

Direct Simulation Monte Carlo Method Technique with Application to Multiple Interacting High-Speed Jets

Wenhai Li & Foluso Ladeinde
Department of Mechanical Engineering
Stony Brook University

Summary

The goal of this project is to investigate the dynamics of multiple interacting high speed jets. To this end, a parallel three-dimensional direct simulation Monte Carlo (DSMC) code is developed. An iterative flux method is used to implement the pressure boundary conditions in DSMC. Jets expanding into a vacuum and into a specified pressure background are investigated, as is the influence of the background gas on the interacting jets. Empirical models will be developed. In the past year, the code has been successfully parallelized and executed in Seawulf Cluster, using domain decomposition with the MPI protocol. Code validation is in progress, for which a one-dimensional shock tube problem with Mach number up to 10 has been investigated. Preliminary results for three-dimensional interacting jets have also been obtained using the parallel code.

1. Background

With the rapid development of space technology, a lot of emphasis has been placed on the study of the interaction between rarefied free jets. Various jet interaction phenomena exist in the design of spacecraft, one example being multi-nozzle rockets.¹ The rockets are usually equipped with two or more nozzles, to provide large impulse and stability. Because of the high altitude, the pressure is low, which causes the plumes from each nozzle to have a large radial extent. Therefore, an interaction between neighboring plumes may occur. Another application is the spacecraft's Orbiter Reaction Control System

(RCS),² which comprises of many primary and vernier engines. The RCS can provide the thrust for altitude maneuvers and small velocity changes (along the orbiter axis) by firing the selected engines. If adjacent engines are fired simultaneously, an interaction between the two jets can occur. The jet interaction phenomena also can be seen in a satellite's Altitude Control System (ACS).³ This system is generally formed by an array of small thrusters. Because the size of the satellite is relatively small and the plume size is large in high altitude, jet interaction between the adjacent plumes can be observed.

The interaction between multiple plumes is receiving attention also

because of the development of microspacecraft,⁴ which can significantly lower fabrication and launch costs because of its smaller size and lower weight compared to conventional spacecraft. Many designs of micro-propulsion systems involve the use of thruster arrays for orbital maneuvers, such as attitude control and orbit rising. The thruster arrays can be batch fabricated using Microelectromechanical Systems (MEMS) techniques. They can also increase the flexibility for microspacecraft since thrusters can be fired in specific sequences or simultaneously to get desired impulse profile and thrust level for a particular maneuver. In thruster array, the distance between the thrusters is always very small, enhancing interaction between the plumes.

Interaction between the jets can have several effects on spacecraft operation, such as changes in the thrust impulse profile, the dynamics of jet impingement, heat flux and pressure force on spacecraft surfaces, contamination, stability, and sound generation. These phenomena can cause a lot of difficulties in the design of a spacecraft. For example, a backflow region is generated in the interaction region when the interaction effects are strong.³ The backscattered molecules can make the occurrence of contamination more likely and can also lead to relatively high heat flux and pressure force on spacecraft surfaces. Finally, it is noted that noise reduction could be a side effect of multiple jet interaction,⁵ as is the dynamic pressures that exceed the fatigue failure limit for metallic aircraft structures.⁶

2. Objectives of the Project

Understanding the physics of under-expanded interacting jets is the focus of this work, a three-dimensional parallel direct simulation Monte Carlo (DSMC) code is being developed to simulate the flow fields. An iterative flux method⁷ is investigated for the implementation of the pressure boundary conditions. Both jets expanding into vacuum and those expanding into a specified pressure background are being investigated. The influences of the background gas on the interacting jets are being studied and a rarefaction parameter will be developed to evaluate this influence. The 'primary-secondary' cells shock structures (Fig. 1) will be reproduced for the dual interacting jets in the near continuum regime, and an empirical expression will be developed for the position of the Mach disk in the secondary cell, with guidance from the numerical simulation results. The heat flux and pressure force at the orifice plane will also be investigated.

3. Computational Methods

The implemented DSMC algorithm is built around the same physical concepts as described by Bird. As a particle simulation method, DSMC does not solve the Boltzmann equation directly. Rather, the equation is solved by mimicking the physical nature of the (gas) molecular motion and intermolecular collisions, where simulated molecules are used in place of real molecules. (A simulated molecule can be regarded as a group of real molecules.) DSMC is also a statistical method. It is the collision statistical probability that determines whether or not a pair of molecules will collide, not

the distance between the two molecules. This is the main difference from the Molecular Dynamics (MD) method, which, on the other hand, is a deterministic method.

4. Current Status of Simulation Work

The DSMC code based on Bird's sequential algorithm⁸ has been parallelized using domain decomposition and the MPI message passing protocol. Preliminary results have been obtained. Particle movement and collisions are calculated in each domain. Communication between processes occurs only when computational particles cross domain boundaries. The parallel algorithm of the parallel DSMC code is shown in Fig. 2.

The code is written in Fortran 90 where dynamic memory allocation technique is used to save on memory. (A linked list data structure is used to store particle information.) Fig. 3 shows a segment of code that sends particles to neighboring processes. "Non-blocking receive" is used to receive particles from neighboring processes. Fig. 4 shows a segment of code that receives particles from neighboring processes. Table 1 compares the computation time for the argon normal shock wave (Mach 10) simulation using 2 processes. It shows that by using non-blocking communication algorithm, the computation speed increases by approximately 30%.

The one-dimensional (1D) DSMC code has been successfully parallelized and executed on the Seawulf cluster. The 1D shock tube problem has been used to validate the code. Some results are

shown below. Figs. 5 through 7 compare the solutions (density, translational temperature, and rotational temperature profiles of an nitrogen normal shock wave at Mach 10) for sequential and parallel simulations, using 2, 3, and 10 processes. Agreement between the sequential and parallel simulations is evident.

The results are also compared with the experimental data by Alsmeyer⁹ in Fig. 8, where agreement between the two results is also evident. Fig. 9 is a plot of the shock wave reciprocal thickness for different Mach numbers. The DSMC values are smaller than the experimental results.

"Load balancing" is an important issue for parallel application. Proper load-balancing can, to first order, be achieved by using an equal number of particles in each processor. A dynamic domain decomposition method is used to distribute an equal number of computational particles to each process. The technique has also been successfully implemented in the 1D shock tube code. Fig. 10 is the simulation result (2 processes) for an argon normal shock wave at Mach 10 with equal decomposed blocks. Fig. 11 is the simulation result for the same case but with dynamic domain decomposition. Thus, the particles are successfully equally distributed between the processes. The computation time performance for these two cases is compared in Table 2. It shows that by using dynamic domain decomposition, the computation time is reduced by approximately 50%.

The three-dimensional DSMC code has also been parallelized. The simulation of dual interacting jets

expanding into vacuum is reported. Fig. 12 shows the computation domain for the simulation. Figs. 13 and 14 are the density contours in the $x-z$ and $x-y$ symmetry planes for $Kn_s = 0.003$, $L/D = 3$ and $Kn_s = 0.03$, $L/D = 3$, respectively, where Kn_s is the stagnation Knudsen number, L is the distance between two orifices, and D is the diameter of the orifice.

5. Significance of Seawulf Cluster for the project

Since DSMC is a very expensive technique, and the problem is three-dimensional, intensive computational resources are needed. Table 3 lists the simulation cases proposed for the project. The last column in the table shows the completion status for each case using the Seawulf Cluster. It is clear that this project would not have been possible without the massive parallelization capability that is available in Seawulf Cluster. The estimated CPU hours needed for the computation are shown by adding the "CPU Hours" column in the table, we see that this project requires approximately 30,000 CPU hours.

For further information on this project contact:

Professor Foluso Ladeinde

Department of Mechanical Engineering
Stony Brook University
Stony Brook, NY, 11794-2300
631-632-9293

foluso.ladeinde@sunysb.edu

Reference

1. Houshang, B. E., Jay Levine and Alan Kawasaki, Numerical Investigation of Twin-Nozzle Rocket Plume Phenomenology, *Journal of Propulsion and Power*, Vol. 16, No. 2, 2000, pp. 178-186
2. Dagum, L. and Zhu, S. H. K., DSMC Simultaion of the Interaction Between Rarefied Free Jets, AIAA Paper 93-2872, 1993
3. Koppenwallner, G., Scaling Laws for Rarefied Plume Interaction with Application to Satellite Thrusters, Proceedings of the 14th International Symposium on Space Technology and Science, Tokyo, May 1984, pp. 505-512
4. Ketsdever, A., Selden, N., Gimelshein, S., Alexeenko, A., Vashchenkov, P., and Ivanov, M., Plume Interactions of Multiple Jets Expanding into Vacuum: Experimental Investigation, AIAA Paper 2004-1348
5. Cluass, J. S., Wright, B. R., and Bowie, G. E., Twin Jet Noise Shielding for a Supersonic Cruise Vehicle, AIAA Paper 79-670
6. Seiner, J. M., Manning, J. C., and Pooton, M. K., Dynamic Pressure Loads Associated with Twin Supersonic Plume Resonance, AIAA Paper 86-1539
7. Wu, j.-S., Lee, F., and Wong, S.-C., Pressure Boundary Treatment in Micromechanical Devices Using the Direct Simulation Monte Carlo Method, *JSME International Journal, Series 2*, Vol. 44, 2001, pp. 439-450 (2001)
8. Bird, G. A., *Molecular Gas Dynamics and Direct Simultation of Gas Flows*, 1st Ed., Oxford University Press, Oxford, 1994
9. H. Alsmeyer, Density profiles in argon and nitrogen shock waves measured by the absorption of an electron beam, *Journal of Fluid Mechanics*, Vol. 74, 1976, pp. 497.

List of Figures

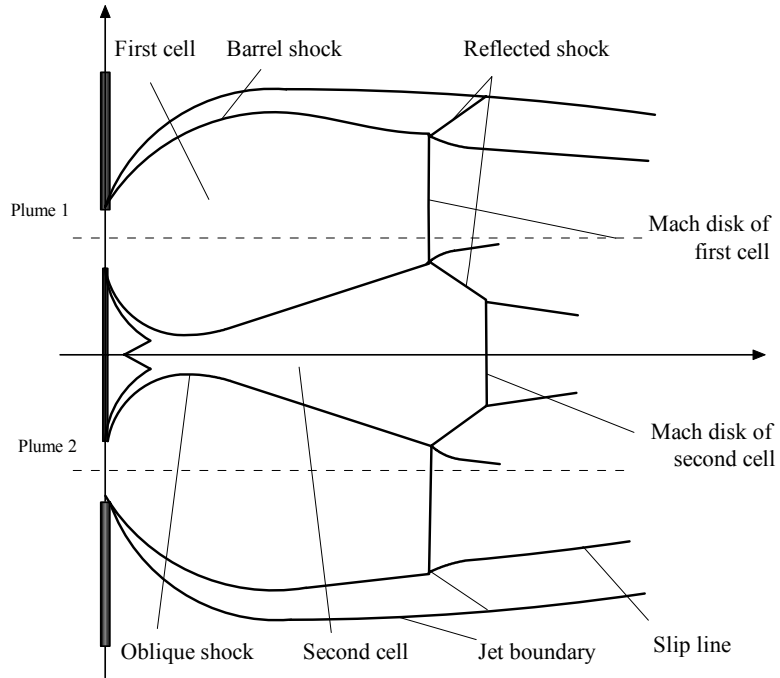


Fig. 1 The schematic structure of the flow field of dual interacting under-expanded jets.

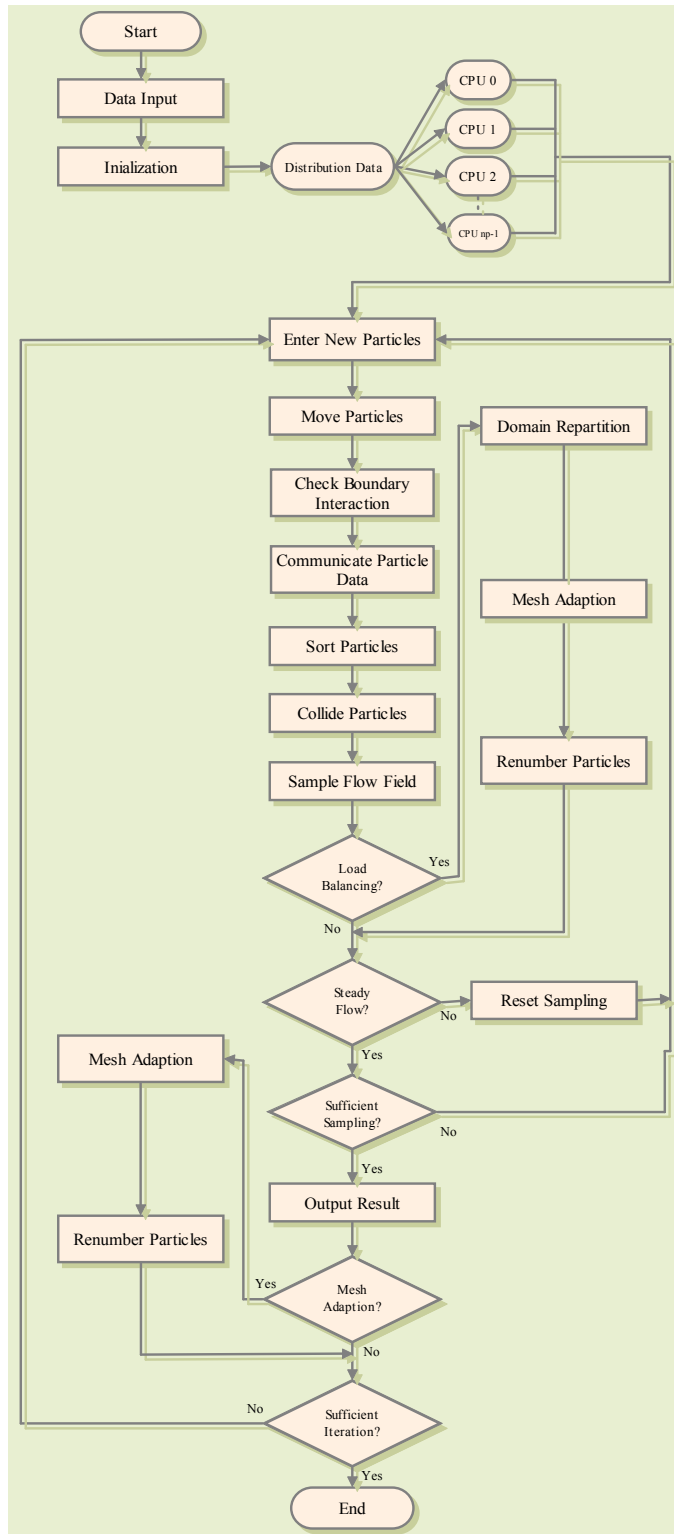


Fig. 2 Parallel algorithm for the DSMC code

```

n_send = 0
particle_ptr => root_sending_particle
do
  if(.not. associated(particle_ptr)) exit
  n_send = n_send + 1
  call MPI_ADDRESS(particle_ptr,elmoffset(n_send),ierr)
  elmsize(n_send) = 1
  particle_ptr => particle_ptr%next_ptr
end do
call MPI_TYPE_HINDEXED(n_send,elmsize,elmoffset,
&   type_particle,particle_msg,ierr)
call MPI_TYPE_COMMIT(particle_msg,ierr)
call MPI_SEND(MPI_BOTTOM,1,particle_msg,dest,tag,
&   MPI_COMM_WORLD,ierr)
call MPI_TYPE_FREE(type_msg,ierr)
<deallocate all the particles which have been sent>

```

Fig. 3 Segment of code that sends particles to neighboring processes

```

i = 0
do
  call MPI_IPROBE(MPI_ANY_SOURCE,MPI_ANY_TAG,
&   MPI_COMM_WORLD,flag,status,ierr)
  if (flag) then
    call MPI_GET_COUNT(status,type_particle,n,ierr)
    allocate(particle_recv(1:n))
    call MPI_RECV(particle_recv,n,type_particle,status(MPI_SOURCE),
&   status(MPI_TAG),MPI_COMM_WORLD,status,ierr)
    <insert received particles to the particle linked list>
    i = i+1
  end if
  if (i >= n_recv_msg) exit
end do

```

Fig. 4 Segment of code that receives particles from neighboring processes

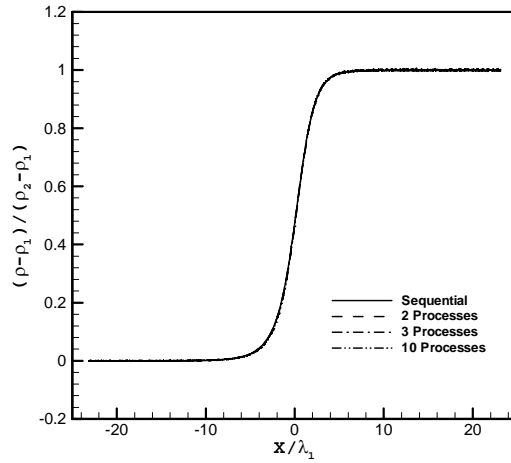


Fig. 5. Normalized density profiles of nitrogen normal shock wave at Mach 10 for the sequential results and parallel results using 2, 3, and 10 processes

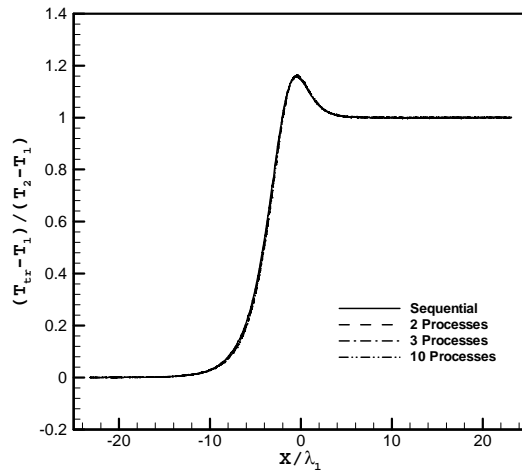


Fig. 6. Normalized translational temperature profiles of nitrogen normal shock wave at Mach 10 for the sequential results and parallel results using 2, 3, and 10 processes

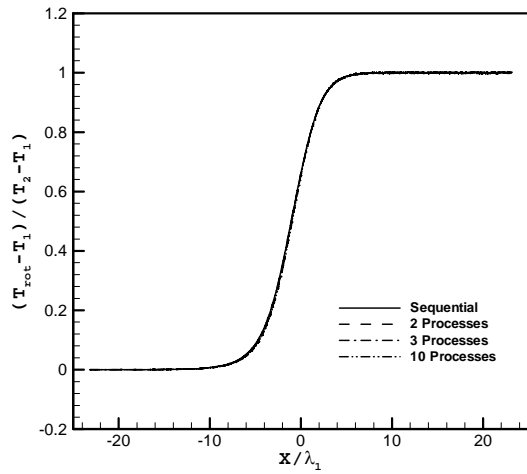


Fig. 7. Normalized rotational temperature profiles of nitrogen normal shock wave at Mach 10 for the sequential results and parallel results or 2, 3, and 10 processes

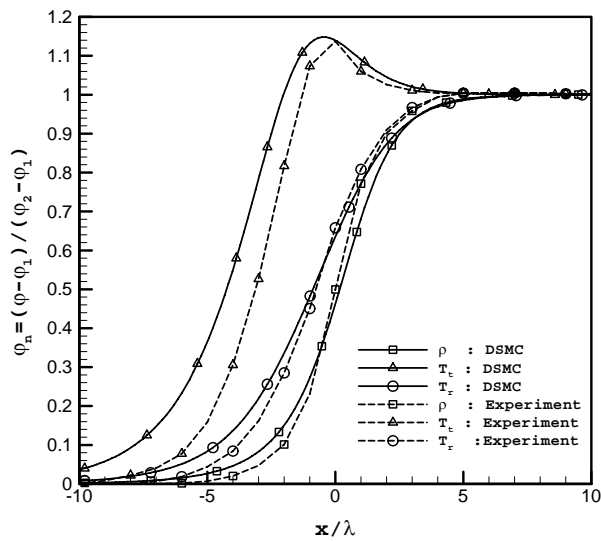


Fig. 8. Normalized density and temperature profiles for nitrogen at Mach 10 showing comparison with the experimental result

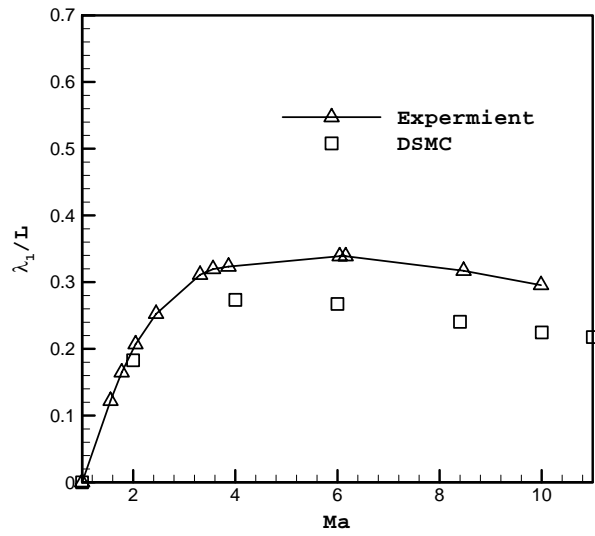


Fig. 9. Normalized shock wave reciprocal thickness for nitrogen showing comparison with the experimental results

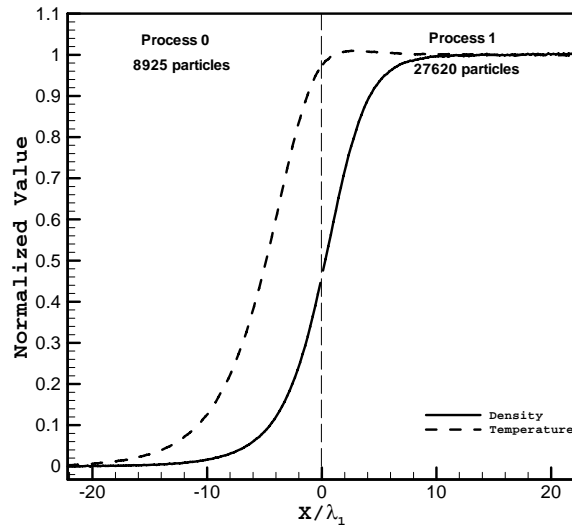


Fig. 10. Density and temperature profiles of argon normal shock wave at Mach 10 with using the equally decomposed domains.

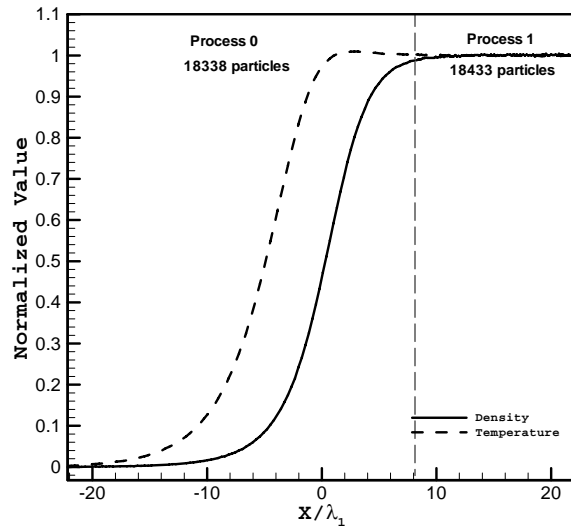


Fig. 11. Density and temperature profiles of argon normal shock wave at Mach 10 with using the dynamic domain decomposition technique.

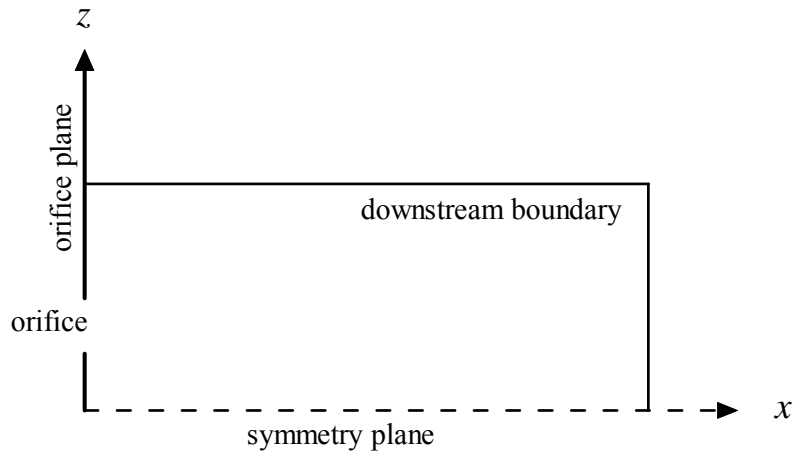


Fig. 12. Computation geometry for dual jet interaction

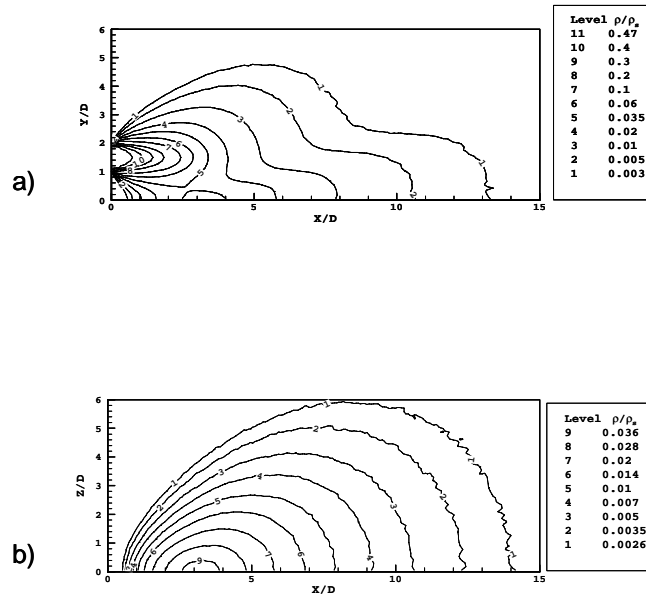


Fig. 13. Normalized density contour in (a) $x-z$ and (b) $x-y$ symmetry planes for $Kn_s = 0.003$ and $L/D = 3$

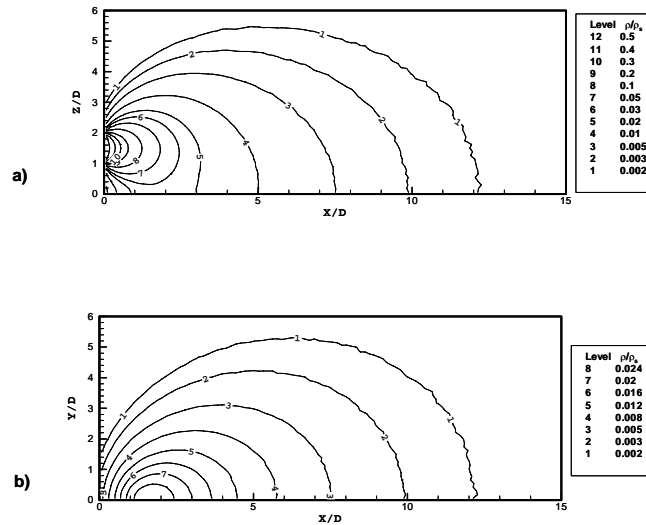


Fig. 14. Normalized density contour in (a) $x-z$ and (b) $x-y$ symmetry planes for $Kn_s = 0.3$ and $L/D = 3$

List of Tables

“Receive” Type	CPU Time (sec)
Using general blocking receive	660
Using non-blocking receive	452

Table 1. Computation time comparison for the parallel simulation of argon normal shock wave at Mach 10 by using 2 processes.

Decomposition Type	CPU Time (sec)
Static domain decomposition (domain is equally decomposed)	869
Dynamic domain decomposition (particles are equal distributed)	452

Table 2. Computation time comparison for two domain decomposition type in the parallel simulation of argon normal shock wave at Mach 10 by using 2 processes.

Project Catalogue	Simulation Cases	Number of Processes	Estimated CPU Hour	Progress (\checkmark = done)
1D parallel code development and debug	Fortran 90 code development	--	50	\checkmark
	Parallelization with general blocking communication	--	50	\checkmark
	Non-blocking communication	--	50	\checkmark
	Adaptive mesh	--	50	\checkmark
	Dynamic domain decomposition	--	50	\checkmark
1D code validation: 1D shock tube problem	Nitrogen $Ma = 10$	2	2	\checkmark
	Nitrogen $Ma = 10$	3	2	\checkmark
	Nitrogen $Ma = 10$	10	2	\checkmark
	Nitrogen $Ma = 2$	2	2	\checkmark
	Nitrogen $Ma = 4$	2	2	\checkmark
	Nitrogen $Ma = 6$	2	2	\checkmark
	Nitrogen $Ma = 8$	2	2	\checkmark
1D code parallel efficiency and scalability test: 1D shock tube problem	Argon $Ma = 10$ (blocking communication)	2	2	\checkmark
	Argon $Ma = 10$ (non-blocking communication)	2	2	\checkmark
	Argon $Ma = 10$ (static equally domain decomposition)	2	2	\checkmark
	Argon $Ma = 10$ (dynamic domain decomposition)	2	2	\checkmark
	Argon $Ma = 10$	2	2	x
	Argon $Ma = 10$	3	2	x
	Argon $Ma = 10$	5	2	x
	Argon $Ma = 10$	10	2	x
	Argon $Ma = 10$	20	2	x
3D parallel code development and debug	Fortran 90 code development	--	50	\checkmark
	Code parallelization	--	50	\checkmark
	Adaptive mesh	--	50	\checkmark
	Dynamic domain decomposition	--	50	\checkmark
	Pressure boundary condition implementation	--	50	\checkmark
3D parallel code validation: single under-expanded jet	Test the implemented pressure boundary condition	--	200	x
	Test the shock cell structure	--	200	x
	$Kn_s = 1.0$ Vacuum	4	20	x
	$Kn_s = 1.0$ $P_s / P_b = 20$	8	80	x
	$Kn_s = 0.1$ Vacuum	4	40	x
	$Kn_s = 0.1$ $P_s / P_b = 20$	8	120	x

	$Kn_s = 0.01$ Vacuum	8	80	x
	$Kn_s = 0.01$ $P_s/P_b = 50$	16	160	x
	$Kn_s = 0.01$ $P_s/P_b = 20$	16	240	x
	$Kn_s = 0.005$ Vacuum	10	100	x
	$Kn_s = 0.005$ $P_s/P_b = 50$	20	200	x
	$Kn_s = 0.005$ $P_s/P_b = 20$	20	300	x
	$Kn_s = 0.001$ Vacuum	16	160	x
	$Kn_s = 0.001$ $P_s/P_b = 100$	32	200	x
	$Kn_s = 0.001$ $P_s/P_b = 50$	32	320	x
	$Kn_s = 0.001$ $P_s/P_b = 20$	32	500	x
	$Kn_s = 0.0001$ Vacuum	16	160	x
	$Kn_s = 0.0001$ $P_s/P_b = 100$	32	200	x
	$Kn_s = 0.0001$ $P_s/P_b = 50$	32	500	x
	$Kn_s = 0.0001$ $P_s/P_b = 20$	32	800	x
3D parallel code simulation: dual interacting jets expanding into vacuum	$Kn_s = 1.0$ $L/D = 1.5$	4	40	x
	$Kn_s = 1.0$ $L/D = 3$	4	40	x
	$Kn_s = 1.0$ $L/D = 6$	4	40	x
	$Kn_s = 0.1$ $L/D = 1.5$	4	40	x
	$Kn_s = 0.1$ $L/D = 3$	4	40	x
	$Kn_s = 0.1$ $L/D = 6$	4	40	x
	$Kn_s = 0.01$ $L/D = 1.5$	8	80	x
	$Kn_s = 0.01$ $L/D = 3$	8	80	x
	$Kn_s = 0.01$ $L/D = 6$	8	80	x
	$Kn_s = 0.001$ $L/D = 1.5$	12	120	x
	$Kn_s = 0.001$ $L/D = 3$	12	120	x
	$Kn_s = 0.001$ $L/D = 6$	12	120	x
	$Kn_s = 0.0001$ $L/D = 1.5$	16	160	x
	$Kn_s = 0.0001$ $L/D = 3$	16	160	x
$Kn_s = 0.0001$ $L/D = 6$	16	160	x	
3D parallel code simulation: dual interacting jets expanding into background with specified pressure	$Kn_s = 0.1$ $L/D = 3$ $P_s/P_b = 50$	16	320	x
	$Kn_s = 0.1$ $L/D = 3$ $P_s/P_b = 20$	16	400	x
	$Kn_s = 0.01$ $L/D = 3$ $P_s/P_b = 50$	16	400	x
	$Kn_s = 0.01$ $L/D = 3$ $P_s/P_b = 20$	16	500	x
	$Kn_s = 0.005$ $L/D = 3$ $P_s/P_b = 50$	24	600	x

	$Kn_s = 0.005 \quad L/D = 3 \quad P_s/P_b = 20$	24	800	x
	$Kn_s = 0.001 \quad L/D = 3 \quad P_s/P_b = 100$	24	600	x
	$Kn_s = 0.001 \quad L/D = 3 \quad P_s/P_b = 50$	32	800	x
	$Kn_s = 0.001 \quad L/D = 3 \quad P_s/P_b = 20$	32	1000	x
	$Kn_s = 0.001 \quad L/D = 1.5 \quad P_s/P_b = 50$	32	800	x
	$Kn_s = 0.001 \quad L/D = 1.5 \quad P_s/P_b = 20$	32	1000	x
	$Kn_s = 0.001 \quad L/D = 6 \quad P_s/P_b = 50$	32	800	x
	$Kn_s = 0.001 \quad L/D = 6 \quad P_s/P_b = 20$	32	1000	x
	$Kn_s = 0.0001 \quad L/D = 3 \quad P_s/P_b = 50$	40	1200	x
	$Kn_s = 0.0001 \quad L/D = 3 \quad P_s/P_b = 20$	40	1600	x
3D parallel code simulation: three interacting jets	$Kn_s = 0.001 \quad L/D = 3 \quad \text{Vacuum}$	16	500	x
	$Kn_s = 0.001 \quad L/D = 3 \quad P_s/P_b = 50$	32	1000	x
	$Kn_s = 0.001 \quad L/D = 3 \quad P_s/P_b = 20$	40	1500	x
3D parallel code simulation: four interacting jets	$Kn_s = 0.001 \quad L/D = 3 \quad \text{Vacuum}$	16	500	x
	$Kn_s = 0.001 \quad L/D = 3 \quad P_s/P_b = 50$	32	1000	x
	$Kn_s = 0.001 \quad L/D = 3 \quad P_s/P_b = 20$	40	1500	x
Continuum CFD code: dual jet interaction	$Kn_s = 0.0001 \quad L/D = 3 \quad \text{Vacuum}$	16	1000	x
	$Kn_s = 0.0001 \quad L/D = 3 \quad P_s/P_b = 50$	16	1000	x
	$Kn_s = 0.0001 \quad L/D = 3 \quad P_s/P_b = 100$	16	1000	x

Table 3. List of simulation cases and the estimation of CPU hours needed.