DOI: 10.14529/jsfi240204

Development of Compressible Mixing Layer Instability Simulated Using the Direct Simulation Monte Carlo Method

Alexandr V. Kashkovsky[1], *Alexey N. Kudryavtsev*[1], *Anton A. Shershnev*[1]

The Kelvin–Helmholtz instability developing in the mixing layer between two supersonic streams is simulated with the Direct Simulation Monte Carlo (DSMC) method using the SMILE-GPU software. No initial perturbations are introduced into the flow so that the disturbances are excited by and develop from the statistical fluctuations inherent in the DSMC method because of its stochastic nature. Multiple graphics processing units (GPUs) are employed for numerical simulations and efficient parallelization strategies for DSMC implementation on GPU clusters are presented. Between 0.4 and 1.6 billion of test particles are used to reproduce the development of the flow instability. The influence of the number of particles on mean flow properties and pulsation characteristics is investigated and discussed. It is shown that the pulsation characteristics are substantially affected by the number of particles because of a delay in the instability onset and vortex formation at a lower level of statistical fluctuations. A new algorithm for identification and analysis of vortex motion in noisy flow data is considered and applied to flowfields resulted from the unsteady DSMC simulations.

Keywords: rarefied gas flows, free shear flow instabilities, particle-based methods for kinetic equations, parallelization strategies, GPGPU computations with CUDA.

Introduction

The Direct Simulation Monte Carlo (DSMC) method proposed by G.A. Bird in 1963 is a particle-based method for solving the Boltzmann transport equation using probabilistic Monte Carlo techniques [1]. The gas in the DSMC method is modeled by a large number of test particles, the motion of which, at each time step, is divided into two stages: moving by inertia and collisions with other particles, selected in a stochastic way from the same grid cell.

For many years, the DSMC method has been intensively used to simulate rarefied gas flows, in particular, in applications to high-altitude aerodynamics and microfluid dynamics. In general, the DSMC method is applicable not only to rarefied flows, but also to those in the near-continuum and fully continuum regimes, though such simulations are possible only with a huge number of test particles. Nevertheless, the flows with Knudsen numbers $\mathrm{Kn} = \ell/L \sim 10^{-4} \div 10^{-5}$, where ℓ is the molecular mean free path and L is the macroscopic flow characteristic length, can be routinely performed with modern day computers.

The DSMC method is most often used to simulate steady flows because, in this case, time averaging can be efficiently employed to decrease statistical fluctuations which are an inherent feature of this approach. In recent years, however, there has been a significant increase in simulating unsteady fluid dynamic phenomena, such as hydrodynamic instabilities, using this numerical approach. So, the Richtmeyer–Meshkov and Rayleigh–Taylor instabilities caused by inertial and gravitational body forces, respectively, were simulated in [2, 3] while the first results of DSMC simulations of a shear flow instability were presented in [4] where the Kelvin–Helmholtz (KH) instability developing in a compressible mixing layer between two parallel supersonic streams was reproduced and in [5] where an unstable plane supersonic jet was considered.

Moreover, the DSMC method was even used for simulating fully developed turbulence. In [6] the Taylor–Green vortex flow was simulated and the famous Kolmogorov $-5/3$ spectrum law

[1]Khristianovich Institute of Theoretical and Applied Mechanics, Novosibirsk, Russian Federation

was reproduced. Later, DSMC simulations of the Couette flow between two plane walls were performed and it was shown that the law of the wall and other features of wall-bounded turbulence can also be successfully reproduced using this approach [7]. Further, when simulating near-continuum turbulent flows [8, 9], it was revealed that the continuum approach is not able to describe correctly the dissipation range of turbulence spectrum because it does not take into account thermal fluctuations [9] while they are properly represented in DSMC simulations.

In [2, 3] the authors tried to reduce the statistical fluctuations by using in 2D simulations of flow instabilities up to 10^{11} test particles, which required tens hours of computer time on 1.57×10^6 processors of Sequoia, an IBM Blue Gene/Q supercomputer. Our approach in [4, 5] was different: we used a large but reasonable number of particles so that the computations could be performed on a conventional hybrid CPU/GPU cluster available in many laboratories.

It means that the level of statistical noise in the computed flowfields was rather noticeable though it could be reduced by averaging over time intervals short in comparison with the hydrodynamic characteristic time scales. In some aspects, the statistical fluctuations inherent in the DSMC method can be even useful because their presence allows us to initiate the development of hydrodynamic instabilities in a “natural” way, without exciting the flow with any artificial initial disturbances. However, the influence of the statistical noise with an amplitude much higher than the level of real-world thermal fluctuations on the results of numerical simulations is rather a delicate question. Also, there is a problem of analyzing the noisy computational flowfields and extracting from them the flow instability characteristics such as the frequencies of the most perceived and the most amplified disturbances, their growth rates, as well as identifying the locations, motion and merging of the vortices developing in the unstable shear flow. Both these subjects are addressed in the current paper.

The rest of the paper is organized as follows. In Section 1 a brief description of the DSMC method and the SMILE-GPU numerical code are given along with the specifics of the GPU-based computations and other features of the numerical algorithm, Section 2 contains the problem formulation and computation results. Section 3 deals with a special procedure for detailed analysis of the obtained numerical results. Brief summary is given in the Conclusion section.

1. Code Implementation of Parallel Computations in DSMC Method

As was mentioned in Introduction, the DSMC method is a particle-based kinetic approach to numerical simulation of gas flows, in which model particles are moving and colliding with each other in a manner quite similar to real gas molecules. This method allows one to obtain a solution satisfying the Boltzmann kinetic equation. Kinetic methods are usually used to study the rarefied flows when the continuum approach based on the Euler or Navier–Stokes equations is inapplicable. The process of numerical simulation is split into consecutive time steps, which, in turn, consist of two main stages: free-molecular movement of particles and binary collisions. The computational domain is divided into cells with sizes comparable to a local mean free path. In the simulated collision process randomly selected particles located in the same cell collide with one another. This algorithm requires the so-called particle index that indicates which cell the particle is located in. This index list is updated on each computational step. Also, on each step the particles and their velocities are sampled in cells and this statistical data is used to calculate the mean number density, velocity, temperature and other macroscopic parameters in each cell.

The number of particles used in a DSMC simulation is limited only by the magenta CPU computational power. Usually, the number of model particles is fewer than the number of real molecules by a factor of 10^{10}–10^{20}. Thus, the DSMC method is typically employed for the numerical simulation of steady flows, where statistical fluctuations can be reduced averaging the solution over a sufficiently large time interval. When simulating unsteady flows, the information is averaged over a relatively small number of time steps, usually not exceeding few hundreds. This leads to large statistical fluctuations and makes the data analysis quite difficult. The fluctuations can be reduced by increasing the number of model particles, but since the fluctuations magnitude decrease proportionally to the inverse square root of samples, the sampling size has to be increased by orders of magnitude. Thus, DSMC simulations of unsteady flows, in particular of shear flow instabilities, inevitably require a large number of test particles used and can be performed only with efficiently parallelized software.

The SMILE-GPU software used in the current study is a highly efficient numerical solver designed to run on graphical processor units (GPUs). The amount of memory available on a GPU is quite limited, so computations were carried out on several GPUs using OpenMP and MPI technologies combined with domain decomposition technique.

1.1. Computations on GPU

A GPU is a device containing many multicore processors originally designed for manipulations with 3D graphical images. However, over time, their computational performance increased to levels where they became comparable with conventional CPUs. On the basic level, a modern GPU consists of a large number multicore processors and a specialized scheduler managing their workloads. Other specific feature of GPUs are as follows.

- A single core of GPU is considerably slower than a single core of CPU. However, the total number of GPU cores is an order of magnitudes higher than on a CPU. So, in theory, GPUs could performs computations faster.
- All computations on GPU are performed in the so-called SIMD (Single Instruction Multiple Data) regime, when all cores execute the same instruction in the code applied to a different set of data on each core. As a result, each statement in "if-else" operator is executed regardless of condition match. However, results of non-matching branch execution are not used. This specific manner of computation require appropriate numerical algorithms.
- GPU cores can only access the memory of their GPU and cannot use CPU memory. All memory exchange in initiated via CPU and is very slow. So, the best scenario for computation is to upload all the data to GPU, perform all required computations and download results into CPU memory.

SMILE-GPU uses CUDA (Compute Unified Device Architecture) application programming interface (API), developed by Nvidia for their GPU devices. This interface allows one to use any GPU as some abstract device that has some number of computational threads grouped into blocks. Computations are performed in so-called kernels, specific function executing on all cores of a device. Built-in global variables allow each thread to identify its index number and the index of its block, and the indices of thread and block can then be used to assign specific portion of data to process. The total number of computational threads can be larger than the number of physical cores on the device and CUDA scheduler manages their execution order to provide maximum efficiency (at least, in theory). This allows a programmer to focus on the numerical algorithm and not on details of hardware architecture.

Considering these features, the approach of parallelization by data, when all threads can simultaneously perform the same operation on different data sets, seems to be optimal for programs running on GPUs. Main entities in the DSMC method are particles and cells, so at any given time all computational threads should process either particle or cell. However, because of the SIMD approach number of operations for each thread would be the same. As a result, if the size of data is different on each thread, the efficiency would drop, because there would be threads performing basically useless computations. This can happen, for example, when there is a different number of particles in cells, because of different gas density.

Unfortunately, it is hard to make a theoretical estimate for the algorithm efficiency when using GPU, because it depends not only on the data size, but also on the data structures in program implementation, and their location in memory. So, usually, the most computationally efficient implementation is a result of trial and error. For example, during the SMILE-GPU development, 7 variants of collision algorithms and 4 algorithm of sampling were tested.

In binary collisions any pair of molecules can engage in chemical reaction resulting in pair of particles of different species. Chemical reactions occur relatively infrequently, requiring additional computations and subsequent rebuild of particle-cell index. The straightforward implementation of the algorithm leads to a dramatic order of magnitude drop in efficiency. So, in SMILE-GPU chemical reactions are "delayed", meaning that in collision procedure the pair of reacting particles is flagged and added to a list of dissociating/exchanging particles, and reaction itself is realized in separate kernel afterwards. This approach allows to even out the computations on different threads and the particles index will not be needed until next iteration of collisions, so it is rebuilt only on the next time step. But, unfortunately, this trick is not free because the preliminary flagging of particles makes them unavailable for other reactions in this cell, so the time step needs to be reduced to match the number of collisions in cell. Tests show that when the number of collisions is 10–20% of total number of particles delayed reactions have negligible effect on the efficiency. However, for the processes of internal energy exchange it leads to an unacceptably small time step, so these process are realized during main collision procedure despite the certain amount of useless computation.

1.2. Workload Distribution between GPUs

Domain decomposition is used for computations on multiple GPUs. The domain is split into subdomains and each is assigned to a GPU. If some particle moves outside of the domain, it is transferred to the respective GPU during the exchange stage of the algorithm. This is a conventional approach to the parallelization of a DSMC code, but there are some differences stemming from the GPU architecture.

Because the CPU-GPU transfers are usually very slow, the domain decomposition should minimize inter-partition exchange. This is achieved using the wind reference frame, when the velocity vector is oriented along the X axis. The subdomains are formed by grouping cells along the inflow velocity vector. So, the particles following straight streamlines will fly through the domain without crossing partition boundaries. In this case, only particles with relatively large transversal thermal velocity can move to neighboring partitions. In real problems of practical interest with complex geometries, shock waves, mixing and boundary layers this simple assumption about velocity direction does not always hold, however, it allows to considerably increase efficiency of the computation.

Another important aspect of the computation is the balanced workload on all computational devices. This problem applies to both CPU and GPU codes. It is virtually impossible to obtain a perfectly balanced data distribution in a DSMC run because of the statistical nature of the method and fluctuations in instant flowfields. This also leads to fluctuation in a single step execution time. A popular approach used in CPU DSMC codes is to distribute cells randomly among the threads to ensure that on each thread there are cells with both high and low computational loads. This allows for balanced computation with acceptable efficiency achieved via more extensive exchange. This technique cannot be used in GPU computations because the exchange is slower, and the wall time for each step is considerably lower, so any slow thread would drop total efficiency and performance drastically.

DSMC computations of steady flows usually start from uniform flowfield which evolves during the simulation to a resulting flowfield with some distribution of gasdynamic parameters. The dynamic balancing is required for this unsteady stage of computations. For unsteady flows balancing is iteratively carried out throughout the whole computation. The main idea behind the balancing process is similar to the greedy algorithm approach: transfer (i.e. reassign) cells from GPUs with high workload to GPUs with low workload. We call this algorithm Direct Timer Load Balance (DTLB). Each GPU has a correction factor C_i which is proportional to N_i, the number of cells assigned to ith GPU. If N_c is the total number of cells in the domain and N_g is the total number of GPUs, then N_i is calculated as

$$N_i = C_i N_c / N_g. \tag{1}$$

The coefficients C_i are recalculated using measurement of timers $T_{c,i}$. At the start of computation $C_i = 1$ for all GPUs. We calculate the maximum $T_{c,max}$ of all $T_{c,i}$, and then the value of C_i is updated according to formula

$$C_i^{(n)} = C_i^{(n-1)} \cdot \left(1 + \frac{T_{c,max} - T_{c,i}}{T_{c,max}}\right), \tag{2}$$

where (n) is the recalculation iterator, $T_{c,max} - T_{c,i}$ is the ith GPU time of wait and should be equal to zero for a perfectly balanced distribution. As can be seen, C_i for the GPU with maximum workload ($T_i \equiv T_{max}$) will remain unchanged, and for the rest of GPUs it will increase. Finally, the factors are renormalized

$$C_i = C_i^{(n)} / \sum_k^{N_g} C_k^{(n)}. \tag{3}$$

Tests show that after 10–20 iteration this approach yields approximately 95–99% efficiency.

2. Numerical Simulation of the KH Instability in Compressible Mixing Layer

2.1. Problem Formulation and Numerical Setup

It is well known that a mixing layer between two parallel streams is unstable even at low Reynolds numbers [11]. The linear instability of a compressible mixing layers was investigated in detail in [12–14]. The linear stability analysis shows that the instability characteristics of the compressible mixing layers are strongly affected by the so-called convective Mach number

$$\mathrm{M}_c = \frac{U_1 - U_2}{a_1 + a_2}, \tag{4}$$

where U_1 and U_2 are the velocities of two streams and a_1 and a_2 are the sound speeds in them. At subsonic M_c the KH instability dominates in the mixing layer while at $\mathrm{M}_c > 1$ two new, the so-called supersonic instability, modes emerge whose disturbances radiate from the mixing layer as acoustic waves. Moreover, at $\mathrm{M}_c \leq 0.6$, 2D disturbances are the most unstable and grow faster than the 3D ones.

The mixing layer is a canonical example of a highly-unstable free shear flow and its instability occurs even at low Reynolds numbers, so that the instability should manifest itself in rarefied flows. Nevertheless, it seems that until recently the mixing layer instability was simulated using a kinetic approach only in one work [15] where the simulation was performed by solving the model kinetic equations with a deterministic finite difference method. Lately, for this purpose, DSMC simulations were also used [4] as well as the simulations based on the model kinetic equations and the Boltzmann transport equation [16].

In the current study, we simulate a plane mixing layer between two streams of monatomic gas (argon, Ar) with the temperature $T_1 = T_2 = 300$ K and the pressure $p_1 = p_2 = 2.5$ Pa. These conditions correspond to the mean free path $\lambda_1 = \lambda_2 = 0.0029$ m. The flow velocities of two streams are equal to $U_1 = 805$ m/s and $U_2 = 483$ m/s so that their Mach numbers are $\mathrm{M} = 2.5$ and $\mathrm{M} = 1.5$, respectively. Thus, the convective Mach number $\mathrm{M}_c = 0.5$ and the most unstable mode is the 2D KH disturbances.

The numerical simulations are performed in a rectangular domain 40×16 meters. The two streams start mixing on the inflow boundary where model particles are introduced with the corresponding Maxwellian distributions. No artificial disturbances are imposed on the flow and instability emerges in a natural manner from statistical fluctuations inherent to the DSMC approach. However, the number of model particles is many order of magnitude smaller than the number of real molecules so that the amplitude of fluctuations is considerably larger than in nature. To investigate the effects of fluctuations on the instability emergence and development three simulations with $N_p = 0.4$, 0.8 and 1.6 billion (10^9, bln) of model particles, respectively, are carried out.

The computations are performed using a two-level grid. The first level grid consists of 1000×400 cells, while the second level grid used for particle collisions is constructed automatically according to local flow gradients to ensure there are 5 or more model particles in a collision cell. The time step is set to $\Delta t = 10^{-6}$ s. From 6 up to 12 Nvidia V100 GPUs are used in computations.

All obtained flow parameters are averaged over a relatively small interval equal to 100 time steps (or 10^{-4} s). During first 100,000 time steps no data are gathered because this period corresponds to a transient starting process. The averaged flowfields will be referred below as *frames*. The number of frames n_f obtained in the three simulations are 2000, 2358 and 5254, respectively.

Reducing the averaging period will increase the fluctuations in flowfields, while its increasing will smooth out some of the features and disturbances in the flow. Moreover, from the generalization of the Kotelnikov [17] theorems for restoring a continuous multi-frequency signal from discrete samples, it follows that the maximum correctly restored frequency is 2 times higher than the polling frequency. For the case considered, the maximum frequency equals 5 kHz, but in fact most frequencies of interest in our study are lower than 1 kHz.

A typical frame of the transversal velocity flowfield $u_y(x, y)$ is shown in Fig. 1. It is seen that the flow is unstable and there are well known KH "billows" inside the mixing layer. The vortices emerge close to the left (inflow) boundary, grow as they travel downstream and merge into larger vortices. The statistical noise is also clearly visible in Fig. 1.

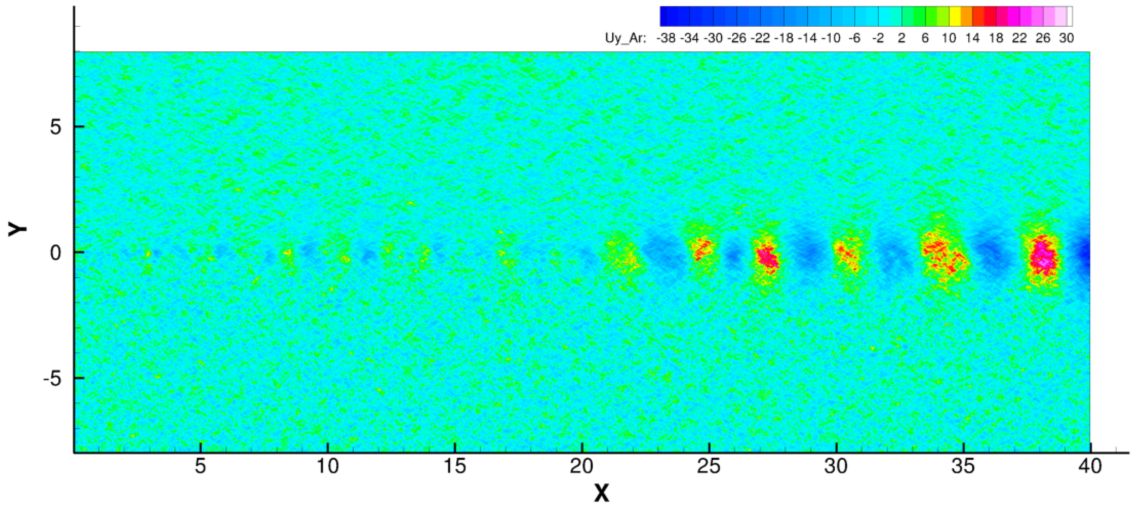


Figure 1. Flowfield of transversal velocity $v(x, y)$ averaged over 100 time steps for $N_p = 0.4$ bln

2.2. Mean Flowfields

The mean flowfield $Q(x, y)$ and the root mean square (RMS) pulsation flowfield $q'_{rms}(x, y)$ of an arbitrary flow quantity $q(x, y, t)$ are calculated by averaging over n frames:

$$Q = \frac{\sum_{i=1}^{n} q_i}{n}, \qquad q'_{rms} = \sqrt{\frac{\sum_{i=1}^{n} (q_i - Q)^2}{n}}. \tag{5}$$

In Fig. 2 flowfielfds of mean longitudinal $U(x, y)$ and transversal $V(x, y)$ velocities are presented for $N_p = 0.4$ bln. It is seen that the mixing layer thickness gradually grows downstream. The weak shock waves (Mach waves) originating from the meeting point of two streams at the left boundary are clearly visible in the transversal velocity flowfield $V(x, y)$. The inclination angles of the weak shock waves in the upper and lower streams equal 23.6° and 41.8°, respectively. They match perfectly with the Mach angles $\alpha_M = \sin^{-1}(1/M)$ for these streams. The oblique intersecting straight lines behind the shock waves are acoustic characteristic lines of two families. They are visible due to weak velocity perturbations propagating from the mixing layer as well as from the upper and lower boundaries. The same oblique intersecting Mach lines can be seen in many experimental schlieren visualizations of supersonic flows.

The mean profiles of longitudinal velocity $U(y)$ close to the outflow boundary, at $x = 39$ m, are shown in Fig. 3. The profiles obtained in the simulations with different numbers of test particles are compared in Fig. 3a. It is evident that they coincide almost perfectly.

In Fig. 3b the computed velocity profile at the same cross-section $x = 39$ m is compared with the analytical expression

$$U(x, y) = U_c + \frac{\Delta U}{2} \operatorname{erf}\left[\frac{\sqrt{\pi}(y - y_c)}{\delta_\omega(x)}\right], \qquad U_c = \frac{U_1 + U_2}{2}, \qquad \Delta U = U_1 - U_2 \tag{6}$$

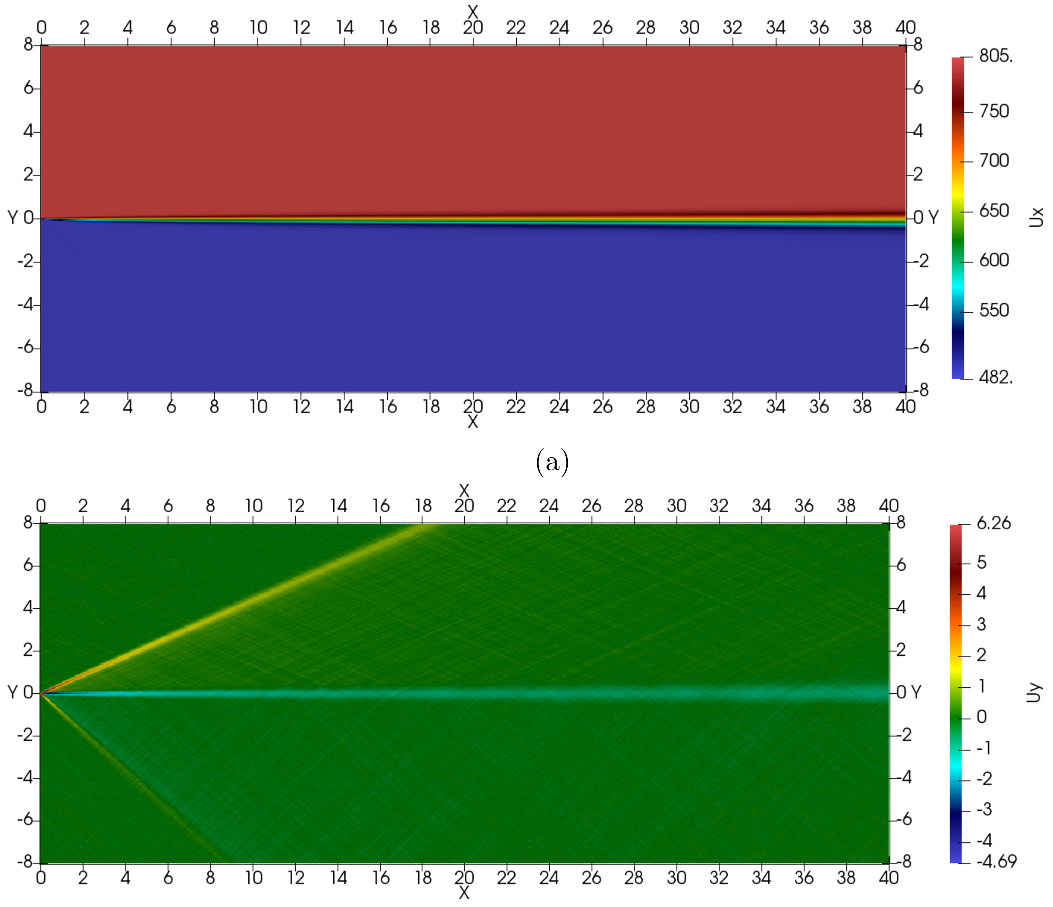


Figure 2. Mean longitudinal $U(x, y)$ (a) and transversal $V(x, y)$ (b) velocity flowfields for $N_p = 0.4$ bln

obtained in [18] as an approximate solution of the boundary layer equations. Here δ_ω is the so-called vorticity thickness defined as

$$\delta_\omega = \frac{\Delta U}{(\partial U/\partial y)_{\max}} \tag{7}$$

and y_c is the coordinate of the point where $U = U_c$.

One can see that the numerical and analytical solutions are in close agreement and the values of δ_ω and y_c for this cross-section equal 0.79 m and 0.085 m, respectively. It is worth noting than another analytical profile $U(x, y) = U_c + (\Delta U/2)\tanh(2y/\delta_\omega)$, which is also frequently used to approximate the mixing layer profile, does not yield so good agreement with our numerical results.

The Görtler approximate solution [18] predicts that $y_c = 0$, i. e. the mixing layer centerline coincides with the axis $y = 0$. However, as was shown in [19], the next order approximation implies a centerline deflection. The DSMC simulation reproduces this peculiar feature. The centerline location as a function of the longitudinal coordinate $y_c(x)$ is shown in Fig. 4a. As can be seen, it slightly deflects from the line $y = 0$ toward the lower (slower) stream and the deflection grows downstream.

The mixing layer thickness increases with the longitudinal coordinate. It is well known that initially the mixing layer spreads similar to a laminar boundary layer on a flat plate so that its

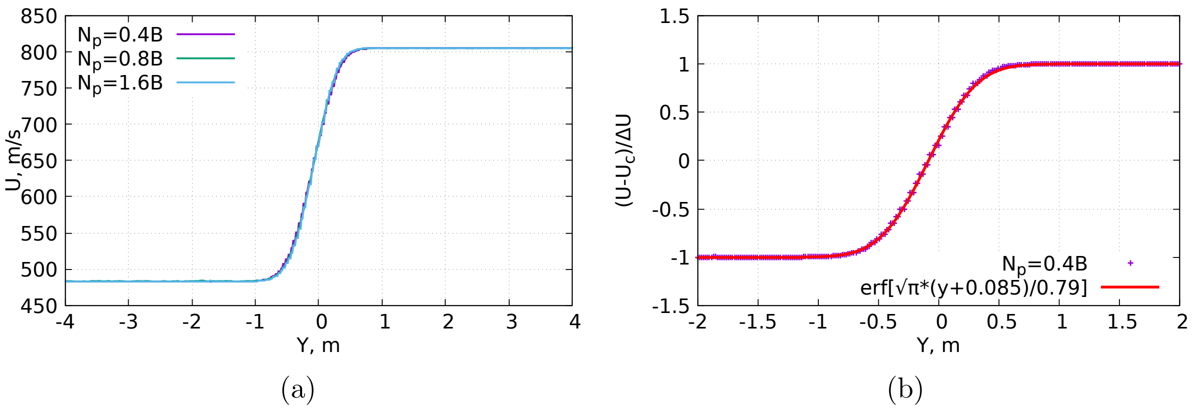


Figure 3. Mean velocity profiles $U(y)$ at $x = 39$ m, comparison of the profiles for different numbers of test particles (a) and comparison with an analytical expression (b)

thickness grows proportional to $\sqrt{x}$. However, after the emergence of large-scale vortices, their pairing becomes the main mechanism of thickness growth [20]. It seems that this mechanism continues to dominate even in the turbulent regime when the large-scale vortices persist on the background of small-scale 3D pulsations [21]. As a result of successive vortex pairings, the mixing layer thickness grows linearly with the x coordinate.

To determine the character of this dependency in the numerical simulations performed, a fitting curve has been calculated using a built-in function of the open source visualization software Gnuplot [22]. The result is shown in Fig. 4b. It can be seen that at $x \leq 15$ m the square root fit reproduces closely the behavior of the numerical curves. Farther downstream they are well-fitted by linear functions, however, the slopes are slightly different with the maximum slope for $N_p = 0.4$ bln and the minimum one for $N_p = 1.6$ bln. It can be explained by a smaller statistical noise level at a large particle number. As a result, the mixing layer is later excites and the growth of disturbances delays (see below). Thus, this characteristic reveals a dependence on the number of test particles.

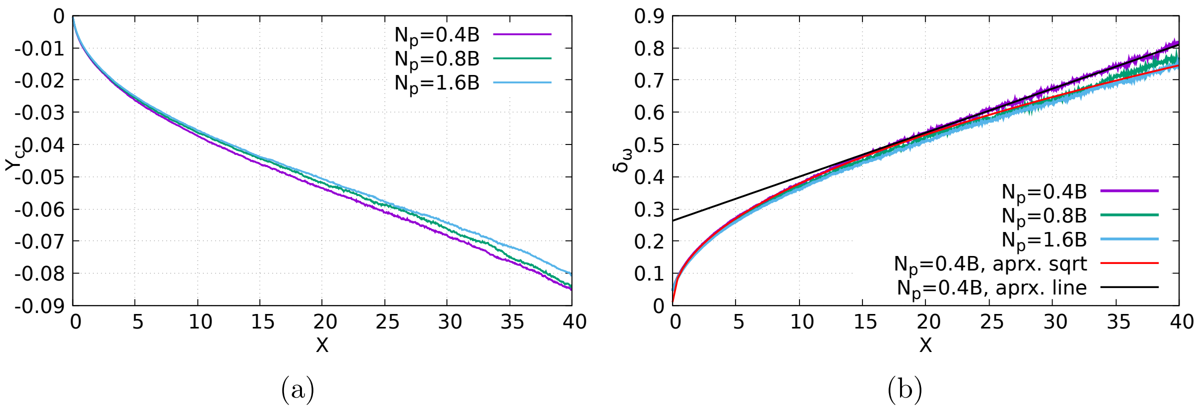
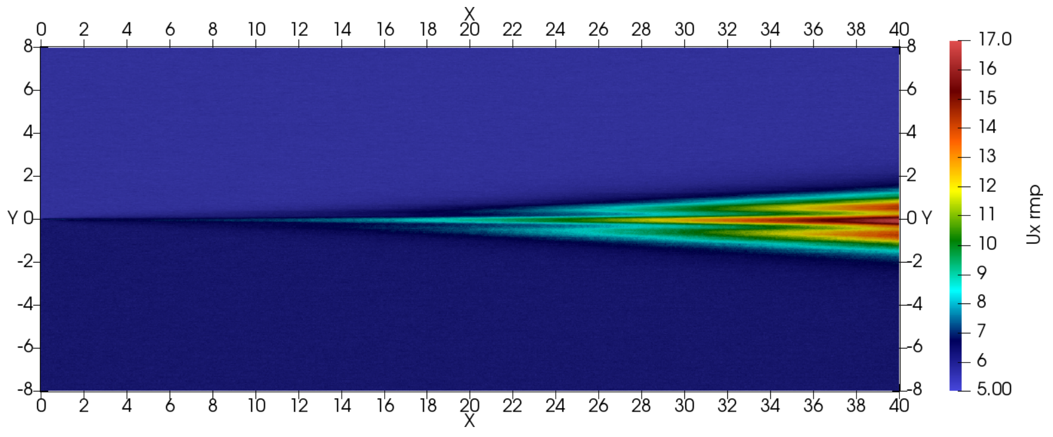


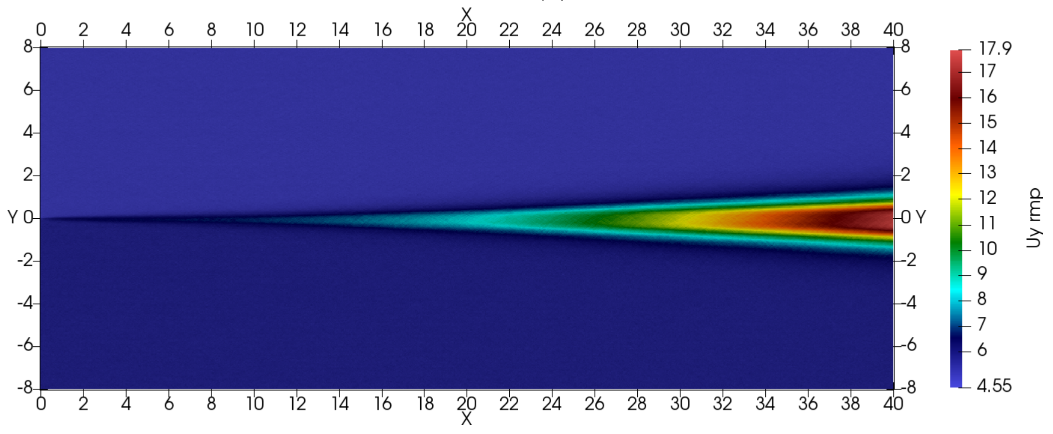
Figure 4. Centerline deflection (a) and growth of vorticity thickness (b) for mixing layer

2.3. Growth of Disturbances

Figure 5 shows the flowfields of u'_{rms} and v'_{rms}, RMS pulsations of two velocity components, for $N_p = 0.4$ bln. One can see their fast growth starting approximately from the streamwise location where the layer thickness begins grows linearly. A specific feature of the u'_{rms} flowfield that, closer to the outflow boundary, its distribution across the mixing layers has three maxima.



(a)



(b)

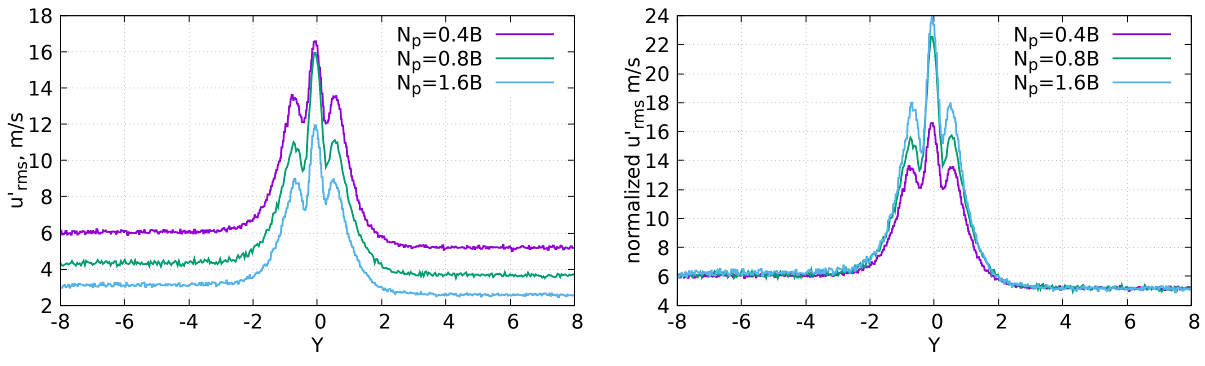
Figure 5. Flowfields of u'_{rms} and v'_{rms}, RMS pulsations of longitudinal (a) and transversal (b) velocity components, for $N_p = 0.4$ bln

The profiles of RMS pulsations of u'_{rms} in the cross-section $x = 39$ m for the same case $N_p = 0.4$ bln can be seen in Fig. 6.

Figure 6a shows that the profiles for the different number of particles are similar on qualitative level but differ quantitatively. This difference is resulted from the different level of statistical noise. It is well known that an increase in the number of samples by a factor of N reduces the standard deviation of a quantity by a factor of $\sqrt{N}$. So, the RMS pulsation profiles are multiplied by the factor $\sqrt{N_p/N_p^0}$ where $N_p = 0.4$ bln and the normalized RMS pulsations $\sqrt{N_p/N_p^0}\, u'_{rms}$ are presented in Fig. 6b. As can be seen, after that the pulsations in the free stream regions coincide but in the mixing layer itself they differ substantially.

Thus, the characteristics of pulsations depend substantially on the number of particles used in a simulation N_p. The free stream values of fluctuations are simply proportional to $\sqrt{N_p}$. Thus, they will decrease with an increase in N_p and vanish as $N_p \to \infty$. For the number of test particles equal to the number of real molecules, the magnitude of the free stream pulsations would be equal to that of thermal fluctuations in real gas.

However, the amplitude of disturbances in the mixing layer does not follow this normalization because they are resulted from the physical process of disturbances amplification due to the flow instability. The disturbances in the mixing layer are instability waves (with superimposed statistical fluctuations). They are originated from statistical fluctuations but then they are selectively amplified due to pumping of the energy by the Reynolds stresses from the mean



(a) (b)

Figure 6. Original u'_{rms} (a) and normalized $\sqrt{N_p/N_p^0}\,u'_{rms}$ (b) profiles of RMS pulsations of longitudinal velocity in the cross-section $x = 39$ m

shear flow. At early stages of the hydrodynamics instability they grow exponentially, later their growth saturates because of nonlinear effects. With an increase in the number of particles, the magnitude of statistical fluctuations decreases so that the initial amplitude of instability waves decreases too. As a result, they will grow and reach approximately the same amplitude, however it will happen farther downstream from the meeting point of two mixing streams.

It is interesting to compare the RMS profiles with the results of the linear stability theory (LST). The linearized compressible Navier–Stokes equations along with the disturbance boundedness conditions at $y \to \pm\infty$ comprise a LST eigenvalue problem, see [11]. The eigenvalue problem is solved using the VMLS3D code developed by one of the authors. The LST computations are performed for the analytical profile (6). In Fig. 7 the profile v'_{rms} at $x = 39$ m is compared with the LST eigenfuction of a spatially growing disturbance for the frequency corresponding to the maximum growth rate. The distributions are different in the freestreams where the LST eigenfunction exponentially decays while the flow disturbance in the DSMC simulation approaches a nearly constant value equal to the freestream level of statistical noise. At the same time, the two distributions are in close agreement inside the mixing layer.

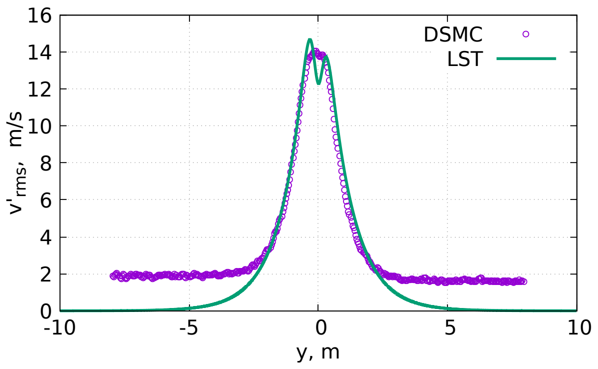


Figure 7. Comparison of the DSMC profile of v'_{rms} at $x = 39$ m ($N_p = 0.4$ bln) with a LST eigenfunction

The statistical noise introduced by the DSMC method has a broadband spectrum. However, the flow instability amplifies its harmonic components selectively. As a result, the most unstable disturbance predicted by the linear theory starts dominating at some distance downstream. Even farther downstream nonlinear interactions between the harmonics result in an amplification of the subharmonic waves and the vortex pairing. A few successive pairings occur within the computational domain, however, as can be seen, even near the outflow boundary the transverse velocity fluctuation profile v'_{rms} within the mixing layer keeps resemblance with the LST eigenfuction.

3. Identification of Vortices and Analysis of Vortex Motion

This section deals with a specialized procedure developed specifically for an analysis of instabilities in the presence of statistical fluctuations. Calculation of vorticity, Q-criterion and other techniques based on the spatial derivatives of flow quantities do not work in the case of statistically noisy data, so another approach, based on some integral quantities, was proposed.

For any frame, for each cell i we calculate the quantity

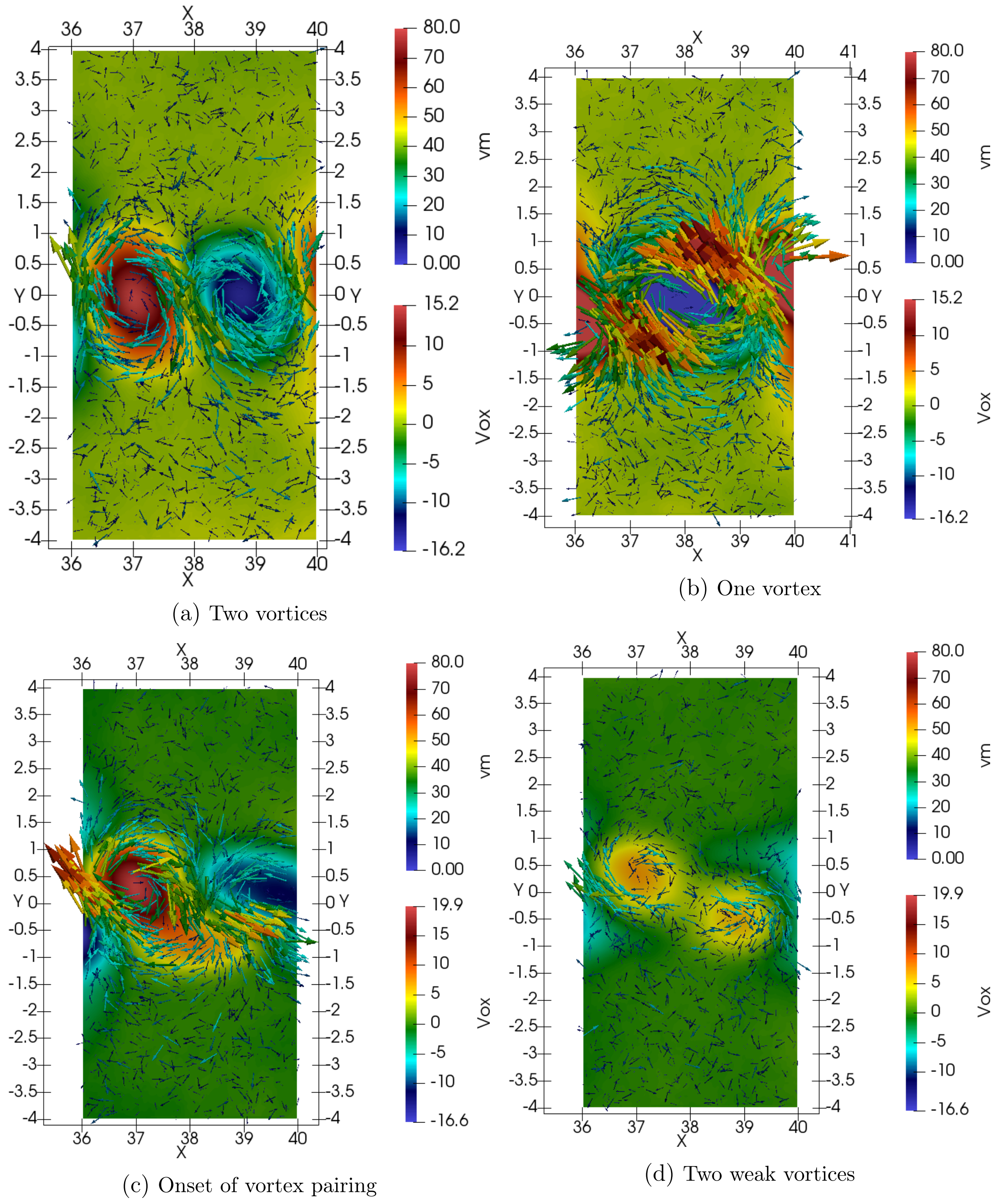
$$L_i = \frac{1}{\mathcal{N}} \sum_{j\in \mathrm{Nei}(i,\delta)} \left[u'_j(y_j - y_i) - v'_j(x_j - x_i)\right], \qquad u'_j \equiv u_j - U_j, \quad v'_j \equiv v_j - V_j, \tag{8}$$

where x_i, y_i and x_j, y_j are the coordinates of cell centers i and j, respectively, $\mathrm{Nei}(i,\delta)$ is the list of cells in the δ-vicinity of the cell i, i.e. $|\mathbf{r}_j - \mathbf{r}_i| \equiv \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} < \delta$, $\mathcal{N}$ is the number of cells in the list, u_j, v_j and u'_j, v'_j are the velocity components and their perturbations in the center of the cell j for this frame, U_j, V_j are the velocity components at the same cell averaged over all frames. Thus, the velocity disturbance flowfield imposed on the mean flow is analyzed. It is easy to see that Eq. (8) is the z-component of a vector product including the vectors of perturbed velocity and relative position of the cell centers i and j; for a 2D flow only this z-component is of importance. The parameter δ can be varied, in the present paper it is taken equal to 1 m.

The sign of L determines if the vortex rotates in the clockwise, or counterclockwise direction. The definition ensures that in the center of a vortex the value of L will reach a local maximum. Another feature of this quantity that it grows as the velocity of fluid rotating around the point x_i, y_i increases.

The resulting flowfields of L along with the original vector fields of the perturbed velocity $\mathbf{u}' \equiv (u', v')$ in the region $x > 36$ m near the outflow boundary are shown in Fig. 8 for 4 different time moments. The color palette **Vox** shows the vortex intensity L, while the vector length and their color in accordance with the scale **vm** show the magnitude of the perturbed velocity $|\mathbf{u}'|$. The frames presented are chosen in such a way that they demonstrate different dynamic events and phases of vortex motion. It is evident that they enable us to identify and recognize characteristic flow features despite the background statistical fluctuations.

The obtained flowfields of L are further post-processed. For each frame k the cells with maximum and minimum values of L are found. They are considered centers of vortices rotating in the clockwise and counterclockwise direction, respectively, because, in our case, the maximum value is always positive while the minimum value is negative. After that, the mean values of y-coordinates of vortex centers for both these types of vortices are deduced by averaging over n_f frames: $\overline{Y} = \sum_{k=1}^{n_f} Y^{(k)}/n_f$ where $Y^{(k}$ is the y-coordinate of the vortex center. The respective standard deviations $\sigma_Y = \sqrt{\sum_{k=1}^{n_f} (Y^{(k)} - \overline{Y})^2/n_f}$. The results are shown in Tab. 1. As one can see, on the average, the centers of counterclockwise vortices are located lower than those of clockwise vortices. Taking into account that the mixing layer centerline is somewhere between $y = -0.078$ m and $y = -0.085$ m, the intuitive description of the process would be that the



(a) Two vortices

(b) One vortex

(c) Onset of vortex pairing

(d) Two weak vortices

Figure 8. Vortices identified with the new criterion. Flowfields of L (color scale **Vox**) with imposed perturbed velocity vectors (arrows, their length and color, in accordance with color scale **vm**, correspond to the magnitude of perturbed velocity)

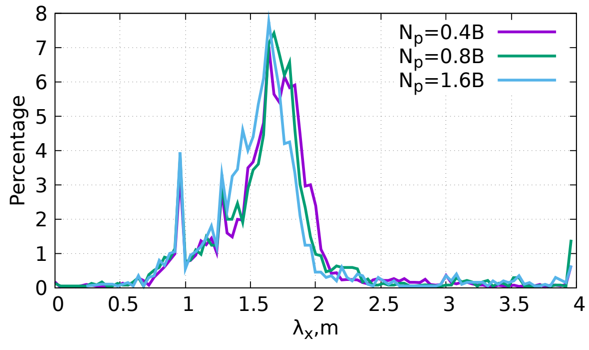
counterclockwise vortices move along the flow centerline with clockwise vortices slight above them.

The range 36 m $< x <$ 40 m (see Fig. 8) is chosen in such a way that for each frame it contains two vortices, one rotating clockwise and another – counterclockwise. The distribution of λ_x, the distances along x between vortex centers, is shown for all frames in Fig. 9a. The distances

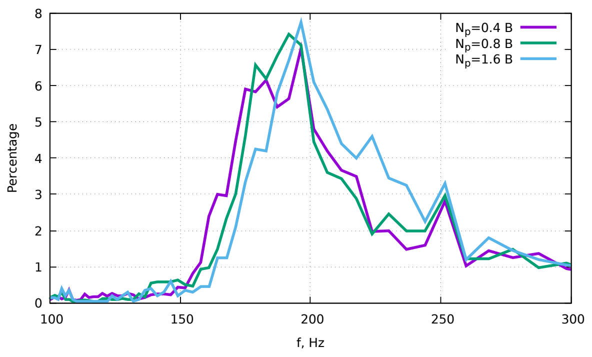
Table 1. Averaged positions of clockwise and counterclockwise vortices

N_p, 10^9	Clockwise		Counterclockwise	
	$\overline{Y}$, m	σ_Y, m	$\overline{Y}$, m	σ_Y, m
0.4	−0.086	0.29	−0.042	0.31
0.8	−0.076	0.31	−0.044	0.33
1.6	−0.077	0.27	−0.029	0.29

of 1.6–1.8 m dominate. The results of numerical simulations with different numbers of particles are quite similar. In all simulations there is a suspicious peak at $\lambda_x = 0.9$ m; it is possible that the peak is connected with an unknown numerical artifact. If one supposes that the vortices convect downstream with the average velocity $U_c = (U_1 + U_2)/2 = 644$ m/s, then the frequency of occurrence of vortices rotating in the same direction is $f = U_c/(2\lambda_x)$. The percentage of vortices as a function of the frequency is shown in Fig. 9b. The most probable frequencies are 170–200 Hz.



(a)



(b)

Figure 9. Distribution of distances along x between vortex centers (a) and distribution of frequency of vortex occurrence (b)

It can be concluded that the number of test particles and, therefore, the level of statistical noise have no impact on such characteristics of vortices as the distance between them and the frequency of their occurrence. It looks reasonable because these quantities are connected with parameters of the KH instability. They are determined by the wavelength of the most unstable disturbance and its frequency, respectively, i. e. by quantities that depend on mean flow properties but not on the initial amplitude of disturbances. At the same time, the initial amplitude of disturbances should have impact on their amplitudes in successive cross-sections and, consequently, on the intensity of vortex motion.

The vortex intensity can be measured for each frame as a difference between the maximum and minimum values of L. Time evolution of this parameter calculated over the range 36 m $< x <$ 40 m in three simulations with different numbers of particles is shown in Fig. 10. As can be seen, most of the time the vortex intensity is the largest in the simulation with $N_p = 0.4$ bln, and the smallest – at $N_p = 1.6$ bln. The time averaged vortex intensity in this cross-section is equal to 21.5, 17.4 and 13.95 for $N_p = 0.4$ bln, 0.8 bln and 1.6 bln, respectively.

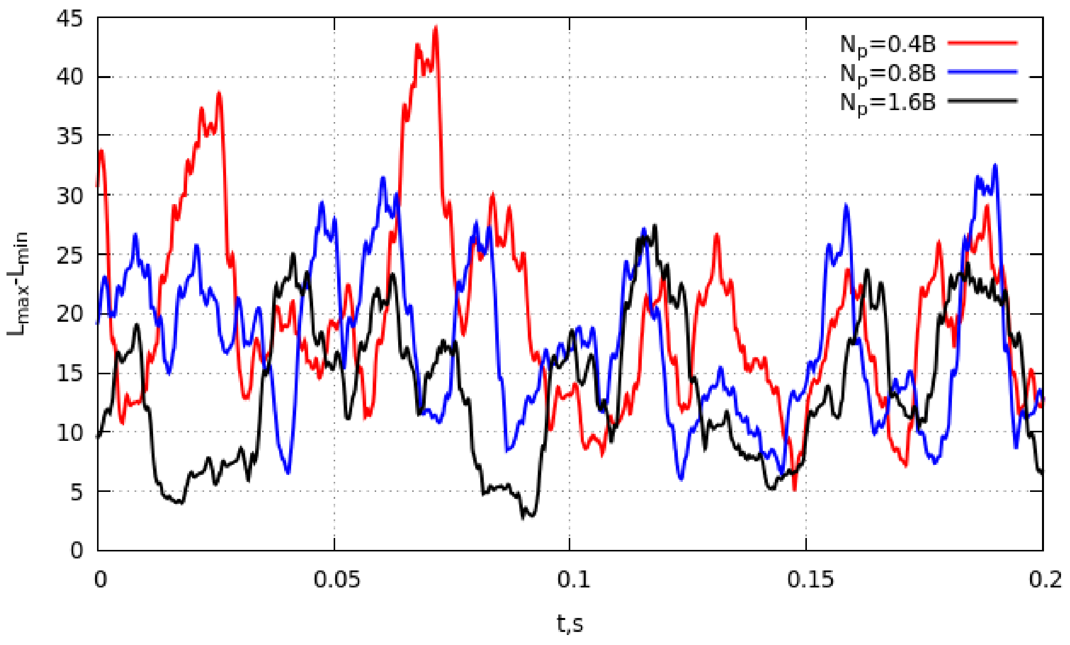


Figure 10. Time evolution of vortex intensity for different numbers of model particles

Conclusion

A numerical study of the instability developing in a supersonic mixing layer of monatomic gas (argon) is performed using the DSMC method, a particle-based method for solving the Boltzmann transport equation. The computations are carried out with the SMILE-GPU software system implemented on a hybrid multiple GPU/CPU computational cluster. No other flow excitation except statistical noise resulted from the stochastic DSMC method is used. A new algorithm for postprocessing and analyzing noisy computational flowfields is proposed and applied to extract data on vortex motion in the unstable mixing layer.

The main features of the Kelvin–Helmholtz instability such as vortex formation and successive vortex pairings are reproduced in the DSMC simulations. It is shown that the number of test particles used has no impact on instability parameters independent on the initial amplitudes of flow disturbances but can substantially change the pulsation characteristics and the intensity of vortex motion at a fixed distance from the mixing layer starting point.

Acknowledgments

This work was supported by the Russian Science Foundation, project No. 23-11-00258. All computations were performed using the equipment of the Equipment Sharing Center “Mechanics” of ITAM SB RAS.

We are also grateful to an anonymous reviewer whose remarks helped us noticeably improve the paper.

References

1. Bird, G.A.: Molecular Gas Dynamics and the Direct Simulation of Gas Flows, Clarendon Press, Oxford (1994).

2. Gallis, M.A., Koehler, T.P., Torczynski, J.R., Plimpton, S.J.: Direct simulation Monte Carlo investigation of the Richtmyer–Meshkov instability. Phys. Fluids 27(8), 084105 (2015). https://doi.org/10.1063/1.4928338

3. Gallis, M.A., Koehler, T.P., Torczynski, J.R., Plimpton, S.J.: Direct simulation Monte Carlo investigation of the Rayleigh–Taylor instability. Phys. Rev. Fluids 1(4), 043403 (2016). https://doi.org/10.1103/PhysRevFluids.1.043403

4. Kashkovsky, A.V., Kudryavtsev, A.N., Shershnev, A.A.: Investigation of Kelvin–Helmholtz instability with the DSMC method. AIP Conf. Proceedings 2125, 030028 (2019). https://doi.org/10.1063/1.5117410

5. Kashkovsky, A.V., Kudryavtsev, A.N., Shershnev, A.A.: DSMC simulation of instability of plane supersonic jet in coflow. AIP Conf. Proceedings 2504, 030087 (2023). https://doi.org/10.1063/5.0132264

6. Gallis, M.A., Bitter, N.P., Koehler, T.P., *et al.*: Molecular-level simulations of turbulence and its decay. Phys. Rev. Lett. 118(6), 064501 (2017). https://doi.org/10.1103/PhysRevLett.118.064501

7. Gallis, M.A., Torczynski, J.R., Bitter, N.P., *et al.*: Gas-kinetic simulation of sustained turbulence in minimal Couette flow. Phys. Rev. Fluids. 3(7), 071402 (2018). https://doi.org/10.1103/PhysRevFluids.3.071402

8. Gallis, M.A., Torczynski, J.R., Krygier, M.C., *et al.*: Turbulence at the edge of continuum. Phys. Rev. Fluids. 6(1), 013401 (2021). https://doi.org/10.1103/PhysRevFluids.6.013401

9. McMullen, R.M., Krygier, M.C., Torczynski, J.R., Gallis, M.A.: Navier–Stokes equations do not describe the smallest scales of turbulence in gases. Phys. Rev. Lett. 128(11), 11450 (2022). https://doi.org/10.1103/PhysRevLett.118.064501

10. Kashkovsky, A.V.: 3D DSMC computations on a heterogeneous CPU-GPU cluster with a large number of GPUs. AIP Conf. Proceedings 1628(1), 192–198 (2014). https://doi.org/10.1063/1.4902592

11. Betchov, R., Criminale W.O.: Stability of Parallel Flows, Academic Press, New York (1967).

12. Jackson, T.L., Grosch, C.E.: Inviscid spatial stability of a compressible mixing layer. J. Fluid Mech. 208, 609–637 (1989). https://doi.org/10.1017/S002211208900296X

13. Ragab, S.A., Wu, J.L.: Linear instabilities in two-dimensional compressible mixing layers. Phys. Fluids A 1(6), 957–966 (1989). https://doi.org/10.1063/1.857407

14. Kudryavtsev, A.N., Solov'ev, A.S.: Stability in the shear layer of compressible gas. J. Appl. Mech. Techn. Phys. 30(6), 949–956 (1989). https://doi.org/10.1007/BF00851504

15. Yang, J.-Y., Chang, J.-W.: Rarefied flow instability simulation using model Boltzmann equations. AIAA Paper No. 97-2017 (1997). https://doi.org/10.2514/6.1997-2017

16. Kudryavtsev, A.N., Poleshkin, S.O., Shershnev, A.A.: Numerical simulation of the instability development in a compressible mixing layer using kinetic and continuum approaches. AIP Conf. Proceedings 2125, 030033 (2019). `https://doi.org/10.1063/1.5117415`

17. Kotel'nikov, V.A.: On the transmission capacity of 'ether' and wire in electric communications. Physics–Uspekhi 49(7), 736–744 (2006). `https://doi.org/10.1070/PU2006v049n07ABEH006160`

18. Görtler, H.: Berechnung von Aufgaben der freien Turbulenz auf Grund eines neuen Näherungsansatzes. ZAMM 22(5), 244–254 (1942). `https://doi.org/10.1002/zamm.19420220503`

19. Ting, L.: On the mixing of two parallel streams. J. Math. Phys. 38, 153–165 (1959). `https://doi.org/10.1002/sapm1959381153`

20. Winant, C.D., Browand, F.K: Vortex pairing: the mechanism of turbulent mixing layer growth at moderate Reynolds number. J. Fluid Mech. 63(2), 237–255 (1974). `https://doi.org/10.1017/S0022112074001121`

21. Brown, G.L., Roshko, A.: On density effects and large structure in turbulent mixing layers. J. Fluid Mech. 64(4), 775–816 (1974). `https:/doi.org/10.1017/S002211207400190X`

22. Gnuplot homepage. `http://www.gnuplot.info`