
Acceleration of NOISEtte Code for Scale-Resolving Supercomputer Simulations of Turbulent Flows

A. V. Gorobets^{1*}, P. A. Bakhvalov^{1**}, A. P. Duben^{1***}, and P. V. Rodionov^{1****}

(Submitted by E. E. Tyrtshnikov)

¹*Keldysh Institute of Applied Mathematics of Russian Academy of Sciences, Moscow, 125047 Russia*

Received February 28, 2020; revised April 5, 2020; accepted April 20, 2020

Abstract—The present work is devoted to accelerating the NOISEtte code and lowering its memory consumption. This code for scale-resolving supercomputer simulations of compressible turbulent flows is based on higher-accuracy methods for unstructured mixed-element meshes and hierarchical MPI + OpenMP parallelization for cluster systems with manycore processors. We demonstrate modifications of the underlying numerical method and its parallel implementation, which consist, in particular, in using a simplified approximation method for viscous fluxes and mixed floating-point precision. The modified version has been tested on several representative cases. The performance measurements and validation results are presented.

DOI: 10.1134/S1995080220080077

Keywords and phrases: *turbulent flows, scale-resolving simulation, viscous fluxes, aerodynamics, aeroacoustics, parallel CFD, MPI + OpenMP.*

1. INTRODUCTION

The numerical solution of computational fluid dynamics (CFD) problems usually imposes rather high computing demands, especially in the case of scale-resolving supercomputer simulations. Such simulations, among other things, allow obtaining unsteady aerodynamic and aeroacoustic properties, as well as more accurately predict integral characteristics (lift, drag). A fine enough spatial mesh, a sufficiently small time step and a long enough time integration period (for accumulation of flow statistics) are required to resolve dynamics of turbulent structures correctly. All this leads to high computational costs.

The scientific computing community spends a lot of effort on accelerating and cheapening CFD simulations. The development of mathematical models and simulation approaches is aimed at obtaining accurate results on increasingly coarse meshes. For instance, hybrid approaches that efficiently combine Reynolds-averaged Navier–Stokes (RANS) and Large Eddy Simulation (LES) methods allow to significantly reduce mesh resolution requirements. Among such approaches, the non-zonal Detached Eddy Simulation (DES) methods are widely used, comprehensively investigated and validated. The most advanced versions of this method are formulated in [1, 2].

Another direction that leads to significant acceleration is the use of massively parallel computing accelerators, such as graphics processing units (GPUs). Representative examples of porting codes to the GPU architecture can be found, for instance, in [3, 4]. In [3], a rather mature and advanced GPU implementation for scale-resolving simulations using high-order schemes is compared with CPU implementations of well-known commercial CFD codes in terms of computing cost and accuracy. In [4], a GPU code for simulations of flows at high Mach numbers is presented.

* E-mail: andrey.gorobets@gmail.com

** E-mail: bahvalo@mail.ru

*** E-mail: dualks@gmail.com

**** E-mail: rodionov.cs@gmail.com

Another important direction of acceleration is the development of faster solvers for sparse systems of linear equations (SLAE), which often arise in CFD applications and in many cases consume most of the computing time (as in the case of the present work). An example of thereof can be found in [5], where a GPU implementation of a multigrid-based SLAE solver is connected to the OpenFOAM platform to speed up simulations.

Significant acceleration can be achieved by taking care of the data locality in memory. In [6], the so-called locally recursive non-locally asynchronous algorithm is used for the explicit discontinuous Galerkin method on structured grids in order to speed up computations by increasing the reuse of data in cache.

Other relevant examples of acceleration can be found in the following works. In [7], combined distributed and shared memory parallelization is implemented to accelerate simulations on cluster systems with Intel Xeon Phi manycore devices. The benefits of architecture-specific optimizations in a CFD code for scale-resolving simulations are demonstrated in [8] on the example of the Elbrus multicore processor architecture. Significant speedup can be achieved by improving a time integration method and its parallel implementation, as it was shown, for instance, in [9] for a Lower-Upper Symmetric Gauss–Seidel method.

In the present article, we accelerate the NOISEtte code for scale-resolving simulations of compressible turbulent flows. This code is based on higher-accuracy edge-based schemes [10, 11] for unstructured mixed-element meshes. It has multilevel MPI + OpenMP parallelization [13] for supercomputers based on manycore processors. Details about the data structure implementation that focuses on both performance and reliability are presented in [14].

The proposed modifications aimed at reducing computing cost and memory consumption consist, first, in a simplified approximation method for viscous fluxes and, second, in the use of mixed single and double floating-point precision. The changes mainly affect the part of the numerical algorithm that is associated with the computation of viscous fluxes, as well as the SLAE solver inside the Newton iterative process.

The rest of the paper is organized as follows. The mathematical model and numerical method are outlined in Section 2. Section 3 is devoted to the modification of viscous fluxes calculations. The changes in software implementation are presented in Section 4. The resulting performance gain is presented in Section 5. Validation of the proposed changes is demonstrated in Section 6 using several representative cases. Finally, conclusions are summarized in Section 7.

2. SUMMARY OF MATHEMATICAL MODEL AND NUMERICAL METHOD

2.1. Mathematical Model

The modeling of a turbulent, compressible flow is considered. The Navier–Stokes equations for a compressible ideal gas are used as a basic mathematical model:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla_j(\rho u_j) &= 0; \\ \frac{\partial \rho u_i}{\partial t} + \nabla_j(\rho u_i u_j + \delta_{ij} p - \tau_{ij}) &= 0; \\ \frac{\partial}{\partial t} \left(\frac{\rho u_k u_k}{2} + \frac{p}{\gamma - 1} \right) + \nabla_j \left(\left(\frac{\rho u_k u_k}{2} + \frac{\gamma p}{\gamma - 1} \right) u_j - \tau_{jk} u_k + q_j \right) &= 0. \end{aligned} \quad (1)$$

We assume the Einstein summation rule. The tension tensor $\tau = \{\tau_{ij}\}$ and the heat flux $\mathbf{q} = \{q_j\}$ are defined as

$$\begin{aligned} \tau_{ij} &= \mu \left(\nabla_i u_j + \nabla_j u_i - \frac{2}{3} \delta_{ij} \nabla_k u_k \right), \\ q_j &= -\frac{\gamma}{\gamma - 1} \frac{\mu}{\text{Pr}} \nabla_j \left(\frac{p}{\rho} \right). \end{aligned}$$

Turbulence is modeled using one of the following approaches: Direct Numerical Simulation (DNS), Reynolds-Averaged Navier–Stokes (RANS), including the basic ones [15, 16]; Large Eddy Simulation (LES), including [17, 18]; hybrid RANS-LES approaches, such as Detached Eddy Simulation (DES)[1].

2.2. Numerical Method

The set of higher-accuracy vertex-centered schemes [10, 11] based on quasi-one-dimensional reconstruction is used for finite-volume spatial discretization of the governing equations on an unstructured mesh. The extension to mixed-element meshes, which can contain tetrahedrons, quadrangular pyramids, triangular prisms and hexahedrons, is achieved by using direct control volumes [12]. The numerical flux through a face between two control volumes is computed using one of several Riemann problem solvers [19], depending on the flow conditions (the Roe scheme for subsonic flows, the HLLC for supersonic flows, among others).

Explicit Runge–Kutta schemes up to fourth order and the classical backward differentiation formula schemes BDF1 and BDF2 are used for discretization in time. Denote by Q^n the set of all conservative variables on the timestep t_n , then the scheme BDF1 can be represented in the form $(Q^{n+1} - Q^n)/\tau + \Phi(Q^{n+1}) = 0$, where Φ is a mesh approximation of the divergence of fluxes. To solve this system, the simplified Newton method is used:

$$\begin{aligned} \Delta^{(0)} = 0, \quad \Delta^{(s+1)} &= \Delta^{(s)} - \left(I + \frac{d\Phi}{dQ}(Q^n) \right)^{-1} \left(\frac{\Delta^{(s)}}{\tau} + \Phi(Q^n + \Delta^{(s)}) \right), \\ Q^{n+1} &= Q^n + \Delta^{(s)}. \end{aligned} \quad (2)$$

To solve the systems of linear algebraic equations (SLAE) with the Jacobi matrix, $I + \frac{d\Phi}{dQ}(Q^n)$, the preconditioned Bi-CGSTAB [20] method is used. The scheme BDF2 is treated similarly.

The set of boundary conditions for exterior faces consists of no-slip conditions for impermeable solid walls (isothermal, adiabatic, constant heat flux), Dirichlet's conditions, characteristic inflow/outflow conditions [21].

3. MODIFICATION OF THE NUMERICAL ALGORITHM

3.1. Calculation of Viscous Fluxes

The viscous and heat terms in Navier–Stokes equations (1) can be represented in a general form $D_{ij}u = \nabla_i(\mu \nabla_j u)$, where μ is a scalar coefficient, which can be dynamic viscosity or heat conductivity. For the vertex-centered schemes the most robust and popular method to approximate $D_{ij}u$ is the P1-Galerkin method.

Consider an unstructured mixed-element mesh. Let E be the set of mesh elements, N be the set of mesh nodes, $\Omega = \cup_{e \in E} e$ be the computational domain, $\partial\Omega$ be its boundary. Define the basis functions $\{\phi_G(\mathbf{r})\}$, where \mathbf{r} is a radius vector and $G \in N$ such that the following properties hold.

1. ϕ_G is a continuous function on Ω .
2. The support of ϕ_G coincide with the set of elements containing vertex G .
3. $\phi_G(\mathbf{r}_G) = 1$; $\phi_G(\mathbf{r}_K) = 0$ for $G \neq K$.
4. Any linear function can be exactly represented by a linear combination of the basis functions.

For a simplicial mesh these properties define the basis functions uniquely, otherwise it is possible to use different sets. To proceed, we do not need the exact specification.

Average $D_{ij}u$ over space with the weight ϕ_G :

$$\begin{aligned} (D_{ij}u)_G &\approx \frac{1}{\int \phi_G dV} \int_{\Omega} \nabla_i(\mu(\mathbf{r}) \nabla_j u) \phi_G dV \\ &= \frac{1}{\int_{\Omega} \phi_G dV} \left[\int_{\partial\Omega} n_i \mu(\mathbf{r}) (\nabla_j u) \phi_G dS - \int_{\Omega} \mu(\mathbf{r}) (\nabla_j u) (\nabla_i \phi_G) dV \right], \end{aligned}$$

where n_i are the components of the unit outward normal to the computational domain. Then substitute $u(\mathbf{r}) = u_G + \sum(u_K - u_G)\phi_K(\mathbf{r})$. We get

$$(D_{ij}u)_G \approx \frac{1}{\int_{\Omega} \phi_G dV} \left[B_G - \sum_K (u_K - u_G) \int_{\Omega} \mu(\mathbf{r})(\nabla_j \phi_K)(\nabla_i \phi_G) dV \right],$$

where

$$B_G = \sum_K (u_K - u_G) \int_{\partial\Omega} n_i \mu(\mathbf{r})(\nabla_j \phi_K) \phi_G dS.$$

The approximation we get is the mass-lumped Galerkin approximation of $D_{ij}u$. The terms in sums are nonzero only if there is an element containing nodes K and G . B_G is nonzero only if G is a boundary node. The approximation of B_G depends on the boundary conditions imposed, so we will not consider it later.

In solving Navier–Stokes equations the coefficient $\mu(\mathbf{r})$ in (4) is also a mesh function so the integrals need to be approximated. We can approximate $\mu(\mathbf{r})$ by a constant inside each mesh element, which is equal to the average of the nodal values at the vertices of the element. Then

$$(D_{ij}u)_G \approx \frac{1}{\int_{\Omega} \phi_G dV} \left[B_G - \sum_{K \in N} (u_K - u_G) \sum_{e \ni G, K} \mu_e \varkappa_{i,j}^{G,K,e} \right], \tag{3}$$

where the inner sum is over mesh elements e containing nodes G and K , and

$$\varkappa_{i,j}^{G,K,e} = \int_e (\nabla_j \phi_K)(\nabla_i \phi_G) dV.$$

Obviously, $\varkappa_{i,j}^{G,K,e} = \varkappa_{j,i}^{K,G,e}$. The coefficients $\varkappa_{i,j}^{G,K,e}$ can be computed once and then stored in memory.

In 3D calculation the storage of the coefficients $\varkappa_{i,j}^{G,K,e}$ takes a lot of memory and a lot of time to get them from memory at each time step. We can write the simplified approximation:

$$(D_{ij}u)_G \approx \frac{1}{\int_{\Omega} \phi_G dV} \left[B_G - \sum_K (u_K - u_G) \frac{\mu_G + \mu_K}{2} m_{i,j}^{G,K} \right], \tag{4}$$

where

$$m_{i,j}^{G,K} = \sum_{e \ni K, L} \mu_e = \int_{\Omega} (\nabla_j \phi_K)(\nabla_i \phi_G) dV.$$

In this case there are only 9 coefficients for one pair of nodes belonging to one element (obviously, $m_{i,j}^{G,K} = m_{j,i}^{K,G}$). Moreover, if K and G do not belong to the domain boundary, then

$$m_{i,j}^{G,K} = \int_{\Omega} (\nabla_j \phi_K)(\nabla_i \phi_G) dV = - \int_{\Omega} (\nabla_i \nabla_j \phi_K)(\phi_G) dV = \int_{\Omega} (\nabla_i \phi_K)(\nabla_j \phi_G) dV = m_{j,i}^{G,K}$$

so we can store only 6 coefficients exploiting this symmetry. The test cases for the low-Reynolds problems shows that this method can provide a wrong solution if μ varies in space (for example, shock wave profile with the constant kinematic viscosity). However, for convection-dominant problems with moderate Mach numbers the solution is admissible. This will be shown in Section 6.

In summary, the transition from the original P1-Galerkin method to the simplified formula (4) eliminates the double summation in (3), which results in at least 3 times lower memory requirements and computing costs in the three-dimensional case.

4. MODIFICATION OF THE SOFTWARE IMPLEMENTATION

4.1. Using Mixed Floating-Point Precision for Viscous Fluxes

We use an adjacency graph to deal with the coefficients $\alpha_{i,j}^{G,K,e}$ or $m_{i,j}^{G,K}$ needed to compute (3) or (4), respectively. The vertices of this graph correspond to mesh nodes. A pair of vertices is connected by an edge if there is a mesh element that includes the two corresponding nodes. We store the coefficients in blocks for each edge of this adjacency graph. The block size is variable in the original version, because for each edge it depends on the number of mesh elements that include its pair of nodes. In the modified version, the block size is fixed, 6 real values per edge.

We use special containers for block data [14], which have a plain pointer in the backend and provide an access interface similar to a two-dimensional array. In the case of a variable block size, the pointer is supplemented with a compressed sparse row (CSR) representation for positioning. In a loop over the set of graph's edges, the contribution of each edge is computed and added into its two incident nodes in order to obtain values of (3) or (4) in each node.

The coefficients $\alpha_{i,j}^{G,K,e}$ or $m_{i,j}^{G,K}$ can be downshifted to the single-precision floating-point format without significant loss of accuracy, which will be proved in Section 6. By doing so, the memory consumption and memory traffic are reduced by almost half.

Let us summarize memory costs. In the case of an unstructured tetrahedral mesh, the array of coefficient $\alpha_{i,j}^{G,K,e}$ for the original version (3) consumes 2.5 KB per mesh node. The simplified version (4) needs about 0.34 KB per mesh node to store $m_{i,j}^{G,K}$, 7.5 times less. In the case of a hexahedral mesh, the original version consumes 2 KB per node, while the modified version needs about 0.62 KB per node (3.2 times less). The above memory costs correspond to the double-precision floating-point format. If we use the single-precision format for these coefficients, the memory size is accordingly halved. Further details about memory consumption can be found in the following section in Table 2.

Finally, it should be noted that the modified version (4) is much better suited for the GPU architecture for the reason explained below. In the CPU version, the viscous fluxes are computed in a loop over the edges of the corresponding adjacency graph. Therefore, processing one edge can be an elementary task in the stream processing paradigm. The GPU computing subroutine (the so-called kernel) for the viscous fluxes calculates the contribution of a given edge and adds it to its nodes. In order to eliminate the data dependency (race condition) when summing into nodes, we can either use an intermediate storage for subsequent summation (as explained in [22], subsection 4.3.1), or graph coloring that splits the set of edges into subsets that have no edges with a common node.

In the original version, the second sum in (3) leads to the presence of a variable-sized loop inside the kernel, which significantly slows down the computations. This loop causes divergence of threads running simultaneously in one stream multiprocessor. There is no such a loop in the modified version, therefore, it is not only less computationally expensive, but also more efficient on the GPU.

4.2. Using the Single-Precision Floating-Point Format for the Jacobi System

In the case of a Newton-based implicit scheme, the tensor coefficients for viscous fluxes become the second biggest memory consumer, while the Jacobi matrix takes the first place. The solution of the SLAE with the Jacobi matrix at each iteration is needed to direct Newton's process towards convergence by finding the correction $\Delta^{(s)}$ in (2) to the mesh functions of the conservative variables. To solve this system, we use an iterative method (the preconditioned Bi-CGSTAB [20]) with a relatively coarse residual criterion. In our practical applications, the solver tolerance (the ratio of the residual vector norm to the norm of the right-hand side vector) is usually in the range from about 10^{-3} to 10^{-2} . This level of accuracy is sufficient, which means that a further increase in accuracy does not affect the convergence of Newton's process. Taking this into account, we can expect that solving the Jacobi system using single precision is sufficient. In this case, the correction obtained in single precision is converted into double precision and applied to the mesh functions. By doing so, Newton's process remains unchanged and is carried out in double precision. Therefore, using single precision in the SLAE solver should not affect the results of the numerical simulation if the following conditions are satisfied: 1) the solver succeeds to find the solution with the requested accuracy; 2) the convergence of Newton's process is not broken.

Table 1. Timing results (per one time step) for the original (3) and reduced-cost (4) versions of viscous fluxes using DNS approach, the EBR5 scheme [10], the explicit 4-th order Runge-Kutta and implicit 2-nd order Newton-based integration schemes, mixed (single precision for the “viscous” coefficients $\chi_{i,j}^{G,K,e}$, $m_{i,j}^{G,K}$ and the Jacobi matrix) and double floating-point precision, a hexahedral mesh of 1.5 million nodes and a tetrahedral one of 1 million nodes

Version	Precision	Mesh, nodes	Overall time, s		Viscosity time, s		Solver time, s
			explicit	implicit	explicit	implicit	implicit
Basic	Double	hexa, 1.5×10^6	0.742	2.62	0.287	0.457	1.70
Reduced	Double	hexa, 1.5×10^6	0.567	2.53	0.126	0.367	1.71
Basic	Mixed	hexa, 1.5×10^6	0.739	1.69	0.281	0.380	0.90
Reduced	Mixed	hexa, 1.5×10^6	0.563	1.59	0.123	0.283	0.90
Basic	Double	tetra, 10^6	0.839	1.85	0.305	0.297	1.10
Reduced	Double	tetra, 10^6	0.568	1.71	0.075	0.166	1.11
Basic	Mixed	tetra, 10^6	0.806	1.29	0.269	0.251	0.61
Reduced	Mixed	tetra, 10^6	0.566	1.15	0.071	0.136	0.62

To verify this, we tested the use of the single-precision floating-point format for the Jacobi matrix (stored in a specialized CSR-based block format) and the SLAE solver in various representative simulations. It appeared that there is no affect on the convergence of the Newton process in our target applications, which are scale-resolving simulations with a time step limited by the necessary resolution of turbulent dynamics. The choice of the floating-point precision for the solver is implemented at compile time with a preprocessor definition. This modification halved the memory requirements for the Jacobi system and significantly accelerated the solver.

5. PERFORMANCE BENEFITS

Since there have been no changes in MPI parallelization, performance is measured on a single computing node. The node has two 16-core Intel Xeon E5-2683 v4 CPUs (2.1 GHz, 77 GB/s in 4 DDR4 channels, 40 MB L3 cache). Detailed information on the parallel performance of the NOISEtte code on various supercomputers can be found in [1], where scalability of up to 10 thousand cores is demonstrated.

Two test cases have been studied: a round jet flow using a hexahedral mesh of 1.5 million nodes (described in the following section) and flow around a sphere using a tetrahedral mesh of 1 million nodes. The parallel execution configuration used in both cases is two MPI processes (one per CPU) with 16 threads each. Performance results are presented in Table 1 in terms of wall clock time. Comparison in terms of memory consumption is given in Table 2.

The viscous fluxes calculation subroutine differs for explicit and implicit time integration schemes. The latter includes the calculation of the coefficients of the Jacobi matrix (two diagonal blocks and two off-diagonal blocks of size 5x5 per edge). Therefore, execution times have been measured for both the explicit 4th order Runge-Kutta and 2nd order Newton-based implicit schemes. Timing results for the basic (3) and reduced-cost (4) versions are given for mixed and double floating-point precision. In the mixed-precision version, the single-precision format is used only for the “viscous” coefficients $\chi_{i,j}^{G,K,e}$, $m_{i,j}^{G,K}$ and the Jacobi matrix.

The results in Table 1 show that the arithmetic intensity of the viscous fluxes calculation is sufficient to hide the memory traffic when reading the coefficients in the case of an explicit scheme, since there is almost no difference in time between the mixed and double precision versions. Comparison of results for tetrahedral and hexahedral meshes shows that the new version gives bigger benefits in tetrahedral zones.

Table 2. Memory consumption for the basic (3) and reduced-cost (4) versions of viscous fluxes using DNS approach, the EBR5 scheme [10], the explicit 4th order Runge–Kutta and implicit 2nd order Newton-based time integration schemes, mixed (single precision for the “viscous” coefficients $\mu_{i,j}^{G,K,e}$, $m_{i,j}^{G,K}$ and the Jacobi matrix) and double floating–point precision, a hexahedral mesh of 1.5 million nodes and a tetrahedral one of 1 million nodes

Version	Precision	Mesh, nodes	Total memory, GB		Storage for coefficients, GB
			explicit	implicit	
Basic	Double	hexa, 1.5×10^6	9.8	18.0	3.03
Reduced	Double	hexa, 1.5×10^6	7.7	15.9	0.94
Basic	Mixed	hexa, 1.5×10^6	8.6	12.4	1.52
Reduced	Mixed	hexa, 1.5×10^6	7.6	11.4	0.47
Basic	Double	tetra, 10^6	5.8	8.6	2.5
Reduced	Double	tetra, 10^6	3.6	6.4	0.33
Basic	Mixed	tetra, 10^6	4.6	5.9	1.25
Reduced	Mixed	tetra, 10^6	3.5	4.8	0.17

Table 3. of drag and lift coefficients for NACA0012 case

	$C_{d,p}$	$C_{d,visc}$	$C_{l,p}$	$C_{l,visc}$
Basic version	0.00625	0.0061626	1.0883	9.32e-5
Reduced-cost version	0.00625	0.0061676	1.0894	9.33e-5
Difference, %	0	0.08	0.10	0.11

In summary, in the case of an explicit scheme, the accelerated version of the viscous fluxes calculation outperforms the baseline version by a factor of $\times 2.3$ for the hexahedral mesh and $\times 4.3$ for the tetrahedral mesh. In the “implicit” case, these factors are smaller, $\times 1.6$ and $\times 2.2$, respectively, because filling in the Jacobi matrix takes a significant fraction of time. In terms of memory costs, the storage for the coefficients in the new version consumes 6.4 and 15 times less for tetrahedral and hexahedral meshes, respectively.

6. VALIDATION OF MODIFICATIONS, APPLICATION EXAMPLES

6.1. RANS Simulations of Flow Around the Model NACA0012 Profile

Firstly we study a subsonic flow around a model airfoil section. In the experimental conditions, the infinite NACA0012 profile with the chord length c was placed inside the free stream fully turbulent flow characterized by Mach number $M = 0.15$ and Reynolds number $\text{Re}_c = 6 \times 10^6$ at the angle of attack (AOA) 10° [23]. The two-dimensional computational domain (Fig. 1) is represented by the square with $-500 < x/c, y/c < 500$. The corresponding mixed-element mesh consists of 83 thousand nodes (442 nodes at the body surface) and is formed by quadrilaterals near airfoil surrounded by an unstructured triangular region. At the body surface, no-slip adiabatic boundary conditions were employed. Hereinafter, we use the Spalart–Allmaras turbulence model as the closure for the RANS approach. The results shown in Fig. 2 and Table 3 confirm that the results of the baseline and modified versions are identical.

6.2. RANS Simulations of Flow Around the Model RAE2822 Profile

Next, we study a transonic flow with the presence of shocks, namely, the fully turbulent flow at $M = 0.729$ and $\text{Re}_c = 6.5 \times 10^6$ around the RAE2822 profile with the chord length c at AOA 2.31° [24].

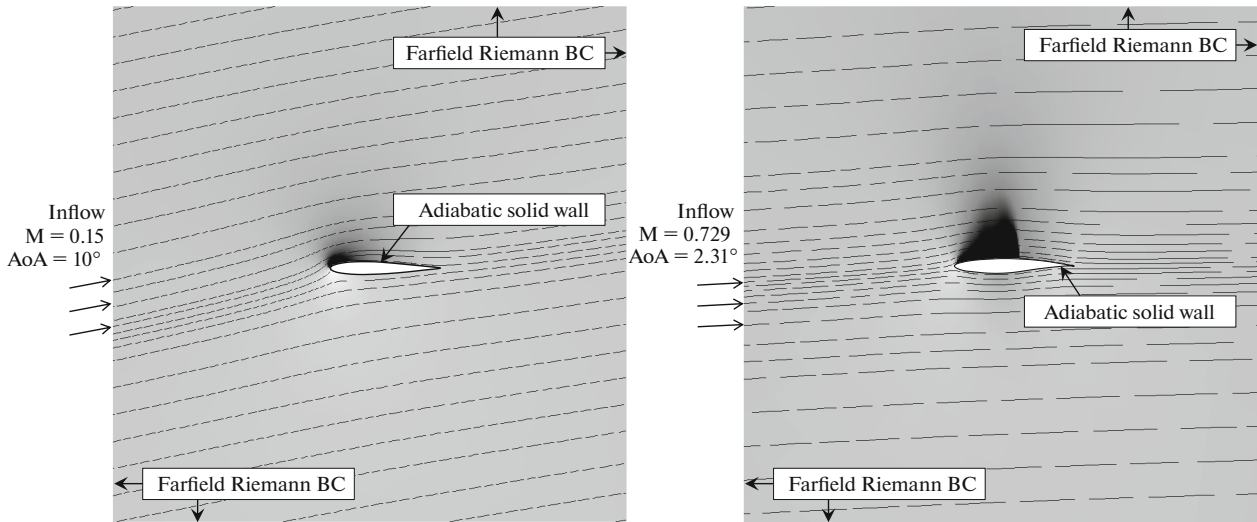


Fig. 1. Scheme of the computational domain (not in scale) for NACA0012 (left) and RAE2822 (right) model problems, rendered using numerical solution (pressure field and streamlines).

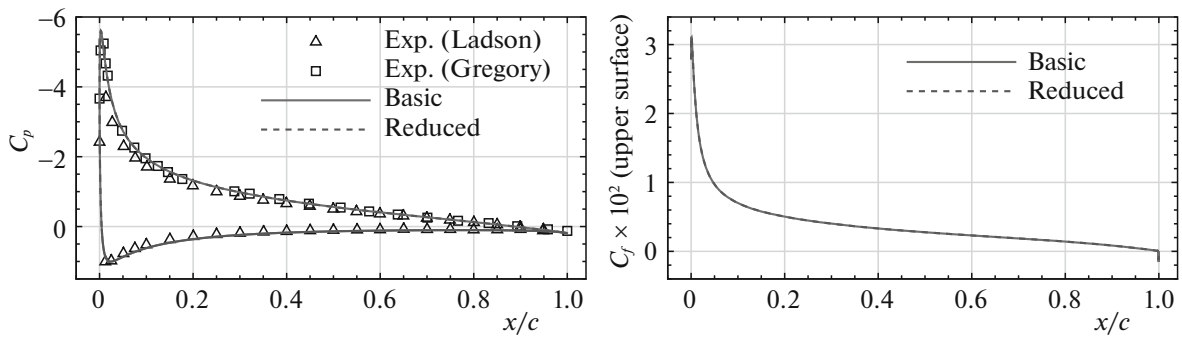


Fig. 2. Comparison of pressure and friction coefficients for NACA0012 case.

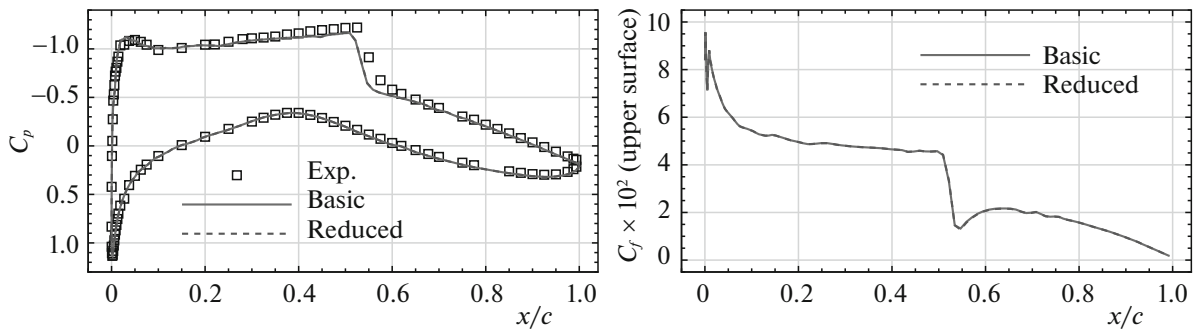


Fig. 3. Comparison of pressure and friction coefficients for RAE2822 case.

The corresponding two-dimensional computational domain (Fig. 1) is defined by the structured mesh of 24 thousand nodes (369×65) in the $-20 < x/c, y/c < 20$ area. As in previous case, we use no-slip adiabatic boundary conditions at the solid surface. The results shown in Fig. 3 and Table 4 lead us to the same conclusion, as in the previous case.

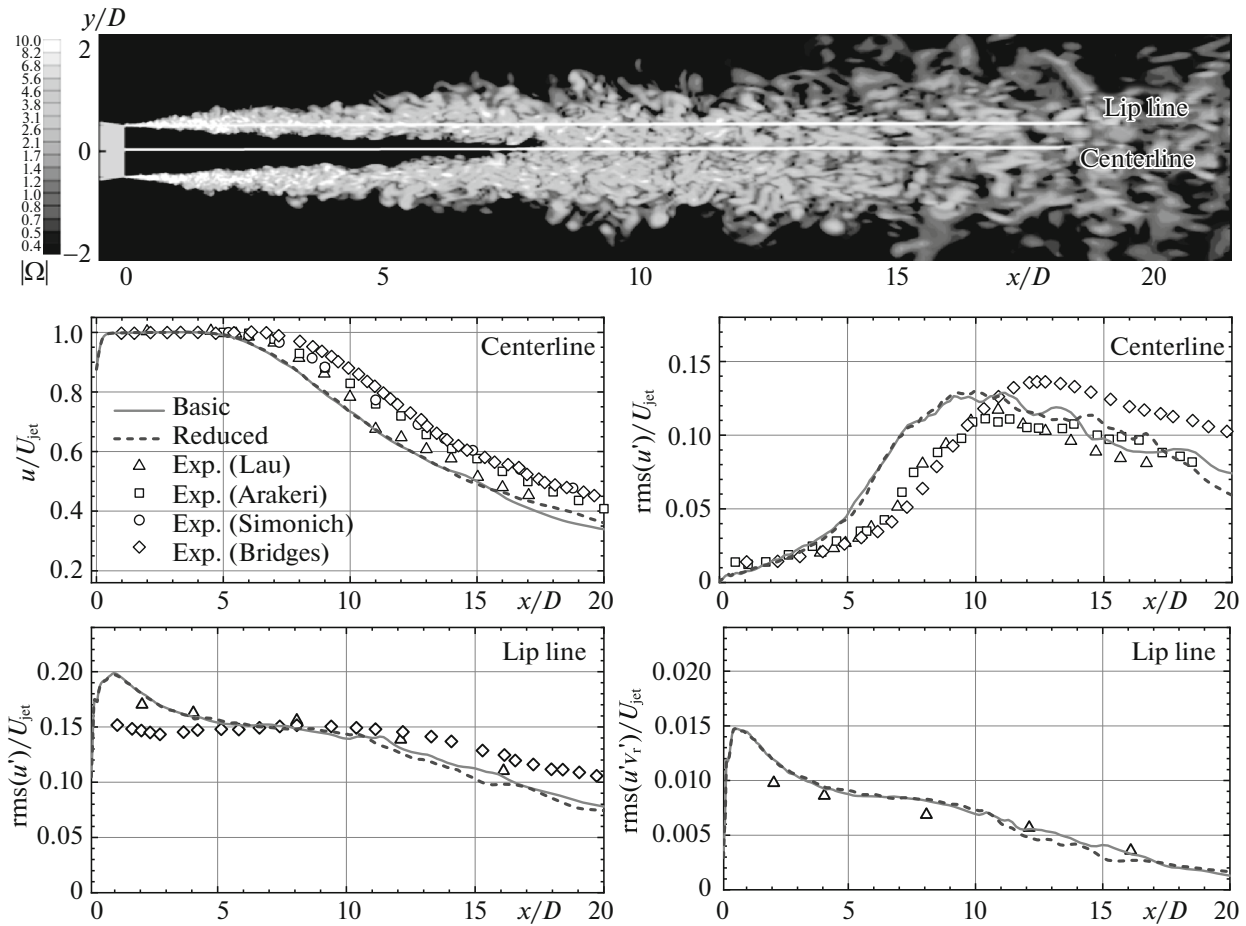


Fig. 4. View of the instantaneous flow field (vorticity magnitude) with indication of profile lines (top). Averaged distributions of the streamwise velocity and its root mean square (rms) values over the jet centerline (middle) and resolved Reynolds stresses over the lip line downstream the nozzle edge (bottom) obtained with the basic and reduced-cost versions.

6.3. Scale-Resolving Simulations

Finally, we take on three-dimensional problems. The immersed unheated turbulent jet exiting from a conical nozzle at $M_{jet} = 0.9$ and $Re_D = 1.1 \times 10^6$ (based on the jet diameter D and the jet exit velocity U_{jet}) is considered. The flow dynamics is similar to the jets which had been investigated experimentally by several authors in the literature [25–28]. The computational setup, including the mesh and boundary conditions, is inherited from the study carried out by Shur et al. [29]. This case was used in various numerical studies by those [1, 30] and another authors [31, 32]. The simulation of the jet follows the two-stages approach [33]. The nozzle and jet-plume simulation is performed using RANS at the first stage. Then, only the jet-plume region is considered at the second stage, imposing at the nozzle exit the profiles obtained at the first stage. These profiles of gas-dynamic and turbulence model variables were provided by M. Shur and M. Strelets from Peter the Great St. Petersburg Polytechnic University. The structured hexahedral Grid 1 of 1.52 million nodes (64 cells in the azimuthal direction) from [29, 32], the coarser one, has been used in the present work. A more detailed description of this numerical study, including demonstration of mesh convergence, can be found in [32].

The results of the jet case computed using both the baseline and modified (new viscous fluxes with single-precision coefficients) versions are presented in Fig. 4. It contains distributions of streamwise velocity and resolved Reynolds stresses over the jet axis (centerline) and lip line downstream the nozzle exit, respectively. Figure 4 demonstrates that the new realization does not lead to any perceptible discrepancy of the solution compared to the baseline. The differences between the two versions that can only be observed in the far region (beyond $x/D = 10$), are related with the finite time integration

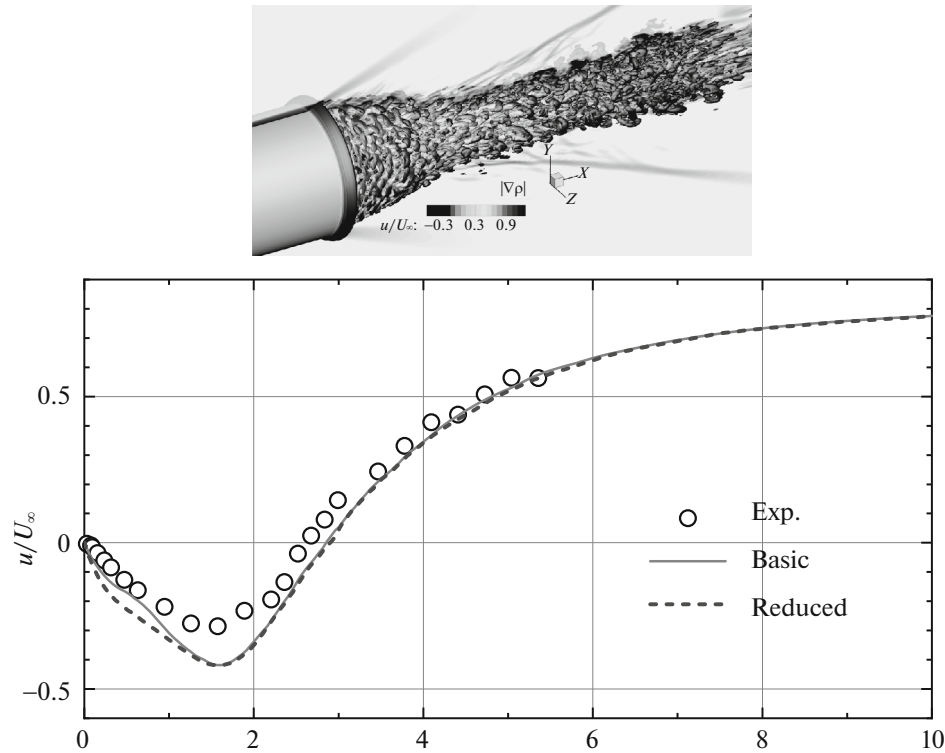


Fig. 5. Supersonic flow behind the cylinder base: an instantaneous view using isosurfaces of Q criterion and numerical Schlieren in the transverse section (left), and averaged distributions of the streamwise velocity over the cylinder axis (centerline) starting from its base (right) obtained with the basic and reduced-cost versions.

period ($250D/U_{\text{jet}}$). The results fit reasonably well with the experimental data for such a coarse mesh used for the sake of saving computing time (our results on finer meshes are presented in [32]).

The second problem is supersonic turbulent flow at $M_\infty = 2.46$ over the cylinder base which was investigated experimentally by Herrin&Dutton [34]. Reynolds number based on the cylinder radius R is $Re_R = 1.63 \times 10^6$.

The mesh of 3.55 million nodes (128 cells in the azimuthal direction) covers the region around the cylinder $-4 \leq x/R < 12$ ($x/R = 0$ corresponds to the cylinder base) in the streamwise direction and $-9 \leq y/R, z/R < 9$ in the azimuthal direction. The pre-computed profiles of gas-dynamic and turbulent characteristics were imposed at the inflow surface ($x/R = -4$) to reach the same boundary layer thickness at $x/R = -0.0315$, as in the experiment. The recent version of the DDES approach [1] and the hybrid adapting numerical scheme [32] were used for the simulations. An instantaneous view of the flow is shown in Fig. 5 (left). It is characterized by the shocks, which interfere with the shear layer developing from the cylinder edges.

The results of simulations shown in Fig. 5 for both the basic and reduced-cost algorithms are in good agreement with each other and with the corresponding experimental data.

Table 4. Components of drag and lift coefficients for RAE2822 case

	$C_{d,p}$	$C_{d,visc}$	$C_{l,p}$	$C_{l,visc}$
Basic version	0.00670	0.00623	0.7197	2.2e-5
Reduced-cost version	0.00672	0.00623	0.7206	2.2e-5
Difference, %	0.30	0	0.13	0

7. CONCLUSIONS

Modifications of the CFD algorithm and its parallel implementation for supercomputer simulations of compressible turbulent flows have been presented. The changes consist in reducing computational costs and memory requirements at the stage of calculation of viscous fluxes by using a simplified approximation method and mixed floating-point precision. In the case of an implicit scheme, the solution stage of the Jacobi system has been accelerated as well. The resulting acceleration of the viscous fluxes calculation stage ranges from 2 to 4 times, depending on the mesh kind. The overall acceleration up to $\times 1.6$ has been achieved. In terms of memory costs, the size of the storage for the coefficients needed to compute viscous fluxes has been reduced by 6 and 15 times for tetrahedral and hexahedral mesh zones, respectively. The modified version has been validated in several representative cases, and no differences in the accuracy of results have been observed compared to the original version.

ACKNOWLEDGMENTS

The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University [35] supercomputer simulations), and using the equipment of the collective use center of the Keldysh Institute of Applied Mathematics (performance tests). The authors thankfully acknowledge these institutions.

FUNDING

This work has been funded by the Russian Science Foundation, project 19-11-00299.

REFERENCES

1. M. Shur, P. Spalart, M. Strelets, and A. Travin, "An enhanced version of DES with rapid transition from RANS to LES in separated flows," *Flow, Turbulence Combust.* **95**, 709–737 (2015).
2. E. K. Guseva, A. V. Garbaruk, and M. K. Strelets, "Assessment of delayed DES and improved delayed DES combined with a shear-layer-adapted subgrid length-scale in separated flows," *Flow, Turbulence Combust.* **98**, 481–502 (2017).
3. B. C. Vermeire, F. D. Witherden, and P. E. Vincent, "On the utility of GPU accelerated high-order methods for unsteady flow simulations: A comparison with industry-standard tools," *J. Comput. Phys.* **334**, 497–521 (2017).
4. A. N. Bocharov, N. M. Evstigneev, V. P. Petrovskiy, O. I. Ryabkov, and I. O. Teplyakov, "Implicit method for the solution of supersonic and hypersonic 3D flow problems with lower-upper symmetric-Gauss-Seidel preconditioner on multiple graphics processing units," *J. Comput. Phys.* **406**, 109–189 (2020).
5. B. Krasnopolsky and A. Medvedev, "Acceleration of large scale OpenFOAM simulations on distributed systems with multicore CPUs and GPUs," *Adv. Parallel Comput.* **27**, 93–102 (2016).
6. B. A. Korneev and V. D. Levchenko, "Effective solving of three-dimensional gas dynamics problems with the Runge–Kutta discontinuous Galerkin method," *Comput. Math. Math. Phys.* **56**, 460–469 (2016).
7. V. A. Titarev, "Application of model kinetic equations to hypersonic rarefied gas flows," *Comput. Fluids* **169**, 62–70 (2018).
8. A. V. Gorobets, M. I. Neiman-Zade, S. K. Okunev, A. A. Kalyakin, and S. A. Soukov, "Performance of Elbrus-8C processor in supercomputer CFD simulations," *Math. Models Comput. Simul.* **11**, 914–923 (2019).
9. A. Chikitkin, M. Petrov, V. Titarev, and S. Utyuzhnikov, "Parallel versions of implicit LU-SGS method," *Lobachevskii J. Math.* **39**, 503–512 (2018).
10. P. A. Bakhvalov, I. V. Abalakin, and T. K. Kozubskaya, "Edge-based reconstruction schemes for unstructured tetrahedral meshes," *Int. J. Numer. Methods* **81**, 331–356 (2016).
11. P. A. Bakhvalov and T. K. Kozubskaya, "EBR-WENO scheme for solving gas dynamics problems with discontinuities on unstructured meshes," *Comput. Fluids* **169**, 98–110 (2018).
12. P. A. Bakhvalov and T. K. Kozubskaya, "Construction of edge-based 1-exact schemes for solving the Euler equations on hybrid unstructured meshes," *Comp. Math. Math. Phys.* **57**, 680–697 (2017).
13. A. Gorobets, "Parallel algorithm of the NOISEtte code for CFD and CAA simulations," *Lobachevskii J. Math.* **39** (4), 524–532 (2018).
14. A. V. Gorobets and P. A. Bakhvalov, "Improving reliability of supercomputer CFD codes on unstructured meshes," *Supercomput. Front. Innov.*, No. 6, 44–56 (2019).
15. P. R. Spalart and S. R. Allmaras, "A one-equation turbulence model for aerodynamic flows," in *Proceedings of the 30th Aerospace Science Meeting, Reno, Nevada, May 20–22, 1992*, AIAA Paper 92-0439.

16. F. R. Menter, “Two-equation eddy-viscosity turbulence models for engineering applications,” *AIAA J.* **32**, 1598–1605 (1994).
17. F. Nicoud and F. Ducros, “Subgrid-scale stress modelling based on the square of the velocity gradient tensor,” *Flow, Turbulence Combust.* **62**, 183–200 (1999).
18. F. X. Trias, D. Folch, A. Gorobets, and A. Oliva, “Building proper invariants for eddy-viscosity subgrid-scale models,” *Phys. Fluids* **27**, 065103 (2015). <https://doi.org/10.1063/1.4921817>
19. E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics* (Springer, Berlin, Heidelberg, 2009). <https://doi.org/10.1007/b79761>
20. H. A. van der Vorst, “Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems,” *SIAM J. Sci. Stat. Comput.*, No. 13, 631–644 (1992). <https://doi.org/10.1137/0913035>
21. Ch. Hirsch, *Numerical Computation of Internal and External Flows*, Vol. 2: *Computational Methods for Inviscid and Viscous Flows* (Wiley, New York, 1990).
22. A. V. Gorobets, “Parallel technology for numerical modeling of fluid dynamics problems by high-accuracy algorithms,” *Comput. Math. Math. Phys.* **55**, 638–649 (2015).
23. 2DN00: 2D NACA 0012 Airfoil Validation Case. https://turbmodels.larc.nasa.gov/naca0012_val.html. Accessed 2020.
24. RAE2822 Transonic Airfoil. <https://www.grc.nasa.gov/WWW/wind/valid/raetaf/raetaf.html>. Accessed 2020.
25. J. C. Lau, P. J. Morris, and M. J. Fisher, “Effects of exit Mach number and temperature on mean-flow and turbulence characteristics in round jets,” *J. Fluid Mech.* **105**, 193–218 (1981).
26. V. Arakeri, A. Krothapalli, V. Siddavaram, M. Alkisar, and L. Lourenco, “On the use of microjets to suppress turbulence in a Mach 0.9 axisymmetric jet,” *J. Fluid Mech.* **490**, 75–98 (2003).
27. C. Simonich, S. Narayanan, T. Barber, and M. Nishimura, “Aeroacoustic characterization, noise reduction and dimensional scaling effects of high subsonic jets,” *AIAA J.* **39**, 2062–2069 (2001).
28. J. Bridges and M. Wernet, “Establishing consensus turbulence statistics for hot subsonic jets,” *AIAA Paper No. 2010–3751* (AIAA, 2010).
29. M. Shur, P. Spalart, and M. Strelets, “LES-based evaluation of a microjet noise reduction concept in static and flight conditions,” *J. Sound Vibrat.* **330**, 4083–4097 (2011).
30. M. Shur, P. Spalart, and M. Strelets, “Jet noise computation based on enhanced DES formulations accelerating the RANS-to-LES transition in free shear layers,” *Int. J. Aeroacoust.* **15**, 595–613 (2016).
31. M. Fuchs, C. Mockett, M. Shur, M. Strelets, and J. C. Kok, “Single-stream round jet at $M=0.9$,” in *Go4Hybrid: Grey Area Mitigation for Hybrid RANS-LES Methods*, Vol. 134 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, Ed. by C. Mockett, W. Haase, and D. Schwamborn (Springer, Switzerland, 2018), pp. 125–137.
32. A. P. Duben and T. K. Kozubskaya, “Evaluation of quasi-one-dimensional unstructured method for jet noise prediction,” *AIAA J.* **57**, 5142–5155 (2019).
33. M. Shur, P. Spalart, M. Strelets, and A. Garbaruk, “Analysis of jet-noise-reduction concepts by large-eddy simulation,” *Int. J. Aeroacoust.* **6**, 243–285 (2007).
34. J. L. Herrin and J. C. Dutton, “Supersonic base flow experiments in the near wake of a cylindrical afterbody,” *AIAA J.* **32**, 77–83 (1994).
35. V. I. Voevodin, A. Antonov, D. Nikitenko, P. Shvets, S. Sobolev, I. Sidorov, K. Stefanov, V. Voevodin, and S. Zhumatiy, “Supercomputer Lomonosov-2: Large scale, deep monitoring and fine analytics for the user community,” *Supercomput. Front. Innov.*, No. 6, 4–11 (2019). <https://doi.org/10.14529/jsfi190201>