Development and validation of the hybrid code for numerical simulation of detonations

S P Borisov¹, A N Kudryavtsev¹,², A A Shershnev¹

¹ Khristianovich Institute of Theoretical and Applied Mechanics SB RAS, 630090, Novosibirsk, Russia
² Novosibirsk State University, 630090, Novosibirsk, Russia

E-mail: bsp5@yandex.ru

Abstract. The present paper describes a numerical code for simulation of detonation flows on hybrid computational clusters (CPU/GPU). The code can solve 1D, 2D and 3D unsteady Euler equations for multispecies chemically reacting flows using high-order shock-capturing TVD schemes and a finite-rate chemistry solver. The implementation is based on the OpenMP, MPI and CUDA technologies allowing for both flexibility and computational efficiency of the code. The program is verified on the analytical Zeldovich-von Neumann-Doering solution for the 1D detonation wave in H₂/O₂ mixture. Some examples of 2D computations are also given.

1. Introduction
Modern studies show that in order to properly simulate a detonation wave (DW) structure detailed chemical mechanisms have to be used [1]. It is also an accepted fact that there should be at least 100 grid points across a detonation cell to correctly simulate the cellular structure and resolve the detonation cells. This results in large computational grids with tens and hundreds of millions of points and prohibitively long wall-clock computational times on modern supercomputers with multiple CPU cores. One of the proven ways to overcome this problem is to employ hybrid computational clusters. In this paper we describe a numerical code for hybrid GPU/CPU supercomputers, provide the essential details of the program implementation and discuss common problems associated with the development of hybrid computational codes.

The rest of the paper is organized as follows. In Sect. 2 the governing equations, detailed chemical mechanisms and numerical techniques are briefly described. The technologies and approaches used for program implementation are given in Sect. 3. Section 4 deals with the verification and testing of the developed code.

2. Governing equations
The Euler equations complemented with $N$ equations for chemically reacting species can be written in the conservation law form as follows:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}_x}{\partial x} + \frac{\partial \mathbf{F}_y}{\partial y} + \frac{\partial \mathbf{F}_z}{\partial z} = \mathbf{S}$$

(1)
The Euler equations are solved using shock-capturing MUSCL (monotonic upwind schemes for conservation laws) TVD schemes. The flow variables are reconstructed on the cell boundaries from cell-centered values with a 2nd or 3rd order of accuracy and then are limited using minmod slope limiter. Numerical fluxes are calculated with the HLLE (Harten-Lax-van Leer-Einfeldt) approximate Riemann solver. Time advancement is carried out using either explicit Runge-Kutta TVD schemes of 2nd or 3rd order, or semi-implicit ASIRK2C scheme of 2nd order [4], which is more efficient in terms of the computational time, because it allows one to integrate stiff chemical source terms using considerably larger time steps.

2.1. Numerical methods

The Euler equations are solved using shock-capturing MUSCL (monotonic upwind schemes for conservation laws) TVD schemes. The flow variables are reconstructed on the cell boundaries from cell-centered values with a 2nd or 3rd order of accuracy and then are limited using minmod slope limiter. Numerical fluxes are calculated with the HLLE (Harten-Lax-van Leer-Einfeldt) approximate Riemann solver. Time advancement is carried out using either explicit Runge-Kutta TVD schemes of 2nd or 3rd order, or semi-implicit ASIRK2C scheme of 2nd order [4], which is more efficient in terms of the computational time, because it allows one to integrate stiff chemical source terms using considerably larger time steps.

3. Implementation

The code is partially based on the existing HyCFS code [5] for the perfect gas and the in-home Fortran code for DW simulation and employs the same stack of technologies as the HyCFS, namely C++, CUDA, OpenMP, and MPI are used. The cost of the hybrid code development is usually considered significantly higher compared to the serial and MPI-based codes. However, in our case the development methodology and some engineering solutions from the HyCFS were re-used in the new code with minor changes. The main efforts were focused on the development of appropriate data structures, allowing for both easier code writing and fast execution on a variety of computational architectures, including GPUs and CPUs.
Flowfield variables are stored as a special `Container` class, which is designed to handle large number of variables required for a multispecies flow simulation. The main idea of the `Container` class is to store all variables for all computational cells in one large array of double precision values and store the required description of the data in the sub-class members of the `Container`. This approach seems to be the simplest way to provide an arbitrary number of quantities in `Container`, without employing more complicated techniques of objective-oriented programming, like multiple inheritance. The class also includes functions to access variables in Fortran-style syntax, locating variable by its name and so on. The physical and chemical properties of the species and descriptions of chemical reactions are also stored in separate structures (class `Element` and class `Reaction`, respectively). The calculations of element-specific quantities, like specific heats $C_p$ and $C_v$, chemical reaction rates $k_{f,r}$, etc., are implemented as methods of the respective classes. It allows for a cleaner and logically structured source code without reducing computational efficiency of the program. A set of utilities for importing chemical mechanisms in the CHEMKIN format and exporting results for visualization to the VTK and Tecplot formats are also created.

4. Computations
The problem of DW propagation in a channel filled with the stoichiometric mixture of $H_2$ and $O_2$ is used to verify the developed code. Initial pressure in the channel is set to 10132.5 Pa. On the right boundary the supersonic inflow with the Mach number 5.1 is imposed. The Zeldovich-von Neumann-Doering (ZND) solution profile is used as an initial flowfield, and the end state of the ZND profile is fixed at the left boundary of the computational domain.

1D computations are performed on an Intel Xeon E5-2683v3 processor using the CPU version of the developed code, while 2D computations are performed using 2 Nvidia Tesla K40 GPUs of a computational cluster.

4.1. 1D detonation wave
The computational domain length $L$ can be determined so that all quantities in the ZND solution must reach constant values at the rear boundary. For the case considered $L = 6$ cm, including 1 cm added ahead of the DW to keep it inside the computational domain. It takes into account that the DW propagation velocity changes periodically due to the 1D galloping instability.

![Temperature & Pressure profiles](image1.png)

**Figure 1.** Pressure and temperature profiles, stoichiometric mixture, $M = 5.1$.

![Mass fraction](image2.png)

**Figure 2.** Species mass fractions distribution, stoichiometric mixture, $M = 5.1$. 
The results of the 1D numerical simulations are compared with the ZND solution. Figure 1 shows pressure and temperature profiles. One can see close agreement of peak values in the pressure profiles. The values at the left boundary are also in close agreement. There is a certain difference in the shock wave width. It can be explained by the difference in the spatial resolution. The ZND solution has 50 000 points, while the numerical profile contains only 1200 points.

Figure 2 shows comparison of the species mass fractions. The final state values are also in close agreement. The burning zone widths are almost identical and we can conclude that the ignition times are also similar.

4.2. Detonation wave in a 2D channel
In this case channel length is the same $L = 6$ cm, height $H = 2$ cm. The grid resolution is 1200×400 points. All other parameters are the same as in 1D channel. To decrease the computational time and initiate the development of the instability, initial short wave sine disturbances of the transverse velocity are superposed to the flowfield at the beginning of the simulation. Such way of instability initiation was tested and used before in [6].

**Figure 3.** Detonation cells evolution. Stoichiometric mixture, $M = 5.1$. 

| $H$, cm | $L$, cm |
|--------|--------|
| 0      | 1      | 2      | 3      | 4      | 5      | 6      |
| 0      | 0      | 0.5    | 1      | 1.5    | 2      |

| $T = 2.5 \times 10^5$ s. |
|---------------------------|

| $H$, cm | $L$, cm |
|--------|--------|
| 0      | 1      | 2      | 3      | 4      | 5      | 6      |
| 0      | 0      | 0.5    | 1      | 1.5    | 2      |

| $T = 5.0 \times 10^5$ s. |
|---------------------------|

| $H$, cm | $L$, cm |
|--------|--------|
| 0      | 1      | 2      | 3      | 4      | 5      | 6      |
| 0      | 0      | 0.5    | 1      | 1.5    | 2      |

| $T = 15 \times 10^5$ s. |
Figure 3 shows the evolution of the cellular structure during the simulation. At $T \approx 2.5 \times 10^{-5}$ s only small detonation cells are visible, later, at $T \approx 5 \times 10^{-5}$ s, they transform into larger ones. It can be seen that the cellular structure consists of differently-sized cells and the number of half-cells is decreasing. Finally, at $T \approx 15 \times 10^{-5}$ s, the largest observed cells appear and a quasi-steady regime of the detonation is set in.

The number of half-cells $N_h$ across the channel at $T \approx 2.5 \times 10^{-5}$ s is approximately $N_h \approx 22$, while at $T = 15 \times 10^{-5}$ s it decreases to about $N_h \approx 10$. This behavior is in correspondence with the previous study conducted for a simple chemical model with one irreversible Arrhenius-type reaction [6] and can be generally explained by non-linear effects. Providing more detailed explanation of this effect is an interesting challenge for the future study.

4.3. Speed-up measurements

To estimate the efficiency of the parallel implementation, the 2D channel problem is solved using both GPU and CPU versions of the code. The GPU computations are performed on the Nvidia Tesla K40 GPU with 2880 CUDA cores and the CPU computations are performed on a 14-core Intel Xeon E5-2683v3 with hyper-threading capabilities. The efficiency of explicit and semi-implicit time integration schemes is also tested. The results are shown in tables 2 and 3.

Table 2. CPU vs GPU speed-up comparison.

| Wall-clock time, 1 step | CPU (Intel Xeon E5-2683v3) | GPU (Nvidia Tesla K40) | Speed-up |
|-------------------------|-----------------------------|------------------------|-----------|
| 9.6 s                   | 0.62 s                      | 15.48                  |

The data for the CPU version are given for the single-core run, but for more objective comparison the results should probably be scaled to the total number of CPU cores. If we assume 100% efficiency of OpenMP or MPI parallelization of the CPU version, the computational time would be $9.6/14 \approx 0.68$ s. So, the resulting speed-up, when using GPU, would be only $0.68/0.62 \approx 1.1$ times. Of course, in the real computation the efficiency for this number of cores would be about 60-70%. Thus, preliminary tests show that Tesla K40 GPU is roughly equivalent to 20-core CPU.

Table 3 shows the times of 2D computations performed on 2 Tesla K40 GPUs using ASIRK2C and RKTVD2 schemes. It is evident that the speed-up effect of the semi-implicit integration is rather small. One of the possible causes is emergence of high-pressure points on the DW front, which put severe limitations on the time step. More detailed investigation of this effect and further optimization efforts are required.

Table 3. Runge-Kutta TVD 2nd order vs ASIRK2C time integration speed-up comparison.

| Wall-clock time, 1250 steps | Runge-Kutta TVD 2nd order | ASIRK2C | Speed-up |
|-----------------------------|---------------------------|---------|----------|
| 26 min                      | 21 min                    | 1.23    |

| Wall-clock time, full computation | Runge-Kutta TVD 2nd order | ASIRK2C | Speed-up |
|----------------------------------|---------------------------|---------|----------|
| 5600 min                         | 4640 min                  | 1.20    |

5. Conclusions

A code for numerical simulation of detonation flows on hybrid GPU/CPU computational clusters has been developed and verified against the ZND solution of the DW. Preliminary speed-up tests show that a single Tesla K40 GPU is equivalent to approximately 20 cores of a CPU. In future, more detailed tests of parallelization efficiency will be performed.

The roadmap of the code development includes an improvement of accuracy by using the weighted essentially non-oscillatory (WENO) scheme of 5th order for spatial approximation and ASIRK4C scheme of 4th order for the time advancement.
Another important direction of the code development is the creation of various tools and utilities for the post-processing and analysis of the results including the DW position sensor for stability research, and built-in procedures for the shifting computational domain. The latter will allow one to keep the DW inside the computational domain in very long-time simulations and minimize the size of the zone filled with the incoming flow ahead of the DW.

Acknowledgments
This work was supported by the Russian Foundation for Basic Research (Grants № 18-33-00740, 18-08-01442, 16-57-48007).

References
[1] Shimizu H, Tsuboi N, Hayashi A K 2001 Study of detailed chemical reaction model on hydrogen-air detonation AIAA paper 2001–0478
[2] http://garfield.chem.elte.hu/Burcat/burcat.html
[3] Davidenko D, Gokalp I, Dufour E, Magre P 2006 Systematic numerical study of supersonic combustion in an experimental combustion chamber AIAA paper 2006–7913
[4] Zhong X 1996 Additive semi-implicit Runge-Kutta methods for computing high-speed nonequilibrium reactive flows Journal of Computational Physics 128 19–31
[5] Shershnev A, Kudryavtsev A, Kashkonsky A and Khotyanovsky D 2016 HyCFS, a high-resolution shock capturing code for numerical simulation on hybrid computational clusters AIP Conference Proceedings 1770 030076
[6] Borisov S, Kudryavtsev A 2017 Investigation of cellular detonations structure formation via linear stability theory and 2D and 3D numerical simulations AIP Conference Proceedings 1893 030042