruce, M., Speares, W.: Restoration of the contact surface in the Harten–Lax–van Leer Riemann solver. Shock Waves 4, 25–34 (1994). https://doi.org/10.1007/BF01414629

33. Batten, P., Leschziner, M.A., Goldberg, U.C.: Average-state Jacobians and implicit methods for compressible viscous and turbulent flows. J. Comput. Phys. 137, 38–78 (1997). https://doi.org/10.1006/jcph.1997.5793

34. Roe, P.L.: Approximate Riemann solvers, parameter vectors, and difference schemes. J. Comput. Phys. 43(2), 357–372 (1981). https://doi.org/10.1016/0021-9991(81)90128-5

35. van Leer, B.: Flux-vector splitting for the Euler equations. In: Krause, E. (eds) Eighth International Conference on Numerical Methods in Fluid Dynamics. Lecture Notes in Physics, vol. 170, pp. 507–512. Springer, Berlin, Heidelberg (1982). https://doi.org/10.1007/3-540-11948-5_66

36. Liou, M.-S., Steffen, C.J.: A new flux splitting scheme. J. Comput. Phys. 107(1), 23–39 (1993). https://doi.org/10.1006/jcph.1993.1122.

37. Edwards, J.R., Liou, M.-S.: Low-diffusion flux-splitting methods for flows at all speeds. AIAA J. 36, 1610–1617 (1998). https://doi.org/10.2514/2.587

38. Kim, K.H., Kim, C., Rho, O.-H.: Methods for the accurate computations of hypersonic flows – I. AUSMPW+ scheme. J. Comput. Phys. 174(1), 38–80 (2001). https://doi.org/10.1006/jcph.2001.6873

39. Liou, M.-S.: A sequel to AUSM, Part II: AUSM$^+$-up for all speeds. J. Comput. Phys. 214(1), 137–170 (2006). https://doi.org/10.1016/j.jcp.2005.09.020

40. Collins, J.P., Ferguson, R.E., Chien, K., *et al.*: Simulation of shock-induced dusty gas flows using various models. AIAA paper 94-2309.

41. Shu, C.-W., Osher, S.: Efficient implementation of essentially non-oscillatory shock-capturing schemes. J. Comput. Phys. 77(2), 439–471 (1988). https://doi.org/10.1016/0021-9991(88)90177-5

42. Thompson, R.J.: Improving round-off in Runge–Kutta computations with Gill's method. Commun. ACM 13(12), 739–740 (1970). https://doi.org/10.1145/362814.362823

43. Zhong, X.: Additive semi-implicit Runge–Kutta methods for computing high-speed nonequilibrium reactive flows. J. Comput. Phys. 128(1), 19–31 (1996). https://doi.org/10.1006/jcph.1996.0193

44. Wright, M.J., Candler, G.V., McDonald, J.D.: Data-parallel lower-upper relaxation method for reacting flows. AIAA Journal 32(12), 2380–2386 (1994). https://doi.org/10.2514/3.12303

45. Bondar, Ye.A., Markelov, G.N., Gimelshein, S.F., Ivanov, M.S.: Numerical modeling of near-continuum flow over a wedge with real gas effects. Journal of Thermophysics and Heat Transfer 20(4), 699–709 (2006). https://doi.org/10.2514/1.18758

46. Shuen, J.-S.: Upwind differencing and LU factorization for chemical non-equilibrium Navier–Stokes equations. J. Comput. Phys. 99(2), 233–250 (1992). https://doi.org/10.1016/0021-9991(92)90205-D

47. Hornung, H.G., Smith, G.H.: The influence of relaxation on shock detachment. J. Fluid Mech. 93(2), 225–239 (1979). https://doi.org/10.1017/S0022112079001865