ISSN 1995-0802, Lobachevskii Journal of Mathematics, 2018, Vol. 39, No. 4, pp. 503–512. © Pleiades Publishing, Ltd., 2018.

Parallel Versions of Implicit LU-SGS Method

A. Chikitkin[1*], **M. Petrov**[1**], **V. Titarev**[1,2***], and **S. Utyuzhnikov**[1,3****]

(Submitted by V. V. Voevodin)

[1]*Moscow Institute of Physics and Technology (State University), Dolgoprudny, Moscow oblast, 141700 Russia*

[2]*Federal Research Center "Computer Science and Control", Russian Academy of Sciences, Moscow, 119333 Russia*

[3]*University of Manchester, Manchester, M13 9PL UK*

Received November 12, 2017

Abstract—Different parallel distributed-memory versions of Lower-Upper Symmetric Gauss–Seidel (LU-SGS) method for solution of discrete equations in finite-volume framework are compared in terms of parallel structure of algorithms. New version based on multilevel recursive decomposition is proposed.

DOI: 10.1134/S1995080218040054

Keywords and phrases: *Finite-volume method, implicit scheme, computational fluid dynamics, LU-SGS method, graph partition, mesh reordering.*

1. INTRODUCTION

Modern supercomputers and parallel algorithms are widely used in computational fluid dynamics (CFD). Parallel structure of all algorithms is usually based on geometric decomposition of computational mesh. Interprocessor data exchange pattern heavily depends on the type of the numerical method used.

Explicit numerical schemes have relatively low computational cost and require minor changes to the serial version to make it parallel. In contrast, implicit numerical methods usually have large computational cost per time step and more complicated structure. Such methods cannot be parallelized straightforwardly. Nevertheless, implicit methods are popular in CFD due to their stability and robustness. That is why many studies are devoted to improving stability and scalability of parallel versions of implicit methods.

In present paper we consider and compare program structures of several parallel versions of Lower-Upper Symmetric Gauss–Seidel (LU-SGS) method used to solve discrete equations in implicit finite volume schemes applied to compressible flow equations [1–5].

The focus of the paper is on data structures and exchange patterns, which arise in different methods. Problems concerned with optimal mesh partition and rigorous convergence analysis are out of scope of this paper.

[*]E-mail: chikitkin.av@mipt.ru
[**]E-mail: petrov.mn@mipt.ru
[***]E-mail: titarev@ccas.ru
[****]E-mail: s.utyuzhnikov@manchester.ac.uk

2. SERIAL VERSION OF LU-SGS METHOD

We will consider the main features of the LU-SGS method as applied to the finite-volume method for 3D compressible Navier–Stokes equations written in conservative form

$$\frac{\partial}{\partial t}\mathbf{U} + \nabla(\mathbf{F} - \mathbf{G}) = \mathbf{0}, \quad \mathbf{F} = (\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3), \quad \mathbf{G} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3), \tag{1}$$

where the vectors of conservative variables $\mathbf{U}$, convective fluxes $\mathbf{F}_k$ and viscous fluxes $\mathbf{G}_k$ are given by

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ E \end{pmatrix}, \quad \mathbf{F}_k = \begin{pmatrix} \rho u_k \\ \rho u_1 u_k + \delta_{1k} p \\ \rho u_2 u_k + \delta_{2k} p \\ \rho u_3 u_k + \delta_{3k} p \\ (E+p) u_k \end{pmatrix}, \quad \mathbf{G}_k = \begin{pmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \\ u_\alpha \tau_{\alpha k} - q_k \end{pmatrix}.$$

Here t is time, $\mathbf{x} = (x_1, x_2, x_3)$ is the coordinates in the physical space, ρ is density, $\mathbf{u} = (u_1, u_2, u_3)$ is gas velocity, p is pressure, E is energy per unit volume of the gas mixture:

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u_1^2 + u_2^2 + u_3^2), \quad p = \rho R T,$$

where R is the gas constant. The components of the viscous stress tensor τ_{ij} and the heat flux q_k are determined by the expressions:

$$\tau_{ik} = \mu\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} - \frac{2}{3}\mathrm{div}\mathbf{u}\right), \quad q_k = -\lambda\frac{\partial T}{\partial x_k},$$

where T is temperature; μ and λ are viscosity and thermal conductivity coefficients, respectively.

To construct a stationary solution, we consider the implicit finite-volume Godunov-type method for solving (1). In the spatial variables, we introduce the computational mesh consisting of cells V_i. Each cell can be tetrahedral, pyramid-shaped, hexahedral, or prismatic; it can be formed by several triangular or quadrilateral faces a_{li}. By integrating (1) over a cell and passing to discrete form, we obtain the following semi-discrete equations

$$\frac{\partial}{\partial t}\mathbf{U}_i = \mathbf{R}_i, \quad \mathbf{R}_i = -\frac{1}{|V_i|}\sum_l \Phi_{li}, \quad 1 \le i \le N_{tot}, \quad \mathbf{\Phi}_{li} = \int\limits_{a_{li}} \mathbf{n}_l \cdot (\mathbf{F} - \mathbf{G}) da. \tag{2}$$

Here $\mathbf{U}_i$ is the cell average value, $\mathbf{\Phi}_{li}$ is the numerical flux for the face l, $\mathbf{n}_l$ is the outward unit normal to the face l.

Note that $\mathbf{R}_i \equiv \mathbf{0}$ for the stationary solution. The convective part of flux is computed numerically, the specific form of the scheme is irrelevant in the context of the present paper. To find the viscous part of the numerical flux, we directly approximate the derivatives using the values of the solution at the centers of the adjacent cells and at the vertices of the face. The unknown values at the face vertices are found by averaging over adjacent (with respect to the face) cells with the geometric weights proportional to the distance from the vertex to the cell center. This approach is an extension of [6].

By approximating the time derivative in the semi-discrete scheme (2) by backward differences, we obtain the following implicit one-step scheme on the arbitrary spatial mesh:

$$\frac{\Delta \mathbf{U}_i}{\Delta t} = \mathbf{R}_i^{n+1}, \quad \Delta \mathbf{U}_i = \mathbf{U}_i^{n+1} - \mathbf{U}_i^n.$$

The linearization with respect to time yields the following expression in the cell i:

$$\left(\mathbf{I} - \frac{\partial \mathbf{R}_i^n}{\partial \mathbf{U}}\right)\Delta \mathbf{U}_i = \Delta t \mathbf{R}_i^n. \tag{3}$$

While the numerical flux is linearized, we will take into account its dependence on the values of $\mathbf{U}$ in the cell i and its immediate neighbor $\sigma_l(i)$ on the face l, which corresponds to the upwind first-order spatial scheme. Next, to approximate the numerical fluxes on the left-hand side of scheme (3) the modified Rusanov flux is used [4]:

$$f(\mathbf{Q}^-, \mathbf{Q}^+) = \frac{1}{2}(\mathbf{F}(\mathbf{Q}^-) + \mathbf{F}(\mathbf{Q}^+)) - \frac{1}{2}\hat{v}(\mathbf{Q}^+ - \mathbf{Q}^-),$$

$$\hat{v} = \max_k(|\lambda_k(\mathbf{Q}^-)|, |\lambda_k(\mathbf{Q}^+)|) + \frac{(\mu^- + \mu^+)}{(\rho^- + \rho^+)h},$$

here λ_k are the eigenvalues of the Jacobian matrix of the convective flux and h "— is the distance between the cell centers. After some transformations, the implicit scheme (3) takes the final form

$$D_i\Delta\mathbf{U}_i + \frac{\Delta t}{2|V_i|}\sum_l \left(T_{li}^{-1}\Delta\mathbf{F}_{li} - \hat{v}_{li}\Delta\mathbf{U}_{\sigma_l(i)}\right)|a_{li}| = \Delta t\mathbf{R}_\mathbf{i}^\mathbf{n}$$

$$D_i = 1 + \frac{\Delta t}{2|V_i|}\sum_l \hat{v}_{li}|a_{li}|, \quad \Delta\mathbf{F}_{li} = \mathbf{F}_1(T_{li}\mathbf{U}_{\sigma_l(i)} + T_{li}\Delta\mathbf{U}_{\sigma_l(i)}) - \mathbf{F}_1(T_{li}\mathbf{U}_{\sigma_l(i)}), \tag{4}$$

where T_{li} is the rotational matrix of the reference frame to the local reference frame for the face l of the cell V_i, see [7]. The resulting sparse system of linear equations (4) can be solved using the approximate LU-SGS factorization [1–4].

The system of linear equations (4) can be written in symbolic form as

$$A\Delta\mathbf{U} = \mathbf{b}. \tag{5}$$

The diagonal of the matrix A contains the quantities D_i and all the off-diagonal nonzero elements have the order Δt. We decompose this matrix into the diagonal one, the strictly lower triangular and strictly upper triangular matrices: $A = L + D + U$. Instead of system (5) we consider the system

$$(D + U)D^{-1}(D + L)\Delta\mathbf{U} = \mathbf{b} + \mathbf{S}, \quad \mathbf{S} = LD^{-1}U\boldsymbol{\Delta}\mathbf{U}. \tag{6}$$

The matrix of this system is the product of the lower triangular, diagonal, and upper triangular matrices. The quantity $\mathbf{S} = O(\Delta t^2)$ is typically neglected. By setting $\mathbf{S} = \mathbf{0}$ we obtain a simplified system of equations, which is easily solved using the forward and backward elimination.

Backward elimination is given by ($i = N_{tot}, N_{tot} - 1, \dots, 1$)

$$D_i\Delta\mathbf{U}_i^* = -\frac{\Delta t}{2|V_i|}\sum_{l:\sigma_l(i)>i}\left(T_{li}^{-1}\Delta\mathbf{F}_{li} - \hat{v}_{li}\Delta\mathbf{U}_{\sigma_l(i)}^*\right)|a_{li}| + \Delta t\mathbf{R}_i^n. \tag{7}$$

Forward elimination is given by ($i = 1, 2, \dots, N_{tot}$)

$$D_i\Delta\mathbf{U}_i = \Delta\mathbf{U}_i^* - \frac{\Delta t}{2|V_i|}\sum_{l:\sigma_l(i)<i}\left(T_{li}^{-1}\Delta\mathbf{F}_{li} - \hat{v}_{il}\Delta\mathbf{U}_{\sigma_l(i)}\right)|a_{li}|. \tag{8}$$

The numerical solution is regarded as convergent to the stationary solution if the norm of the vector on the right-hand side becomes less than a prescribed small tolerance.

It is obvious that result in (8), (7) depends on cell's order. Different reordering techniques can be used for reducing cache misses and accelerating convergence even in serial case, but we will not consider reordering problem in the present paper.

In program implementation a single array is usually used for both $\Delta\mathbf{U}_i^*$ and $\Delta\mathbf{U}_i$. Pseudo-code of the serial algorithm is the following:

```
do while (...)
    Residual calculation and other stuff
    1. backward elimination loop
for $i = N_{tot}, N_{tot} - 1, \dots, 1$
        $D_i\Delta\mathbf{U}_i = -\frac{\Delta t}{2|V_i|}\sum_{l:\sigma_l(i)>i}\left(T_{li}^{-1}\Delta\mathbf{F}_{li} - \hat{v}_{li}\Delta\mathbf{U}_{\sigma_l(i)}\right)|a_{li}| + \Delta t\mathbf{R}_i^n$
```

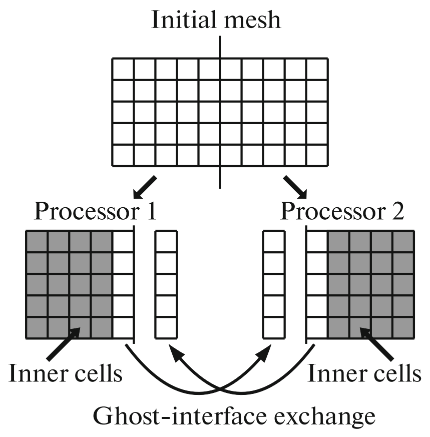


Fig. 1. Partition into two blocks, arrows show ghost cell exchange.

end for

2. *forward elimination loop*

for $i = 1, \dots, N_{tot}$

$$D_i \Delta \mathbf{U}_i = \Delta \mathbf{U}_i - \frac{\Delta t}{2|V_i|} \sum_{l:\sigma_l(i)<i} \left(T_{li}^{-1} \Delta \mathbf{F}_{li} - \hat{v}_{il} \Delta \mathbf{U}_{\sigma_l(i)} \right) |a_{li}|$$

end for

3. PARALLEL VERSIONS OF LU-SGS METHOD

For the sake of simplicity we consider the 1st order scheme. In this case stencil of each cell consists of all neighboring cells, which share face with current cell. All parallel versions described below are based on mesh partition into P non-overlapping blocks, where P is the number of processors. We will use the following notation in order to present pseudo-code for all algorithms in a single form:

- C is set of all cells in the computational mesh. We will also use C for set of all indices assuming some numeration;
- $|C|$ is number of elements in set C;
- $C_p(own)$ is set of cells belonging to block (processor) p, which is obtained from some non-overlapping partition algorithm;
- $C_p(ghost)$ is set of ghost-cells of block(processor) p, e.g. cells from other blocks, which are contained in stencil of cell from block p. For 1st order scheme "— all neighbors of cell from block p which are not in block p;
- $C_p(inner)$ is subset of cells in $C_p(own)$, such that stencil for cell (set of all neighbors) are $\subset C_p(own)$;
- $C_p(interface)$ is subset of cells in C_p, such that at least one neighbor of cell $\notin C_p$. It means that cells in $C_p(interface)$ are ghost cells for some other blocks;
- $C_p(all) = C_p(own) \cup C_p(ghost)$.

Partition into two blocks is shown schematically on Fig. 1. Arrows show exchange of values in ghost (interface) cells.

Here and below we assume that in each set of cells some local ordering of cells is introduced. For instance, pseudo-code "*for* $i \in C_p \dots$" will denote loop over cells according to this local ordering.

MPI is the standard tool for implementation of distributed-memory algorithm. We will use pseudo-MPI commands like *Send, Receive, Wait* in algorithm descriptions omitting details for readability.

3.1. Naive Parallel Version

The most straightforward simplification of the serial algorithm that allows to extend LU-SGS method to distributed-memory case is the following: perform loops (7), (8) for each mesh block separately setting $\mathbf{\Delta U}_{\sigma_l(i)} = 0$ for all neighbors from other mesh blocks.

Using notation described above algorithm can be represented as follows:

```
p = my_rank
do while (...)
*Residual calculation and other stuff*
1. *loop over ghost cells*
for $i_p \in C_p(ghost)$ $\Delta U_i = 0$ end for
2. *backward elimination, loop over cells* $\in C_p(own)$
for $i = |C_p(own)|, |C_p(own)| - 1, \ldots, 1$
    $$D_i \Delta\mathbf{U}_i = -\frac{\Delta t}{2|V_i|} \sum_{l:\sigma_l(i)>i} \left( T_{li}^{-1} \Delta\mathbf{F}_{li} - \hat{v}_{li} \Delta\mathbf{U}_{\sigma_l(i)} \right) |a_{li}| + \Delta t \mathbf{R}_i^n$$
end for
3. *forward elimination, loop over cells* $\in C_p(own)$
for $i = 1, \ldots, |C_p(own)|$
    $$D_i \Delta\mathbf{U}_i = \Delta\mathbf{U}_i - \frac{\Delta t}{2|V_i|} \sum_{l:\sigma_l(i)<i} \left( T_{li}^{-1} \Delta\mathbf{F}_{li} - \hat{v}_{il} \Delta\mathbf{U}_{\sigma_l(i)} \right) |a_{li}|$$
end for
end while
```

This naive parallel modification of LU-SGS method needs no data exchange beetween different processors (although we indeed need to exchange data to compute residual $\mathbf{R}(\mathbf{U})$). However, convergence rate remains good only for small number of processors and mesh blocks (when interface cells in block make up only for small fraction with respect to inner cells). Otherwise, convergence may slow down, or iterations may indeed diverge [5].

3.2. Jacobi Corrections

Instead of nullifying $\mathbf{\Delta U}_i$ in ghost cells we may use some approximations. One possible way is to utilize values $\mathbf{\Delta U}_i^{(0)} = D_i^{-1} \Delta t \mathbf{R}_i^n$. We may interpret this as one iteration of Jacobi method for system (4) with initial guess $\mathbf{\Delta U}_i = 0$. That is why we refer to procedure as Jacobi corrections. After LU-SGS step one can exchange $\mathbf{\Delta U}_i$ in boundary cells and perform corrections iteratively [5]. Algorithm looks as follows:

```
p = my_rank
do while(...)
*Residual calculation and other stuff*
for $i \in C_p(ghost)$
$\Delta\mathbf{U}_i = D^{-1} \Delta t R_i^n$
end for
for $m = 1, \ldots, M$ (*Jacobi sweeps*)
1. *Exchange boundary values*
for $i \in C_p(interface)$
Send $\Delta\mathbf{U}_i$ to neighbors
end for
for $i \in C_p(ghost)$
Receive $\Delta\mathbf{U}_i$ from neighbors
```

```
end for
Wait all
2. *backward elimination loop over cells* $\in C_p(own)$
for $i = |C_p(own)|, |C_p(own)| - 1, \dots, 1$
```

$$D_i \Delta \mathbf{U}_i = -\frac{\Delta t}{2|V_i|} \sum_{l:\sigma_l(i)>i} \left(T_{li}^{-1} \Delta \mathbf{F}_{li} - \hat{v}_{li} \Delta \mathbf{U}_{\sigma_l(i)} \right) |a_{li}| + \Delta t \mathbf{R}_i^n$$

```
3. *forward elimination loop over cells* $\in C_p(own)$
for i = $1, \dots, |C_p(own)|$
```

$$D_i \Delta \mathbf{U}_i = \Delta \mathbf{U}_i - \frac{\Delta t}{2|V_i|} \sum_{l:\sigma_l(i)<i} \left(T_{li}^{-1} \Delta \mathbf{F}_{li} - \hat{v}_{il} \Delta \mathbf{U}_{\sigma_l(i)} \right) |a_{li}|$$

```
end for
end while
```

Jacobi corrections improve robustness of parallel LU-SGS algorithm, but for large number of processors convergence rate is close to that of Jacobi method, which is very low.

3.3. Mesh-Reordering for Structured Mesh

Two parallel algorithms considered above differ slightly from serial algorithm. More complicated versions are usually based on some additional mesh decomposition and reordering.

Obvious strategy for structured mesh is to apply a graph coloring to initial mesh. It yields partition of cells such that all neighbours of cell have different color. In the case of hexahedral mesh one can use simple "chess board" red-black coloring, see paper [8]. Let us denote set of cells of color k as C^k (and C_p^k correspondingly). As before we assume that local numbering is introduced inside each subset C_p^k.

Given the colored mesh we can perform standard partition into blocks and then process cells of one color in parallel with data exchanges for each color. Such operation order guarantees that parallel algorithm is equivalent to serial one up to corresponding reordering of cells. In detail:

```
p = my_rank
do while(...)
*Residual calculation and other stuff*
for $k = K, K-1, \dots, 1$ (*loop over all colors in inverse order*)
1. *backward elimination loop over cells* $\in C_p^k(own)$
for $i = |C_p^k(own)|, |C_p^k(own)| - 1, \dots, 1$ ((loop in inverse order))
```

$$D_i \Delta \mathbf{U}_i = -\frac{\Delta t}{2|V_i|} \sum_{l:\sigma_l(i)>i} \left(T_{li}^{-1} \Delta \mathbf{F}_{li} - \hat{v}_{li} \Delta \mathbf{U}_{\sigma_l(i)} \right) |a_{li}| + \Delta t \mathbf{R}_i^n$$

```
end for
2. *Exchange values in ghost cells*
for $i \in C_p^k(ghost)$
Receive $\mathbf{\Delta U}_i$ values from neighbors
end for
for $i \in C_p^k(interface)$
Send $\mathbf{\Delta U}_i$ values to neighbors
end for
Wait all
end for
for $k = 1, \dots, K$ (*loop over all colors*)
1. *forward elimination loop over cells* $\in C_p^k(own)$
```

```
for $i = 1, \ldots, |C_p^k(own)|$
    $$D_i \Delta\mathbf{U}_i = \Delta\mathbf{U}_i - \frac{\Delta t}{2|V_i|} \sum_{l:\sigma_l(i)<i} \left(T_{li}^{-1} \Delta\mathbf{F}_{li} - \hat{v}_{il} \Delta\mathbf{U}_{\sigma_l(i)}\right) |a_{li}|$$
2. *Exchange values in ghost cells*
for $i \in C_p^k(ghost)$
    Receive $\mathbf{\Delta U}_i$ values from neighbors
end for
for $i \in C_p^k(interface)$
    Send $\mathbf{\Delta U}_i$ values to neighbors
end for
Wait all
end for
```

Approach based on graph coloring is universal an can be extended, in principle, to the general unstructured mesh. Yet the practical implementation faces following possible difficulties:

1. For large 3D unstructured mesh of general form consisting of tetrahedrons, hexahedrons, pyramids and prisms graph coloring require large computational effort;

2. In the case mentioned it is difficult to achieve well balancing. Cells of each color have to be "uniformly distributed" over all mesh so that all mesh blocks contain approximately equal number of cells of each color. Otherwise, parallel efficiency will be low due to bad balancing.

4. MULTI-PROCESSOR LU-SGS IMPLEMENTATION BASED ON MULTILEVEL MESH DECOMPOSITION

Another way to make parallel algorithm equivalent to serial one is to introduce additional recursive or hierarchical decomposition of mesh. In [9] we proposed a parallel LU-SGS algorithm based on recursive mesh decomposition which appeared very convenient for shared-memory realization. It has much in common with parallel nested dissection algorithm [10]. Here we describe its generalization to distributed-memory case.

The first part of algorithm is a preliminary recursive partition. Let us denote number of partition levels as K. Partition procedure is the following:

```
$C^1 = C$
for $k = 1, \ldots, K$
    1. Partition $C^k$ into $p$ blocks $C_1^k(own), \ldots, C_p^k(own)$
    2. For each block compute $C_p^k(inner)$, $C_p^k(interface)$, $C_p^k(ghost)$
    3. $C^{k+1} = \bigcup_p C_p^k(interface)$
end for
```

It is worth mentioning that at all of the steps we take into account connectivity with cells from current level C^k only, i.e. consider cutted mesh graph. As an example, sets C^2, C^3, C^4 for mesh partition into 24 blocks are shown in Fig. 2.

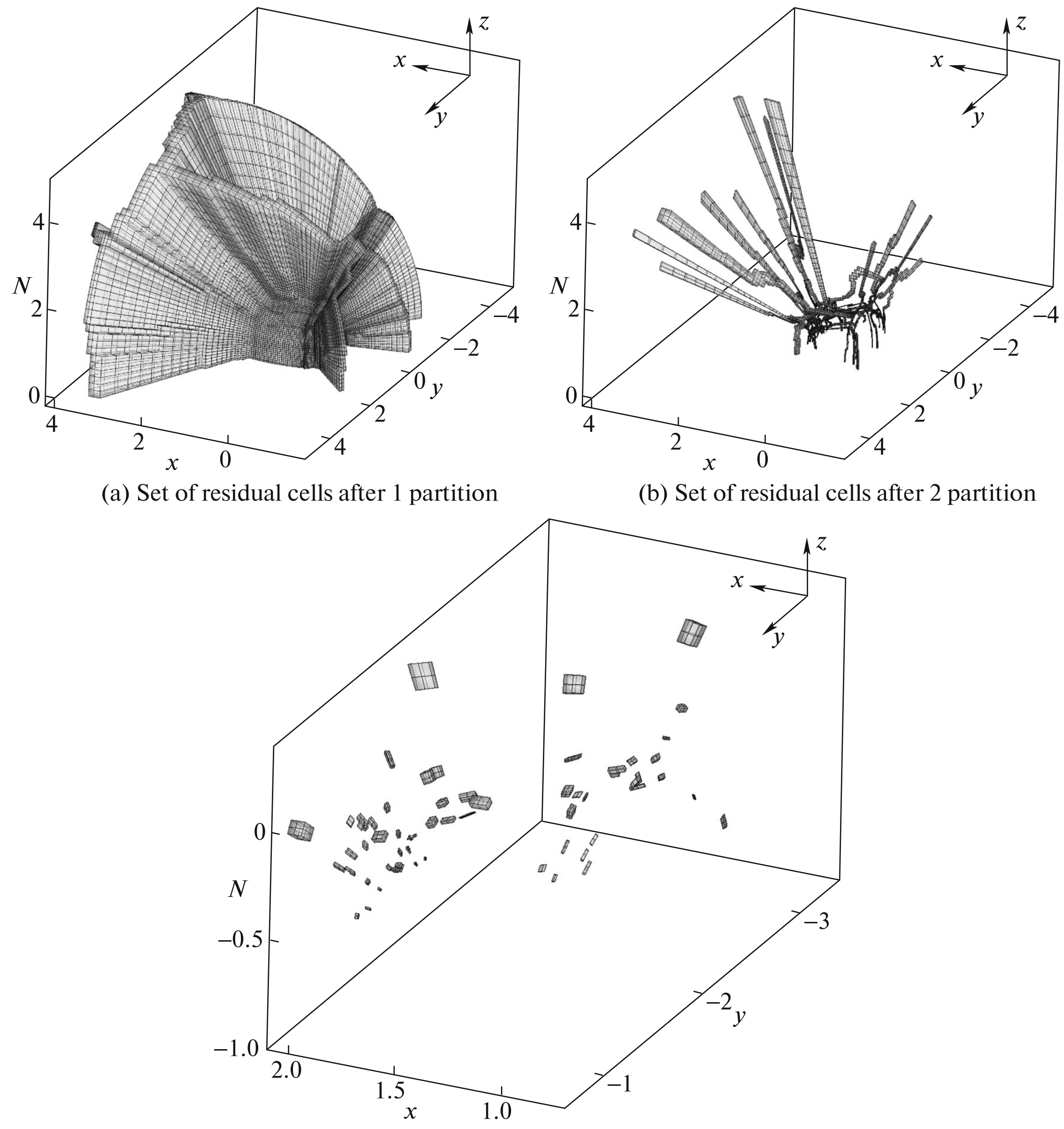
After partition is carried out we implicitly reorder cells in the following order:

$$C_1^1, C_2^1, \ldots, C_{p_{max}}^1, C_1^2, C_2^2, \ldots, C_{p_{max}}^2, \ldots, C_{p_{max}-1}^K, C_{p_{max}}^K$$

Cell ordering in each C_p^k set is arbitrary.

Given such multilevel partition the LU-SGS algorithm has the following form (for brevity we present only backward elimination):

```
p = my_rank
do while (...)
```



(a) Set of residual cells after 1 partition (b) Set of residual cells after 2 partition

(c) Set of residual cells after 3 partition

Fig. 2. Set of residual cells after 1, 2 and 3 levels of partition. Initial mesh contains about 10^6 cells.

```
Residual calculation and other stuff
if $p = 0$ (Serial part - executed by one process)
1. for $i = |C^{K+1}|, |C^{K+1}| - 1, \ldots, 1$

$$D_i \Delta \mathbf{U}_i = -\frac{\Delta t}{2|V_i|} \sum_{l:\sigma_l(i)>i} \left( T_{li}^{-1} \Delta \mathbf{F}_{li} - \hat{v}_{li} \Delta \mathbf{U}_{\sigma_l(i)} \right) |a_{li}| + \Delta t \mathbf{R}_i^n$$

end for
2. for $i \in C^{K+1}$ (Send computed values to the previous level)
Send $\mathbf{\Delta U}_i$
end for
end if
```

```
    *Receive interface values*
    for $i \in C_p^K(interface)$
    Receive $\mathbf{\Delta U}_i$
    end for
    Wait all
    for $k = K, K-1, \ldots, 1$ (*loop over all levels in inverse order*)
    1. *backward elimination loop over cells* $\in C_p^k(inner)$
    for $i = |C_p^k(inner)|, |C_p^k(inner)| - 1, \ldots, 1$
        $$D_i \Delta \mathbf{U}_i = -\frac{\Delta t}{2|V_i|} \sum_{l:\sigma_l(i)>i} \left(T_{li}^{-1} \Delta \mathbf{F}_{li} - \hat{v}_{li} \Delta \mathbf{U}_{\sigma_l(i)}\right) |a_{li}| + \Delta t \mathbf{R}_i^n$$
end for
    2. *Send computed values to the previous level*
    for $i \in C_p^k$
    Send $\mathbf{\Delta U}_i$
    end for
    for $i \in C_p^{k-1}(interface)$
    Receive $\mathbf{\Delta U}_i$
    end for
    Wait all
    end for
    *Forward elimination ...*
    end while
```

The proposed algorithm is equivalent to the serial LU-SGS algorithm applied to the renumbered (reordered) mesh. That is why one may hope that all advantages of the original LU-SGS method are preserved.

Besides, implementation of this algorithm requires only one graph partition procedure, which have to applied at each level of decomposition. That fact guarantees well-balancing of algorithm for meshes of any type. In our experiments we used widespread Metis library [11] and ordering-by-adjacency procedure described in [12].

5. CONCLUSION

We considered different parallel versions of LU-SGS method for solution of the large system of algebraic equations which arise in finite-volume framework for computational fluid dynamics. The parallel algorithms was compared in terms of data-structures, data exchanges and implementation complexity.

ACKNOWLEDGMENTS

The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

REFERENCES

1. S. Yoon and A. Jameson, *Lower-Upper Symmetric-Gauss–Seidel Method for the Euler and Navier Stokes Equations*, AIAA J. **26** (9), 1025–1026 (1988).
2. I. S. Men'shov and Y. Nakamura, *An implicit advection upwind splitting scheme for hypersonic air flows in thermochemical nonequilibrium*, A Collection of Technical Papers of 6th Int. Symp. on CFD **2**, 815 (1995).
3. I. S. Men'shov and Y. Nakamura, *On implicit Godunov's method with exactly linearized numerical flux*, Computers and Fluids **29** (6), 595–616 (2000).
4. D. Sharov, H. Luo, J. D. Baum, and R. Löhner, *Implementation of unstructured grid GMRES+LU-SGS method on shared-memory, cache-based parallel computers*, AIAA-2000-927 – Aerospace Sciences Meeting and Exhibit (38), 10–13 (2000).
5. A. M. Wissink, A. S. Lyrintzis, and R. C. Strawn, *Parallelization of a three-dimensional flow solver for Euler rotorcraft aerodynamics predictions*, AIAA J. **34** (11), 2276–2283 (1996).
6. C. R. Mitchell, *mproved reconstruction schemes for the Navier-Stokes equations on unstructured meshes*, AIAA-94-0642, 1994.
7. E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics* (Springer-Verlag, Berlin, 2009).
8. I. Menshov and P. Pavlukhin, *Highly scalable implementation of an implicit matrix-free solver for gas dynamics on GPU-accelerated clusters*, J. of Supercomputing **73** (2), 631–638 (2017)
9. M. N. Petrov, V. A. Titarev, S. V. Utyuzhnikov, and A. V. Chikitkin, *A multithreaded OpenMP implementation of the LU-SGS method using the multilevel decomposition of the unstructured computational mesh*, Computational Mathematics and Mathematical Physics **57** (11), 1856–1865 (2017).
10. V. Pan and J. Reif, *Efficient parallel solution of linear systems*, Proceedings of the 17th Annual ACM Symposium on Theory of Computing, 143–152 (1985).
11. G. Karypis and V. Kumar, *Multilevel k-way Partitioning Scheme for Irregular Graphs*, Parallel Distrib. Comput. **48**, 96–129 (1998).
12. I. E. Kaporin and O. Yu. Milyukova, *The massively parallel preconditioned conjugate gradient method for the numerical solution of linear algebraic equations*, Collection of Papers of the Department of Applied Optimization of the Dorodnicyn Computing Center, 132–157 (2011).