Computers & Fluids 57 (2012) 66–75

Contents lists available at SciVerse ScienceDirect

Computers & Fluids

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

An object-oriented serial implementation of a DSMC simulation package

Hongli Liu, Chunper Cai *, Chun Zou

Department of Mechanical & Aerospace Engineering, Jett Hall 505, MSC 3450, New Mexico State University, Las Cruces, New Mexico 88003, United States

ARTICLE INFO

Article history:
Received 30 October 2010
Received in revised form 3 December 2011
Accepted 12 December 2011
Available online 19 December 2011

Keywords:
DSMC implementation
Hybrid grid
Object-Oriented Programming
Rarefied gas flow

ABSTRACT

This paper reports a scalar implementation of a multi-dimensional direct simulation Monte Carlo (DSMC) package named "Generalized Rarefied gAs Simulation Package" (GRASP). This implementation adopts a concept of simulation engine and it utilizes many Object-Oriented Programming features and software engineering design patterns. As a result, this implementation successfully resolves the problem of program functionality and interface conflictions for multi-dimensional DSMC implementations. The package has an open architecture which benefits further development and code maintenance. To reduce engineering time for three-dimensional simulations, one effective implementation is to adopt a hybrid grid scheme with a flexible data structure, which can automatically treat cubic cells adjacent to object surfaces. This package can utilize traditional structured, unstructured or hybrid grids to model multi-dimensional complex geometries and simulate rarefied non-equilibrium gas flows. Benchmark test cases indicate that this implementation has satisfactory accuracy for complex rarefied gas flow simulations.

Published by Elsevier Ltd.

1. Introduction

The national new spaceport program stimulates further interest on hypersonic non-equilibrium flows and space weather, including rarefied gas flows, plasma flows, and radiations. A set of compressible gas and plasma simulation packages are implemented to serve as an education and research platform for rarefied gas and plasma flows. These packages are named "Generalized Rarefied gAs Simulation Package" (GRASP) [1]. This paper presents an implementation of the direct simulation Monte Carlo (DSMC) [2] method, for multi-dimensional rarefied gas flow simulations.

The DSMC method is widely adopted for rarefied gas or plasma flow simulations. It is a stochastic simulation method in which each simulation particle represents a large number of physical gas molecules. The simulation particles possess physical properties such as position, velocity and, if applicable, internal energy information. If gas is highly rarefied, molecular movements and collisions are decoupled. Molecules either move freely or interact with boundaries. In the collision step, translational and internal energy is re-distributed between molecules according to a chosen collision model. At wall boundaries, molecules reflect back according to a selected reflection model. When molecules cross inlet or outlet boundaries, they are removed from the simulation without any further interactions. At the same time new molecules are introduced into the flow area from the free stream or inlet regions. The number of molecules introduced into the gas flow area and their velocity components depend on the boundary conditions [2]. During the past 40 years, there have been continuing proof and work to provide stronger supports to the validity, or even some further development for the DSMC method. For example, Wagner [3] provides a rigorous proof that DSMC simulations actually provide solutions to the Boltzmann equation in the limit of vanishing discretization and statistical error. Further evidence indicates that highly refined DSMC simulations provide results that agree with exact solutions to the Boltzmann equation, such as the near-equilibrium infinite-order Chapman–Enskog and the non-equilibrium Moment Hierarchy methods [4,5]. There have been several DSMC implementations in the literature, including those programs by Bird's DS2V/3 V [2], SMILE [6], MONACO [7], DAC [8], Icarus [9] and MGDS [10].

In software engineering with large scale programming, reusability and maintainability are two important requirements that Object-Oriented Programming (OOP) [11,12] can serve well. There are several major modules in a DSMC simulation package, i.e., movement, collision, particle indexing and input–output modules. Only the movement and particle indexing modules are different for different dimensions. As one of the most popular OOP languages, C++ has encapsulation, inheritance, polymorphism, and other features suitable to address the reusability and maintainability issues. To our best knowledge, This implementation is one of a few DSMC implementations completely utilizing C++ and some features are worthy to be represented to the community. There are two major issues to address in this paper, code architecture and three-dimensional simulations.

One common issue for DSMC code architectures is the crowdedness of code interfaces and functionalities, especially when it is desired to design and implement one code for multi-dimensional

* Corresponding author.
E-mail address: ccai@nmsu.edu (C. Cai).

0045-7930/$ - see front matter Published by Elsevier Ltd.
doi:10.1016/j.compfluid.2011.12.007

flows, i.e., two-dimensional, axisymmetric, and three-dimensional situations. As a common practice, programmers use the conditional compilation, such as "#ifdef ··· #else ··· #endif", to achieve the goal of one code for all dimension scenarios. This approach is prone to create confusions and usually results in incompleteness of the whole code structure. In software engineering, this can be classified as conflictions between subroutine functionalities and interfaces. To resolve this problem, OOP and several design patterns are adopted.

On the other hand, three-dimensional DSMC simulations are relatively challenging and time-consuming. Usually, there are two categories of meshing treatments for three-dimensional DSMC simulations. One adopts completely unstructured mesh which accurately triangulates an object surface, particles move cell by cell during each step and collide with surface accurately. However, these schemes need to track each particle to determine whether it passes a cell face or hits a surface. For each time step, the algorithm needs to determine which side of a cell that the particle can cross, and monitor the time left in this time step. Further, the type of DSMC simulations depend on the mesh size, for example, to efficiently and accurately simulate hypersonic flows around a reentry vehicle, at different altitudes, different mesh sizes are recreated according to different mean free paths. As a result, the engineering and simulation cost can be expensive. The other category is hybrid or cartesian grids, because DSMC decouples particles' collisions and movement, after the sub-step of collisions, particles are free to move fast, as long as the particles position can be accurately located in a cell. As a result the computation cost to determine the movement is significantly reduced. Further, simulations do not heavily depend on the mesh regeneration procedure. However, the surface representation can suffer from accuracy loss, when compared with triangulated surfaces in the other category. In general, this implementation can utilize many different grid systems to simulate multi-dimensional rarefied gas flows. Fig. 1 shows several grid types, and unstructured mesh systems are used for two-dimensional and axisymmetric simulations, to guarantee the program's stability and precision. For three-dimensional cases, this hybrid mesh divides the computational domain into solid cubic cells to track molecular trajectories efficiently, whereas the object surface can be triangulated by any major CAD/CAE software packages and read in by the simulator. In general, this implementation is not only accurate with unstructured meshes, but also efficient for three-dimensional simulations. During a three-dimensional simulation process, most of simulated molecules are efficiently tracked and moved within a hybrid or cartesian coordinate system, while a small portion in a small region adjacent to the object surface is treated delicately like an unstructured grid scheme.

The structures of this paper are organized as follows: Section 2 reports the data structure, several design patterns, and OOP styles for this implementation; Section 3 presents a special three-dimensional mesh scheme; Section 4 presents other minor implementation details; Section 5 presents several benchmark test cases; and Section 6 concludes this paper.

2. Data structure and design patterns

This section presents the code data structures and several design patterns to achieve a relatively open implementation architecture for multiple dimensions and methods.

The most important data structures for a DSMC code is particle storage. The complete particle' information includes global position, velocity, internal energy and a cell id. The cell id represents in which cell a particle locates. There are two popular particle storage data structures. One is from the book by Bird [2], with a large single array to store information for all particles. Because the number of particles in flow field is difficult to determine before code starts, the particle array size must be set sufficiently large to store all the particles' information. Actually, this storage scheme is popular for implementations. However, the storage space on hardware is wasted inevitably; and within a simulation, if it is desired to clone more particles to achieve higher precision, the whole simulation process may have to restart. Another approach is to utilize linked lists inside each cell, i.e., to divide the single large particle table into many small linked lists for each cell. As well known, a linked list is an efficient solution to achieve resizable array data structure. The requested storage space can always be adjusted to the number of particles. Some advanced implementations adopt two linked lists for each cell, one works as a current list the other one for backup.

The implementation adopts the first particle storage approach [2] with some modifications. This data structure is less demanding since changing particle's location from one cell to another requires only a cell ID change for the particle. As a result, the computation efficiency can be high. Different from Bird's original implementation, a special container class is introduced and serves as a memory manager for the particle array. This class provides a resizable particle array with two integer flags. Whenever the actual particle number equals to the maximum available particle slots, a larger size memory chunk in the free store is applied, particles information in the original smaller array is copied to the new array. When a simulation reaches a steady state, this resizable particle array can shrink to a sufficiently large size for particle storage. This class guarantees that there is always enough memory available even for a clone process.

As mentioned previously, when users plan to design and implement one code for multi-dimensional flows, it is desirable to resolve the conflictions between subroutine functionalities and interfaces. More importantly, the features of the ideal DSMC code should achieve reusability and modifiability. For DSMC simulation

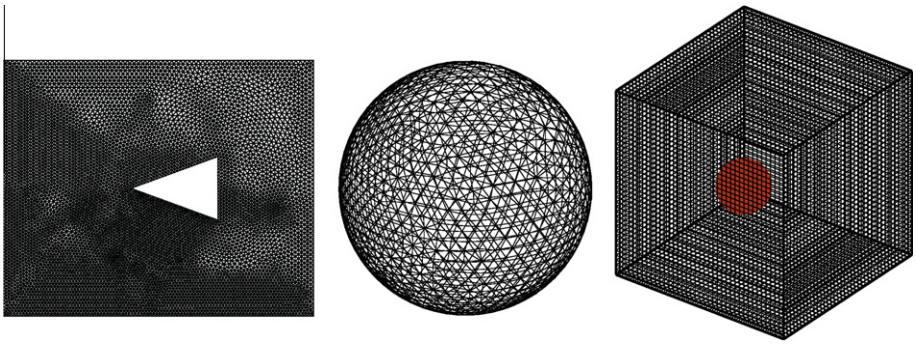

Fig. 1. Examples of grid schemes adopted. left: 2D; middle: object surface; right: 3D.