ISSN 0965-5425, Computational Mathematics and Mathematical Physics, 2016, Vol. 56, No. 9, pp. 1651–1664. © Pleiades Publishing, Ltd., 2016.
Original Russian Text © I.S. Menshov, P.V. Pavlukhin, 2016, published in Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 2016, Vol. 56, No. 9, pp. 1677–1691.

Efficient Parallel Shock-Capturing Method for Aerodynamics Simulations on Body-Unfitted Cartesian Grids

I. S. Menshov[a]* and P. V. Pavlukhin[b]**

[a] *Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, Miusskaya pl. 4, Moscow, 125047 Russia*

[b] *Research Institute Kvant, Chetvertyi Likhachevskii per. 15, Moscow, 125438 Russia*

*e-mail: *menshov@kiam.ru, **pavelpavlukhin@kiam.ru*

Received June 26, 2015; in final form, February 5, 2016

Abstract—For problems with complex geometry, a numerical method is proposed for solving the three-dimensional nonstationary Euler equations on Cartesian grids with the use of hybrid computing systems. The baseline numerical scheme, a method for implementing internal boundary conditions on body-unfitted grids, and an iterative matrix-free LU-SGS method for solving the discretized equations are described. An efficient software implementation of the numerical algorithm on a multiprocessor hybrid CPU/GPU computing system is considered. Results of test computations are presented.

Keywords: computational gas dynamics, Cartesian grids, free boundary method, parallel computations, hybrid computing systems.

DOI: 10.1134/S096554251609013X

INTRODUCTION

Gasdynamic simulation on modern computing systems with new architectures encounters a number of difficulties. One is associated with the discretization of the computational domain. In the case of a domain of complex geometry, unstructured meshes are used in most cases. Accordingly, mesh generation requires considerable computational resources and quite often involves hand operations for mesh correction, which lead to high time costs. Unstructured meshes lead to irregular memory access. As result, the efficiency of software implementations of methods on such meshes is limited not by the number of arithmetic operations executed per unit time (compute-bound), but rather by the memory bandwidth (memory-bound). To a higher degree, this is critical for massively parallel processors, since their performance depends primarily on the order of memory access. Therefore, it is preferable to use structured meshes, which are characterized by a regular pattern of memory access. However, in some cases, the construction of such meshes fitted to the boundary of the computational domain may be hard or impossible to implement.

Another difficulty is associated with the interdependent development of numerical methods and processor architectures. Earlier, low-resolution grids were used for solving problems on systems with limited computational resources. To obtain more accurate solutions on them, high-order numerical methods with a complex algorithmic structure had to be constructed. On the other hand, processor cores also became progressively “heavier”: out-of-order execution of commands, cache data storage, prediction of branching, and vector instructions allowed one to efficiently implement complicated methods. However, the scalability of computing systems on such heavy cores was bounded, and their further complication led to lower efficiency. This motivated the development of systems on new massively parallel architectures with a large number of simple cores. It is at this point where a problem arose: the numerical methods having been developed over decades turned out to be unfit for implementation on new computers, since high efficiency on them is achieved not due to efficient execution of each of a small (about 10) number of “heavy” streams, but rather due to the simultaneous processing of a much greater (about 1000) number of “light” streams. In other words, the simple structure of the core of a massively parallel computer requires computational primitivism for the numerical methods applied.

Explicit methods are well suited for implementation on new architectures, but their application is strongly limited by the stability condition. In problems with complex geometry, where meshes inevitably contain cells of different scales, the global time step is determined by the size of the smallest cell, which leads to unreasonably high growth of computational complexity. Implicit methods do not have this limitation, but they are much more complicated in terms of the above requirements, especially in parallelization. Thus, choosing and creating methods and corresponding efficient parallel algorithms for solving a broad class of problems in gas dynamics is an important task.

The implementation of implicit schemes on graphics processing units (GPUs) is a rather difficult problem, since the development of an efficient solver is considerably hampered by data dependency, especially in the absence of tools for global synchronization on GPUs. Specifically, the LU-SGS method [2] designed for solving systems of linear algebraic equations (SLAEs) generated by implicit schemes was considered in [1] as a good method for computations on GPUs. However, because of data dependency, which strongly complicates parallelization, another method was chosen, namely, DP-LUR, which is free of data dependency, but has a higher computational complexity.

In this paper, we propose an original parallel LU-SGS algorithm that performs exactly as its sequential prototype and is scalable to several hundred GPUs. In addition to the numerical method itself, an important role, as was mentioned above, is played by the mesh type. Specifically, GPUs perform much better with regular data structures typical of structured meshes than with irregular ones characteristic of unstructured meshes. Structured meshes adapted to the geometry of the computational domain impose constraints on the complexity of its geometry. The method proposed in this paper applies to domains with fairly complex geometry and performs on simple body-unfitted Cartesian grids, which makes it a suitable candidate for implementation on GPUs. The method is reduced to an alternative formulation in which internal boundary conditions are modeled using a compensating flux, which is a special correction term added to the right-hand side of the governing system of equations. As a result, shock-capturing computations can be performed over all cells of the computational domain in a unified manner. In other words, the method is algorithmically homogeneous, which is an important characteristic for a massively parallel architecture.

1. BASELINE NUMERICAL METHOD

To solve the compressible Euler equations numerically, they are written in Cartesian coordinates in conservative form:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}_k}{\partial x_k} = 0, \tag{1}$$

where summation is implied over the coordinate direction index k,

$\mathbf{q} = (\rho, \rho u_m, \rho E)^{\mathrm{T}}$ is the vector of conservative variables,

$\mathbf{f}_k = (\rho u_k, \rho u_k u_m + \delta_{km} p, \rho u_k H)^{\mathrm{T}}$ are the flux vectors,

$m = 1,2,3$; δ_{km} is the Kronecker delta; ρ, u_k, p are the density, velocity components, and pressure, respectively; and $E = e + 0.5 u_k u_k$ and $H = e + p/\rho$ are the specific total energy and enthalpy. System (1) is closed by an equation of state, which relates the thermodynamic parameters of the medium. In this study, we used the ideal gas equation of state $p = (\gamma - 1)\rho e$, where γ is the ratio of specific heats. In the computations presented below, γ was used for the case of air, i.e., $\gamma = 1.4$.

In the computational domain, we introduced a Cartesian grid along the coordinate directions with steps h_k, which can generally be variable. Applying the finite volume method to Eqs. (1), we obtain the system of semidiscrete equations

$$\frac{d\mathbf{q}_i}{dt} = -\left(\frac{\Delta \mathbf{F}_k}{h_k}\right)_i, \tag{2}$$

where i is a generalized cell index, which depends on the coordinate direction index k and takes the value of a cell index in this direction.

The flux difference on the right-hand side of (2) is

$$\Delta \mathbf{F}_k = \mathbf{F}_{k,i+1/2} - \mathbf{F}_{k,i-1/2}, \tag{3}$$

where $\mathbf{F}_{k,i+1/2}$ is the numerical flux approximating the differential flux $\mathbf{f}_k$ on the face between cells i and $i+1$. This numerical flux is assumed to be a function of two vector arguments,

$$\mathbf{F}_{k,i+1/2} = \mathbf{F}_k(\mathbf{z}_i^+, \mathbf{z}_{i+1}^-), \tag{4}$$

satisfying the approximation condition $\mathbf{F}_k(\mathbf{z},\mathbf{z}) = \mathbf{f}_k(\mathbf{z})$. Here, $\mathbf{z}$ denotes the vector of primitive variables: $\mathbf{z} = (\rho, u_k, p)$.

The plus and minus superscripts on the right-hand sides of Eqs. (3) indicate that the corresponding values are taken at the center of face $\underline{i} + 1/2$. The choice of these values determines the accuracy of the scheme. For example, with the simple choice

$$\mathbf{z}_i^+ = \mathbf{z}_i, \quad \mathbf{z}_{i+1}^- = \mathbf{z}_{i+1}, \tag{5}$$

we obtain a first-order accurate scheme in space. To increase the order of accuracy of the scheme, a higher order of accurate subgrid-scale reconstruction of the parameters has to be applied [3]. In this paper, we use a piecewise linear subgrid-scale extension that generalizes the monotone upstream-centered scheme for conservation laws (MUSCL) [4] to nonuniform grids:

$$\mathbf{z}^\pm = \mathbf{z} \pm 0.5\delta^\pm\left[(1-k^\pm)\Delta^\mp + (1+k^\pm)\Delta^\pm\right] \tag{6}$$

with finite differences $\Delta^+ = \mathbf{z}_{i+1} - \mathbf{z}_i$ and $\Delta^- = \mathbf{z}_i - \mathbf{z}_{i-1}$. In this equation, the cell index i is omitted for notational brevity, $\delta^+ = h_i/(h_i + h_{i+1})$ and $\delta^- = h_i/(h_i + h_{i-1})$ are grid nonuniformity parameters, and $k^\pm = k(\delta^\pm)$ is a function determining the order of accuracy of the interpolation scheme. For $k(\delta) = -1$, we have the standard second-order accurate MUSCL scheme [5], $k(\delta) = 1$ gives an unstable second-order central-difference scheme, $k(\delta) = 0$ corresponds to Fromm's scheme [6], and $k(\delta) = (12\delta^2 - 1)/(12\delta)$ leads to a third-order accurate MUSCL scheme in space. Below, we use the last version of the scheme.

Schemes of second and higher order of accuracy in space are nonmonotone and give rise to spurious oscillations in numerical solutions near strong discontinuity surfaces [7]. To suppress these oscillations, one uses limiters of derivatives [3, 5], which modify the differences $\Delta^\pm$ so that interpolation (6) does not lead to local extrema.

The most popular are the minmod limiter (see [3, 4])

$$\begin{gathered}\Delta^+ = \operatorname{minmod}(\Delta^+, \varphi\Delta^-), \quad \Delta^- = \operatorname{minmod}(\Delta^-, \varphi\Delta^+),\\ \varphi = \frac{3-k}{1-k}, \quad \operatorname{minmod}(x,y) = \begin{cases}0 & \text{if} \quad xy < 0,\\ \operatorname{sign}(x)\min(|x|,|y|) & \text{if} \quad xy \ge 0,\end{cases}\end{gathered} \tag{7}$$

and the van Albada limiter (see [8])

$$\begin{gathered}\mathbf{z}^\pm = \mathbf{z} \pm 0.5s\delta^\pm\left[(1-sk^\pm)\Delta^\mp + (1+sk^\pm)\Delta^\pm\right],\\ s = \max\left(0, \frac{2\Delta^+\Delta^-}{\Delta^+\Delta^+ + \Delta^-\Delta^- + \varepsilon}\right),\end{gathered} \tag{8}$$

where ε is a small number used to avoid division by zero ($\varepsilon \sim 10^{-12}$ for double precision arithmetic operations and $\varepsilon \sim 10^{-6}$ for single precision arithmetic operations). The former limiter is not a continuously differentiable function and may lead to an infinite loop of the residual in the course of the convergence of the solution. We use the latter limiter (a smooth function), which avoids an infinite loop of the residual in many problems.

Godunov's method [9] is used to approximate the numerical flux function. In this approach, the flux is computed at the solution of the following self-similar Riemann problem for a system of locally one-dimensional equations:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}_k}{\partial x} = 0 \tag{9}$$

with initial data $\mathbf{z}(0,x) = \begin{cases}\mathbf{z}_i^+, & x < 0,\\ \mathbf{z}_{i+1}^-, & x \ge 0.\end{cases}$

The solution of this problem was described in detail in [10]. In fact, the problem is reduced to a single linear equation for the contact zone pressure, which can be solved by applying Newton's iterations. The other parameters of the solution have exact analytical expressions. Let this solution be denoted as

$\mathbf{z}(t,x) = \mathbf{Z}^{R,k}\left(\lambda, \mathbf{z}_i^+, \mathbf{z}_{i+1}^-\right)$, where $\lambda = x/t$ is a self-similar variable. Then the standard Godunov numerical flux is given by

$$\mathbf{F}_k\left(\mathbf{z}_i^+, \mathbf{z}_{i+1}^-\right) = \mathbf{f}_k\left[\mathbf{Z}^{R,k}\left(0, \mathbf{z}_i^+, \mathbf{z}_{i+1}^-\right)\right]. \tag{10}$$

The numerical flux has to be computed on each cell edge and, in fact, is the basic operation of a numerical cycle (in terms of time costs). For this reason, various noniterative approximate Riemann solvers were developed [11] to enhance the efficiency of Godunov's scheme. One of them leads to the numerical flux proposed in [12]. Below, we use it to linearize the numerical flux. It has the form

$$\mathbf{F}_k\left(\mathbf{z}_i^+, \mathbf{z}_{i+1}^-\right) = \frac{1}{2}\left\{\mathbf{f}_k\left(\mathbf{z}_i^+\right) + \mathbf{f}_k\left(\mathbf{z}_{i+1}^-\right) - (|u_k| + c)_{i+1/2}\left[\mathbf{q}\left(\mathbf{z}_{i+1}^-\right) - \mathbf{q}\left(\mathbf{z}_i^+\right)\right]\right\}, \tag{11}$$

where the index $i + 1/2$ means averaging; for example, $(*)_{i+1/2} = 0.5\left[(*)_i^+ + (*)_{i+1}^-\right]$.

The system of semidiscrete equations (2)–(4) with a numerical flux of form (10) was integrated with respect to time by applying an explicit–implicit scheme [13], which ensures the minimum involvement of the dissipative implicit component and guarantees the maximum norm diminishing (MND) property in the case of linear equations. With a suitable choice of the time step, this scheme automatically becomes the baseline explicit second-order accurate scheme in time and space, for which the MND property holds.

To construct a hybrid explicit–implicit scheme, a baseline explicit scheme is chosen to be a two-step predictor–corrector one, although other variants are also possible. In this scheme, at the first step, the predictor values at the time half-step are computed using the explicit Euler scheme

$$\tilde{\mathbf{q}}_i = \mathbf{q}_i^n - \frac{\Delta t}{2}\left(\frac{\Delta\mathbf{F}_k(\mathbf{z}^n)}{h_k}\right)_i \tag{12}$$

with the flux $\mathbf{F}_{k,i+1/2}$ determined by the cell-face interpolated values

$$\Delta\mathbf{F}_k(\mathbf{z}^n) = \mathbf{f}_k(\mathbf{z}_i^+) - \mathbf{f}_k(\mathbf{z}_i^-). \tag{13}$$

Here, the superscript n denotes the discrete time level. The solution at the new time level $n + 1$ is obtained by applying an explicit second-order accurate scheme at the step Δt with the fluxes computed from the predictor values:

$$\mathbf{q}_i^{n+1} = \mathbf{q}_i^n - \Delta t\left(\frac{\Delta\mathbf{F}_k(\tilde{\mathbf{z}})}{h_k}\right)_i; \qquad \mathbf{F}_{k,i+1/2} = \mathbf{F}_k\left(\tilde{\mathbf{z}}_i^+, \tilde{\mathbf{z}}_{i+1}^-\right). \tag{14}$$

The above-described predictor–corrector scheme guarantees the MND property (in the linear case) under the Courant–Friedrichs–Lewy stability condition imposed on the integration time step:

$$\Delta t \le \lambda_i(\mathbf{z}^n) \quad \text{for all} \quad i, \tag{15}$$

where the function on the right-hand side is determined by the local flow velocity and the speed of sound:

$$\lambda_i(\mathbf{z}^n) = K_s\left(\frac{|u_k| + C}{h_k}\right)^{-1}, \tag{16}$$

here, K_s is a safety factor ($0 < K_s \le 1$).

The baseline explicit scheme is written in operator form:

$$\mathbf{q}_i^{n+1} = \mathbf{q}_i^n + \Delta t L_2\left(\Delta t, \mathbf{q}^n\right), \tag{17}$$

where $L_2(\cdot)$ denotes the discrete two-step transition operator in (12)–(14). For each mesh cell, we introduce an intermediate-level parameter, i.e., a scalar parameter ω_i, $0 \le \omega_i \le 1$. The solution vector at the intermediate level is defined as $\mathbf{q}^\omega = \omega\mathbf{q}^n + (1 - \omega)\mathbf{q}^{n+1}$. Then the hybrid explicit–implicit scheme is written as

$$\mathbf{q}_i^{n+1} = \mathbf{q}_i^n + \Delta t L_2\left(\omega_i\Delta t, \mathbf{q}^\omega\right). \tag{18}$$

Obviously, with the choice of $\omega_i = 1$ in all mesh cells, the hybrid scheme becomes its explicit counterpart. In the other limiting case $\omega_i = 0$, it becomes a fully implicit Euler scheme with a second-order accurate approximation in space.

The implicit component of the hybrid scheme introduces excessive dissipation, which stabilizes the numerical solution, but degrades its accuracy. Therefore, an optimal solution in the design of a hybrid scheme is to choose the largest values of ω_i in mesh cells, which ensures the maximum participation of the explicit component, provided that the MND property holds.

It turns out that such a choice can be made. Note that the hybrid scheme can be rewritten in an equivalent form as

$$\mathbf{q}_i^{n+1} = \mathbf{q}_i^{\omega} + \omega_i \Delta t L_2\left(\omega_i \Delta t, \mathbf{q}^{\omega}\right). \tag{19}$$

It is easy to see that Eq. (19) is an explicit two-step scheme for the transition from the variable intermediate level $t^{\omega} = \omega t^n + (1-\omega)t^{n+1}$ to the upper layer t^{n+1}. Therefore, the MND property

$$\left\|\mathbf{q}^{n+1}\right\|_{\infty} \le \left\|\mathbf{q}^{\omega}\right\|_{\infty} \tag{20}$$

holds if

$$\omega_i \Delta t \le \lambda_i(\mathbf{z}^{\omega}). \tag{21}$$

A consequence of (20) is the inequality $\left\|\mathbf{q}^{n+1}\right\|_{\infty} \le \left\|\mathbf{q}^{n}\right\|_{\infty}$ (see [13]). Therefore, inequality (21) is also a necessary condition for the MND property to hold for the numerical solutions of the hybrid scheme.

Inequality (21) is satisfied if we set

$$\omega_i = \min\left[1, \frac{\lambda_i(\mathbf{z}^{\omega})}{\Delta t}\right] \tag{22}$$

or, in view of Eq. (16),

$$\omega_i = \min\left[1, \frac{K_s}{\Delta t}\left(\frac{\left|u_k^{\omega}\right| + C^{\omega}}{h_k}\right)_i^{-1}\right]. \tag{23}$$

It can be seen that the intermediate-level parameter in every mesh cell depends on the solution on the upper time level. Thus, the hybrid scheme is, in fact, implicit and requires the solution of a nonlinear system of equations at every time step. Note also that, with a suitable choice of the time step Δt, the number of equations to be solved can be reduced by making the hybrid scheme in some of the mesh cells purely explicit. In more detail solution of a system of nonlinear equations is discussed below.

2. INTERNAL BOUNDARY CONDITIONS

To implement internal boundary conditions on a Cartesian grid that is not fitted to the geometry of the solution domain, we use the free boundary method [14, 15]. Let us briefly describe the basic principles of this method.

Let Ω be the domain occupied by a solid and $\Gamma = \partial\Omega$ be its boundary. Then the Euler equations (1) describe the gas flow in the domain $\Re^3 \backslash \Omega$, and the boundary condition for these equations is the relation

$$u_k n_k = 0, \quad \mathbf{x} \in \Gamma, \tag{24}$$

where $\mathbf{n} = (n_k)$ is the outward unit normal vector to Ω on the surface Γ.

The idea of the free boundary method is to replace the boundary value problem for the homogeneous Euler equations in a bounded domain by an inhomogeneous system in the entire space. The original system of equations is modified by adding a vector $\mathbf{F}_w$ to the right-hand side (in what follows, this vector is called the *compensating flux*):

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}_k}{\partial x_k} = -\mathbf{F}_w. \tag{25}$$

The modification of Eqs. (25) resembles, in structure, penalty functions used in the immersed boundary method [16]. In fact, these are different approaches. Penalty functions are volumetric and do not take into account the subgrid-scale geometric structure. On the contrary, compensating fluxes are determined on the surface specifying the object's geometry and are computed, as will be shown below, with allowance for the geometric characteristics of the object at the subgrid-scale level.

The system of modified equations (25) is solved in the entire space $\Re^3$, and the compensating flux $\mathbf{F}_w$ is chosen so that the restriction of the solution of Eq. (25) in $\Re^3$ to $\Re^3\backslash\Omega$ coincides with the solution of the original boundary value problem. An expression for $\mathbf{F}_w$ that ensures the fulfillment of this condition was proposed in [14], namely,

$$\mathbf{F}_w = \begin{pmatrix} \rho u_k n_k \\ \rho u_k u_m n_k + (p - p_w) n_m \\ \rho u_k n_k H \end{pmatrix} \delta(\mathbf{x}, \Gamma), \tag{26}$$

where $\delta(\mathbf{x}, \Gamma)$ denotes a generalized Dirac function of Γ defined by the relation

$$\int_V \delta(\mathbf{x}, S)\varphi(\mathbf{x}) dV = \int_{V \cap \Gamma} \varphi(\mathbf{x}) dS \quad \forall \mathrm{V} \in \Re^3. \tag{27}$$

The quantity p_w in Eqs. (26) is the instantaneous reaction of the solid wall to the influence exerted by the fluid flow. For an ideal gas with the ratio of specific heats γ, this reaction is defined depending on the sign of the normal relative velocity as follows.

If $u_k n_k < 0$, then the wall pressure p_w corresponds to the pressure behind the front of a developing shock wave and is given by the formula (see [17])

$$p_w = p\left[1 + \frac{\gamma(\gamma+1)}{4}\mathrm{M}^2 + \sqrt{\gamma^2 M^2 + \frac{\gamma^2(\gamma+1)^2}{16}\mathrm{M}^4}\right]. \tag{28}$$

In the case $u_k n_k < 0$, a rarefaction wave is formed and the wall pressure has the form

$$p_w = p\left[1 - \frac{\gamma - 1}{2}\mathrm{M}\right]^{\frac{2\gamma}{\gamma-1}}. \tag{29}$$

Here, M denotes the relative number of the Mach: $\mathrm{M} = u_k n_k / a$, where a is the speed of sound.

Note that, mathematically, the compensating flux can be chosen not necessarily in the form of (27)–(29). Possibly, there are other right-hand sides for which boundary condition (24) is also satisfied. Our choice is based on natural physical substantiation. The flux $\mathbf{F}_w$ is simulated in two components. The first balances the mass, momentum, and energy flowing past the surface Γ, while the second component determines the reaction of the wall and the corresponding work.

In the numerical implementation of the free boundary method, the surface Γ specifying the geometry of the body is assumed to be representable as a set of discrete elements: $\Gamma = \bigcup_j \Gamma_j$. Usually, these are plane polyhedra or triangular spatial elements. The cells of the Cartesian grid are divided by the body boundary into three types: liquid (lying entirely outside Γ), solid (lying entirely inside Γ), and mixed. The mixed cells are determined by the computed vector and by the subgrid-scale structure of the body geometry. In this paper, we use a linear extension, which is defined by the volume fraction occupied by the liquid in an mixed cell, ω_f, and by the outward normal vector (directed toward the liquid) $\mathbf{n}_f$, $|\mathbf{n}_f| = s_f$, where s_f is the area of the plane element approximating the intersection of the mesh cell with Γ.

These parameters are computed when the intersection of the mesh cell with the surface elements Γ_j is analyzed. The normal vector is computed by averaging the unit normals of the elements with weighting coefficients equal to the area of the intersection: $\mathbf{n}_f = \mathbf{n}_j s_j$, where $s_j = \mathrm{area}(\Gamma_j \cap C)$ and C denotes the grid cell domain. The volume fraction ω_f in an mixed cell can be approximately computed in terms of the mean distances from the liquid vertices of the cell, d^+, and from the solid vertices, d^-, to the plan normal to $\mathbf{n}_f$ passing through the reference point $\mathbf{x}_0$:

$$\omega_f = d^+/(d^+ + d^-). \tag{30}$$

The reference point is determined by the centers of the surface elements $\mathbf{x}_j$:

$$\mathbf{x}^j = \frac{|\mathbf{n}^{j-1}|\mathbf{x}^{j-1} + \mathbf{x}_j s_j}{|\mathbf{n}^{j-1}| + s_j}, \quad \mathbf{n}^j = \sum_{k=1}^{j} \mathbf{n}_k s_k, \quad \mathbf{x}_0 = \mathbf{x}^M, \tag{31}$$

where M is the number of elements on Γ.

System (25) is discretized in two stages with the use of splitting with respect to physical processes. At the first stage, the homogeneous system is integrated over a set of liquid and mixed cells according to the numerical scheme described in the preceding section:

$$\begin{gathered}\mathbf{q}_i^* = \mathbf{q}_i^n - \Delta t\left(\frac{\Delta \mathbf{F}_k(\tilde{\mathbf{z}})}{h_k}\right)_i; \\ \mathbf{F}_{k,i+1/2} = \mathbf{F}_k\left(\tilde{\mathbf{z}}_i^+, \tilde{\mathbf{z}}_{i+1}^-\right).\end{gathered} \tag{32}$$

Thus, the solution $\mathbf{q}_i^*$ at this stage describes the variations in the gasdynamic parameters over a time step according to the Euler model without allowance for the influence exerted by the wall on the flow parameters.

At the second stage, the resulting solution is corrected using the compensating flux on the right-hand side. The corresponding discrete equations are derived from the physical considerations mentioned above.

Using the solution $\mathbf{q}_i^*$ as initial data, we integrate the homogeneous Euler equations over the liquid part of a mesh cell:

$$\omega_f V_i \frac{d\mathbf{q}_i}{dt} = -\sum_{\sigma \in f} \mathbf{F}_\sigma s_\sigma + \mathbf{F}_p s_f, \tag{33}$$

where $\mathbf{F}_p = (0, p_w n_m, 0)^{\mathrm{T}}$ is the numerical flux on the wall, the sum on the right-hand side is taken over cell faces that are completely or partially in the liquid, and s_σ is the area of the liquid part of a face. Since only the influence of the wall is taken into account at this stage, the face fluxes $\mathbf{F}_\sigma$ are computed using the cell values $\mathbf{q}_i^*$.

Since the flow is conservative, Eq. (33) can be rewritten as

$$\omega_f V_i \frac{d\mathbf{q}_i}{dt} = -\mathbf{F}_c s_f + \mathbf{F}_p s_f \tag{34}$$

with the flux $\mathbf{F}_c = (\rho u_k n_k, \rho u_k u_m n_k + p n_m, \rho u_k n_k H)^{\mathrm{T}}$ and $V_i = h_1 h_2 h_3$. The sum of the fluxes on the right-hand side gives the compensating flux introduced above: $\mathbf{F}_w = \mathbf{F}_c - \mathbf{F}_p$.

Equation (34) is integrated with respect to time by applying an implicit scheme to avoid severe restrictions on the time step caused by the volume fraction ω_f, which can generally be arbitrarily small in some mixed cells. This leads to the system of discrete equations

$$\mathbf{q}_i^{n+1} = \mathbf{q}_i^* - \frac{\Delta t s_f}{\omega_f V_i} \mathbf{F}_w\left(\mathbf{q}_i^{n+1}\right). \tag{35}$$

Combining (32) and (35) yields the numerical scheme

$$\begin{gathered}\mathbf{q}_i^{n+1} = \mathbf{q}_i^n - \Delta t\left(\frac{\Delta \mathbf{F}_k(\tilde{\mathbf{z}})}{h_k}\right)_i - \frac{\Delta t s_f}{\omega_f V_i} \mathbf{F}_w\left(\mathbf{q}_i^{n+1}\right), \\ \mathbf{F}_{k,i+1/2} = \mathbf{F}_k\left(\tilde{\mathbf{z}}_i^+, \tilde{\mathbf{z}}_{i+1}^-\right),\end{gathered} \tag{36}$$

which is absolutely stable when the intermediate-level parameter is computed by formula (23) and is third-order accurate in space and second-order accurate in time.

The nonlinear system (36) is solved using the matrix-free approximate factorization method LU-SGS (see [2, 18]). This method is economic and is reduced to forward and reverse cycles over mesh cells that, in fact, implement an explicit scheme. A detailed description of the solution method is given below.

3. SOLUTION OF THE SYSTEM OF DISCRETE EQUATIONS

Equations (36) are solved using Newton's iterative method. It leads to a linear system for the iterative variations $\delta^s(\cdot) = (\cdot)^{n+1,s+1} - (\cdot)^{n+1,s}$, where s is the iterative index:

$$\left(I + \frac{\Delta ts_f}{\omega_f V_i} A_w^s\right)\delta^s\mathbf{q}_i = -\Delta^s\mathbf{q}_i - \Delta t\left(\frac{\Delta^s\mathbf{F}_k}{h_k}\right)_i - \frac{\Delta ts_f}{\omega_f V_i}\mathbf{F}_w\left(\mathbf{q}_i^{n+1,s}\right) - \Delta t\frac{\delta^s\mathbf{F}_{k,i+1/2} - \delta^s\mathbf{F}_{k,i-1/2}}{h_k}. \tag{37}$$

Here, $\Delta^s = (\cdot)^{n+1,s} - (\cdot)^n$ is the iterative increment over a time step and $A_w = \partial\mathbf{F}_w/\partial\mathbf{q}$ is the Jacobian of the compensating flux with respect to the vector of conservative variables.

The numerical flux $\delta^s\mathbf{F}_{k,i+1/2}$ is linearized approximately with the use of the following simplifying assumptions. First, although ω depends on the iterative values, it is assumed to be fixed in the linearization. Second, the subgrid-scale interpolations of the computed vector are not taken into account and the numerical flux is assumed to depend on the parameter values in the mesh cell, as in the standard first-order accurate scheme. Third, the numerical flux is not precisely Godunov's, but rather its approximation in Rusanov's form [12], which is given by (11). Under these assumptions, the linearization of the numerical flux yields the simple expression

$$\delta^s\mathbf{F}_{k,i+1/2} = (1-\omega_i)\frac{A_{k,i}^{\omega,s} + (|u_k| + c)_{i+1/2}^{\omega,s}}{2}\delta^s\mathbf{q}_i + (1-\omega_{i+1})\frac{A_{k,i+1}^{\omega,s} - (|u_k| + c)_{i+1/2}^{\omega,s}}{2}\delta^s\mathbf{q}_{i+1}, \tag{38}$$

where $A_{k,i}^{\omega,s} = A_k\left(\mathbf{z}_i^{\omega,s}\right) = \partial\mathbf{f}_k/\partial\mathbf{q}$ is the Jacobian of the flux in the kth direction with respect to the vector of conservative variables and the superscript ω indicates that the parameter value in the mesh cell is taken from the intermediate level.

In view of Eq. (38), system (37) is reduced to a linear system for the iterative residuals $\delta^s\mathbf{q}$:

$$\begin{aligned} D_i^s\delta^s\mathbf{q}_i = \mathbf{R}_i^s &- \frac{\Delta t(1-\omega_{i+1})}{h_k}\left[\frac{A_{k,i+1}^{\omega,s} - (|u_k| + c)_{i+1/2}^{\omega,s}}{2}\right]\delta^s\mathbf{q}_{i+1} \\ &+ \frac{\Delta t(1-\omega_{i-1})}{h_k}\left[\frac{A_{k,i-1}^{\omega,s} + (|u_k| + c)_{i-1/2}^{\omega,s}}{2}\right]\delta^s\mathbf{q}_{i-1}, \end{aligned} \tag{39}$$

where

$$\begin{aligned} D_i^s &= I + \frac{\Delta ts_f}{\omega_f V_i}A_w^s + \frac{\Delta t(1-\omega_i)}{2h_k}\left[(|u_k| + c)_{i-1/2}^{\omega,s} + (|u_k| + c)_{i+1/2}^{\omega,s}\right], \\ \mathbf{R}_i^s &= -\Delta^s\mathbf{q}_i - \Delta t\left(\frac{\Delta^s\mathbf{F}_k}{h_k}\right)_i - \frac{\Delta ts_f}{\omega_f V_i}\mathbf{F}_w\left(\mathbf{q}_i^{n+1,s}\right). \end{aligned} \tag{40}$$

For the subsequent consideration, it is convenient to make inverse linearization on the right-hand side of Eq. (39) by passing from the product of the Jacobian and the iterative increment of the conservative vector to the increment of the corresponding flux vector. Let the spectral radius of the Jacobian be denoted by $r_k = |u_k| + c$. Then the resulting system of equations can be brought to the form

$$D_i^s\delta^s\mathbf{q}_i = \mathbf{R}_i^s - \frac{\Delta t(1-\omega_{i+1})}{2h_k}(\delta^s\mathbf{f}_{k,i+1}^{\omega} - r_{k,i+1/2}^{\omega,s}\delta^s\mathbf{q}_{i+1}) + \frac{\Delta t(1-\omega_{i-1})}{2h_k}(\delta^s\mathbf{f}_{k,i-1}^{\omega} - r_{k,i-1/2}^{\omega,s}\delta^s\mathbf{q}_{i-1}). \tag{41}$$

This system is solved using the matrix-free iterative LU-SGS method [18]. The matrix of the system is seven-diagonal. In a split operator form, it can be written as

$$D\delta\mathbf{q} + L(\delta\mathbf{q}) + U(\delta\mathbf{q}) = \mathbf{R} \tag{42}$$

with the upper triangular matrix operator

$$L(\delta\mathbf{q}) = \frac{\Delta t(1-\omega_{i+1})}{2h_k}(\delta^s\mathbf{f}^{\omega}_{k,i+1} - r^{\omega,s}_{k,i+1/2}\delta^s\mathbf{q}_{i+1})$$

and the lower triangular matrix operator

$$U(\delta\mathbf{q}) = -\frac{\Delta t(1-\omega_{i-1})}{2h_k}(\delta^s\mathbf{f}^{\omega}_{k,i-1} - r^{\omega,s}_{k,i-1/2}\delta^s\mathbf{q}_{i-1}).$$

After rewriting system (42) as

$$(D+L)D^{-1}(D+U)\delta\mathbf{q} = \mathbf{R} + LD^{-1}U, \tag{43}$$

the solution is then found using its approximate factorization, which consists in dropping the last term on the right-hand side of Eq. (43). The resulting simplified system is

$$(D+L)D^{-1}(D+U)\delta\mathbf{q} = \mathbf{R}, \tag{44}$$

which is efficiently solved in two numerical cycles over cells in forward and reverse directions (over the cell index), respectively:

$$\begin{aligned}\delta\mathbf{q}^* &= D^{-1}[\mathbf{R} - L(\delta\mathbf{q}^*)],\\ \delta\mathbf{q}^s &= \delta\mathbf{q}^* - D^{-1}U(\delta\mathbf{q}^s).\end{aligned} \tag{45}$$

Note that the terms dropped in the approximate factorization are on the order of $O(\Delta t^2)$. Therefore, the error introduced by the factorization is small when a small time step is used to solve the problem.

4. IMPLEMENTATION ON MULTI-GPU

After introducing the compensating flux, the structure of SLAE (37) obtained by linearizing the discrete system of equations does not change, since this flux represents a local addition. Relying on this property, we can construct a parallel LU-SGS algorithm in the same manner as in the case of body-fitted grids. The solution of the system with a factorized matrix can be represented as the traverse of all mesh cells in forward and reverse directions. The data dependence arising is local and is determined only by the order of traversing all geometrically neighboring cells with respect to each of them. In other words, corresponding computations are executed depending on whether a neighboring cell is “before” or “after” the current cell in the traverse. The order of traverse can be chosen arbitrarily, not only based on the geometric neighborhood of cells. This circumstance is used in the design of a parallel algorithm.

Computations in cells can be performed simultaneously only if they are not geometric neighbors. In this case, the choosing of cell traverse is reduced to the problem of coloring a graph so that two neighboring vertices are always of different color. In the case of structured Cartesian grids, which are used in the free boundary method, it is sufficient to use two colors. As a result, a checkered pattern of cell traverse is generated, i.e., the computations are performed first over black cells and then over white cells. The subset of each color can be processed concurrently, which yields a parallel algorithm for a single GPU: only white cells are computed first, followed by computing only black cells. However, in the case of several GPUs, this algorithm leads to inevitable idle time during the exchange of neighboring cells between the GPUs. Eventually, this restricts the scalability of the algorithm, since data exchange between GPUs is performed, in fact, in three stages: from GPU memory to CPU memory, then network transfer to another node, and copying from CPU memory at this node to GPU memory.

To overcome this limitation, the order of traversing cells is changed. The computational domain is partitioned into blocks with a roughly identical number of cells with a topology similar to the mesh partition, i.e., adjacent rectangles arranged in lines. The blocks are also partitioned in a checkered pattern, and internal and boundary parts is identified in each of them. The global traverse is as follows: all internal parts of the black blocks are traversed first; then, in each white block, the first half of the internal cells and then the boundary part are traversed; next, the boundary parts are traversed in all black blocks; and, finally, the remaining halves of the internal cells are traversed in white blocks (Fig. 1). In this scheme, the internal parts of blocks can be computed concurrently with exchange of boundary cells between neighboring blocks on different GPUs. Each part of the blocks is computed according to the above checkered coloring of the cells. More details concerning this algorithm can be found in [19].

The software implementation was performed on CUDA C with Stream API and nonblocking MPI calls, so that the computations were executed in parallel with multiphase data transfer between the GPUs.

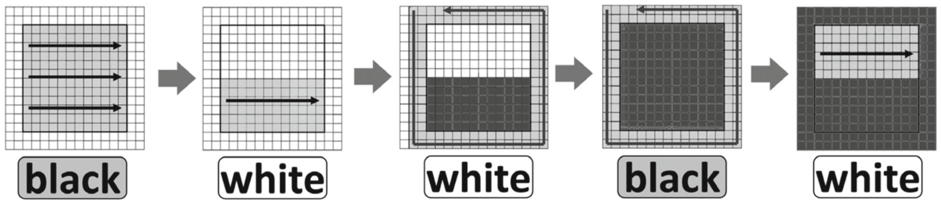

Fig. 1. Schematic of cell traverse in the parallel LU-SGS algorithm.

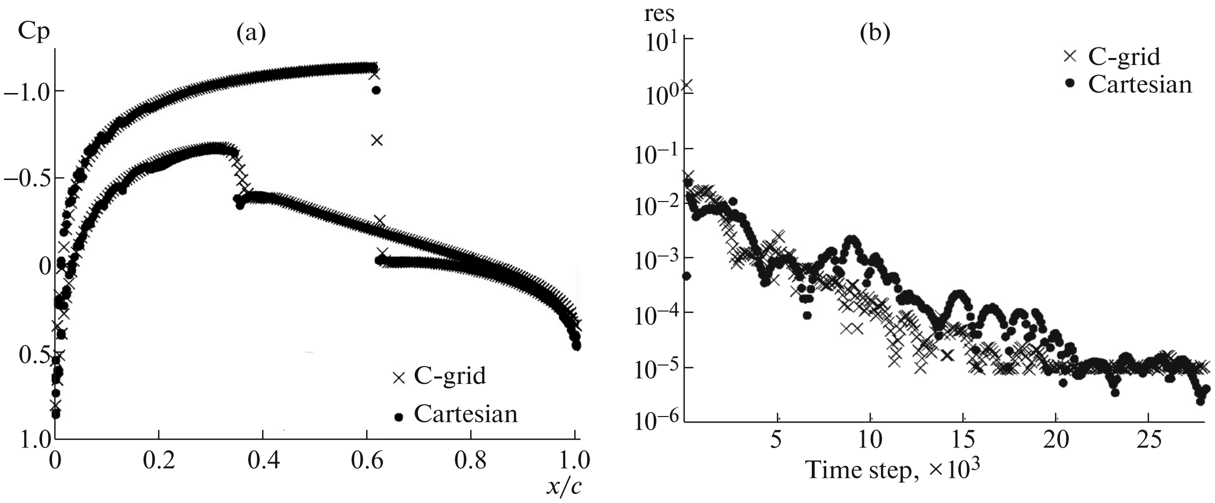

Fig. 2. (a) Cp distribution and (b) convergence rate on the body-fitted C grid and body-unfitted Cartesian grid for NACA0012, M = 0.8, $\alpha = 1.25°$, and a Courant number of 10.

All computational operations were executed on the GPUs, while the CPU were used only for internode exchange.

5. NUMERICAL RESULTS

The method was tested in detail as applied to the flow past the NACA0012 airfoil. The angle of attack was specified as $\alpha = 1.25°$, and the free-stream Mach number was M = 0.8. Meshes of two types were used: a body-fitted structured C-grid with 400 cells along the airfoil boundary and with a general resolution of 1000×150 cells and a body-unfitted Cartesian grid with a resolution of 200×24 inside a rectangle circumscribed about the airfoil and with a general resolution of 650×324.

The computations on these grids were performed using the hybrid explicit–implicit second-order accurate scheme with the Courant number C = 10. Figure 2 shows the distribution of the pressure coefficient Cp and the convergence rates to the steady-state solution. It can be seen the results obtained on different grids, including the convergence rate, agree well. A characteristic feature of the flow is a small local minimum in the pressure distribution arising behind the shock wave on the upstream side of the airfoil. Evidently, this feature is better reproduced on the Cartesian than body-fitted grid. In fact, the local minimum is absent on the latter grid and appears as the grid resolution is increased. This finding is associated with the orthogonality property, which is inherent in the Cartesian grid and ensures higher accuracy of the scheme.

Below are the lift and drag coefficients, Cl and Cd, computed on the body-fitted and body-unfitted Cartesian grids for NACA0012, M = 0.8, $\alpha = 1.25°$. The difference between their values is less than 1%.

Coefficient	Cartesian grid	C-grid
Cl	0.3012	0.3036
Cd	0.02184	0.02199

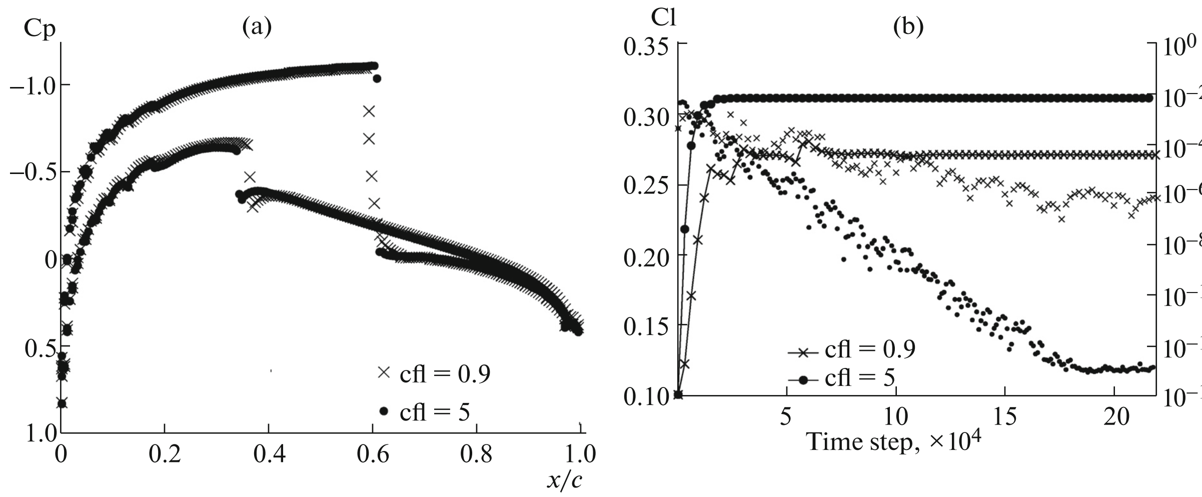

Fig. 3. (a) Cp distribution and (b) Cl as a function of time and the convergence rate for Courant numbers of 0.9 and 5 on a body-unfitted grid, NACA0012, $M = 0.8$, $\alpha = 1.25°$.

The computations on the body-unfitted Cartesian grid were also performed with different time steps. As a result, the well-known property of nonuniqueness of solutions to the Euler equations was detected. In this series of runs, we used the fully implicit scheme ($\omega = 0$) to eliminate the possible dependence of the stead-state solution on the time step. Figure 3a presents two Cp distributions obtained at Courant numbers of 0.9 and 5. The grid and the solution method are absolutely identical. It can be seen that these are two different solutions with different shock-wave positions and profiles (specifically, on the downstream side, the shock wave has a sharper profile for the solution with a Courant number of 5). Figure 3b displays the convergence of the residual and the lift coefficient. It can be seen that both solutions reach a steady state. The differences in the lift coefficient are about 15%. The nonuniqueness of the solution revealed in these computations was also observed and investigated in [20–22] in the case of transonic flows around airfoils.

The validity of the compensating flux was also tested as applied to the interaction of a supersonic gas flow (Mach number $M = 3$) with a wedge (10° aperture) in modes with the formation of a shock and a rarefaction wave. A uniform body-unfitted Cartesian grid with a resolution of 1200 × 480 cells was used. The Courant number was specified as $C = 4$. Below are the shock angle β and the angle γ within which a rarefaction fan is formed (aperture 10°, $M = 3$). The corresponding analytical values are also given for comparison purposes.

Numerical solution, $\beta°$	Analytical solution, $\beta°$	Numerical solution, $\gamma°$	Analytical solution, $\gamma°$
17.4	17.383	13.2	13.24

To verify the method, we computed the nonstationary problem of a shock wave interacting with a group of cylinders from [16]. The computation was performed on a uniform Cartesian grid consisting of 1024 × 1024 cells with the use of 32 GPUs. The numerical results are presented in Fig. 4b, which shows an instantaneous numerical schlieren image of the flow. For comparison, Fig. 4a displays the numerical results produced by a penalization method [16]. The solutions are similar, but the compensating flux method does not produce unphysical perturbations near the surfaces of the cylinders. This is explained by the fact that the penalization method has no subgrid-scale resolution of geometry; geometry in this method has a coarser representation up to a mesh cell.

Finally, we performed a validation computation of the three-dimensional flow over the DLR-F6 model [23] at the free-stream Mach number $M = 0.75$ and the angle of attack $\alpha = 1°$ on a grid consisting of 408 × 520 × 1256 cells. The computations were performed on 162 GPUs of the "Lobachevsky" supercomputer with the use of the hybrid explicit–implicit scheme with the Courant number $C = 4$. The numerical results are presented in Fig. 5. It should be noted that the computed lift coefficient was found to be somewhat higher than the experimental value $Cl_{exp} = 0.5$. A similar excess (of about 0.5–0.6) is also observed in numerical solutions obtained on body-fitted grids (see [23]) and is explained by the limitations of the Euler model.

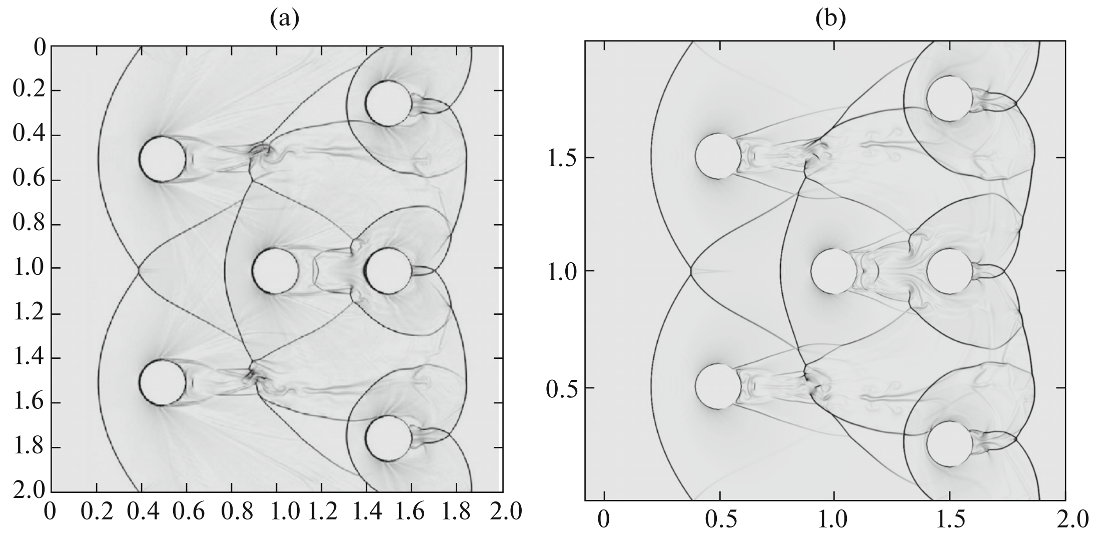

Fig. 4. Numerical visualization of the flow past a set of cylinders, M = 3: (a) penalization method [20] and (b) the present method of compensating fluxes.

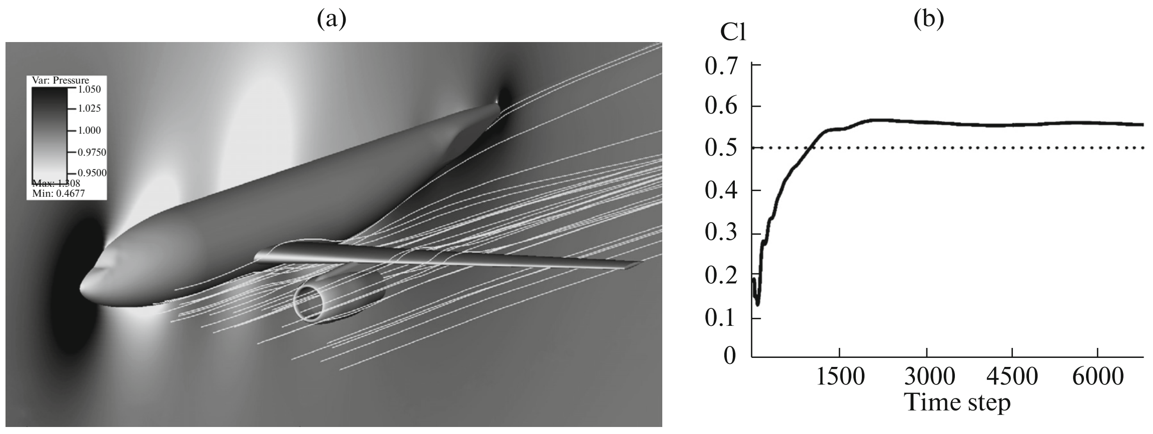

Fig. 5. DLR-F6 model, M = 0.75, $\alpha = 1°$: (a) pressure field in the plane of symmetry and streamlines and (b) the lift coefficient Cl vs. time.

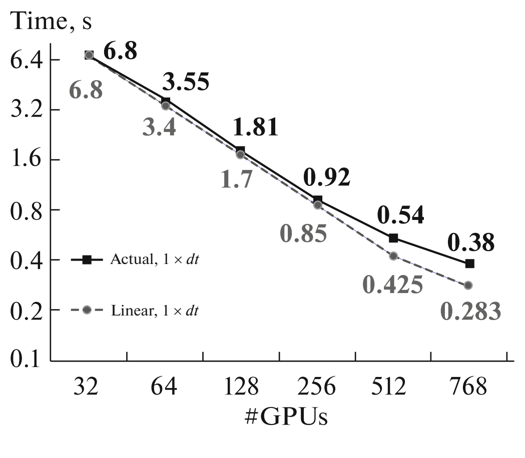

Fig. 6. Computation time vs. the number of GPUs; the dashed line shows linear scalability.

The overall performance of a large number of GPUs was analyzed as applied to a problem with a fixed grid of 150 million cells on the "Lomonosov" supercomputer. The code demonstrated good scalability to several hundred GPUs: the efficiency of the computation was 75% on 768 GPUs (Fig. 6). This result was achieved due to the original cell traverse scheme allowing for simultaneous computations and data exchange between the GPUs and due to its software implementation with the use of asynchronous CUDA and MPI functions.

CONCLUSIONS

An efficient numerical method was proposed for computing complex three-dimensional unsteady gas flows on simple body-unfitted Cartesian grids. The method is based on a hybrid explicit–implicit Godunov scheme ensuring the minimum involvement of the implicit component, while preserving the MND property. A method for implementing boundary conditions on a Cartesian grid was proposed that relies on introducing special compensating fluxes into the right-hand sides of the original equations. As a result, we passed from the boundary value problem in a bounded domain to equations with a compensating flux in the entire space; i.e., in fact, the boundary value problem was replaced by an initial value one. The discrete equations were solved by applying approximate LU-SGS factorization, which considerably simplified the algorithmic part of the method. This property (computational primitivism) made it possible to develop an efficient algorithm for parallel computations on hybrid CPU/GPU computing systems. The presented test results and comparison with alternative computations on body-fitted grids conclusively show the validity and good performance of the method, as well as its high scalability and efficiency.

The technique proposed is easy to formally extend to viscous flows. Viscous fluxes can be included in the baseline hybrid explicit–implicit scheme, for example, as described in [24], and the compensating flux in a mixed cell can be modified by adding viscous friction on the body surface. Here, the basic difficulty is to compute viscous friction at high Reynolds numbers, in which case the maximum possible (in terms of practical implementation) mesh adaptation to geometry cannot ensure the grid resolution required for the boundary-layer near-wall flow. In this case, special near-wall models will possibly have to be constructed and included in the numerical process (see, e.g., [25]). These issues are under development and will be addressed elsewhere.

ACKNOWLEDGMENTS

This work was supported by the Russian Science Foundation, project no. 14-11-00872. The part of this work concerning the adaptation of the algorithm to hybrid computing systems was supported by the Russian Foundation for Basic Research, project no. 14-01-31480.

REFERENCES

1. Lin Fu, Zhenghong Gao, Kan Xu, and Fang Xu, "A multi-block viscous flow solver based on GPU parallel methodology," Comput. Fluids **95**, 19–39 (2014).
2. A. Jameson and E. Turkel, "Implicit schemes and LU decomposition," Math. Comput. **37** (156), 385–397 (1981).
3. V. P. Kolgan, "The principle of minimal derivative values as applied to the construction of finite-difference schemes for the computation of discontinuous gas flows," Uch. Zap. Tsentr. Aerogidrodin. Inst. **3** (6), 68–77 (1972).
4. W. K. Anderson, J. L. Thomas, and B. van Leer, "Comparison of finite volume flux vector splitting for the Euler equations," AIAA J. **24** (9), 1453–1460 (1986).
5. B. van Leer, "Towards the ultimate conservative difference scheme: V. A second-order sequel to Godunov's method," J. Comput. Phys. **32**, 101–136 (1979).
6. J. E. Fromm, "A method for reducing dispersion in convective difference schemes," J. Comput. Phys. **3**, 176–187 (1968).
7. S. K. Godunov and V. S. Ryaben'kii, *Difference Schemes* (Nauka, Moscow, 1977) [in Russian].
8. G. D. van Albada, B. van Leer, and W. Roberts, "A comparative study of computational methods in cosmic gas dynamics," Astron. Astrophys. **108**, 76–84 (1982).
9. S. K. Godunov, "Difference method for computing discontinuous solutions of fluid dynamics equations," Mat. Sb. **47** (3), 271–306 (1959).
10. S. K. Godunov, A. V. Zabrodin, M. Ya. Ivanov, A. N. Kraiko, and G. P. Prokopov, *Numerical Solution of Multidimensional Problems in Gas Dynamics* (Nauka, Moscow, 1976) [in Russian].
11. E. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics* (Springer, Berlin, 2009).

12. V. V. Rusanov, "Third-order accurate shock-capturing schemes for computing discontinuous solutions," Dokl. Akad. Nauk SSSR **180**, 1303–1305 (1968).
13. I. Menshov and Y. Nakamura, "Hybrid explicit-implicit, unconditionally stable scheme for unsteady compressible flows," AIAA J. **42** (3), 551–559 (2004).
14. I. S. Menshov and M. A. Kornev, "Free-boundary method for the numerical solution of gas-dynamic equations in domains with varying geometry," Math. Model. Computer Simul. **6** (6), 612–621 (2014).
15. I. S. Menshov and P. V. Pavlukhin, Preprint No. 92, IPM RAN (Keldysh Inst. of Applied Mathematics, Russian Academy of Sciences, Moscow, 2014).
16. O. Boiron, G. Chiavassa, and R. Donat, "A high-resolution penalization method for large Mach number flows in the presence of obstacles," Comput. Fluids **38**, 703–714 (2009).
17. L. D. Landau and E. M. Lifshitz, *Fluid Mechanics* (Nauka, Moscow, 1986; Butterworth-Heinemann, Oxford, 1987).
18. I. Menshov and Y. Nakamura, "An implicit advection upwind splitting scheme for hypersonic air flows in thermochemical nonequilibrium," *Collection of Technical Papers of 6th International Symposium on CFD* (Lake Tahoe, Nevada 1995), pp. 815–821.
19. P. V. Pavlukhin, "Implementation of a parallel LU-SGS method for gas dynamic applications on cluster GPU systems," Vestn. Nizhegorod. Gos. Univ., No. 1, 213–218 (2013).
20. A. Jameson, "Airfoil admitting nonunique solutions to the Euler equations," AIAA Paper, No. 91-1625 (1991).
21. M. M. Hafez and W. H. Guo, "Nonuniqueness of transonic flows," Acta Mech. **138** (3), 177–184 (1999).
22. A. G. Kuzmin, "Instability and bifurcation of transonic flow over airfoils," AIAA Paper (2004).
23. K. R. Laflin, S. M. Klausmeyer, T. Zickuhr, et al., "Data summary from second AIAA computational fluid dynamics drag prediction workshop," J. Aircraft **42** (5), 1165–1178 (2005).
24. V. E. Borisov, A. A. Davydov, I. Yu. Kudryashov, A. E. Lutskii, and I. S. Menshov, "Parallel implementation of an implicit scheme based on the LU-SGS method for 3D turbulent flows," Math. Model. Computer Simul. **7** (3), 222–232 (2015).
25. A. E. Lutskii and A. V. Severin, Preprint No. 38, IPM RAN (Keldysh Inst. of Applied Mathematics, Russian Academy of Sciences, Moscow, 2013).

Translated by I. Ruzanova