Fully implicit method with lower upper symmetric Gauss–Seidel preconditioner on multiple graphics processing units for hypersonic flows

A N Bocharov¹, V A Bityurin¹, N M Evstigneev², V E Fortov¹, V P Petrovskiy¹, O I Ryabkov¹,² and I O Teplyakov¹

¹ Joint Institute for High Temperatures of the Russian Academy of Sciences, Izhorskaya 13 Bldg 2, Moscow 125412, Russia
² Federal Research Center “Computer Science and Control” of the Russian Academy of Sciences, Vavilova 44 Bldg 2, Moscow 119333, Russia

E-mail: bocharov@ihed.ras.ru

Abstract. In the paper, we present implementation and benchmarking for gas dynamic solver for hypersonic flow problems with fully implicit scheme. The aim of the paper is to verify and measure efficiency of the implicit method for computational architecture based on multiple graphics processing units. The verification is performed on a flow over the sphere-cone problem. We consider hypersonic flow regimes of Mach number around 20–30. New results are cross-compared with results of the previous version of the code for explicit time marching scheme.

1. Introduction

Testing and use of completely implicit methods for supersonic compressible flows is implemented in many papers, e.g., in [1]. Such an implementation for a heterogeneous multi-core computing architecture is a complex task. At each step of the Newton’s method, it is very difficult to solve a poorly conditioned linear system. Such solutions techniques usually require Krylov subspace based methods and sophisticated preconditioning techniques (factorization coupled with multigrid methods, etc). One of the most efficient ways to solve linear systems for higher order finite volume and discontinuous Galerkin discretization for supersonic flows is the usage of incomplete lower upper (ILU) factorization or lower upper symmetric Gauss–Seidel (LU-SGS) preconditioners [1–3]. In this paper we are focusing on the latter approach tuned to be efficient for hypersonic flow problems on the computational architecture with multiple general purpose graphics processing units (GPUs). Such approach for structured grid is suggested in very good papers, e.g., [4], where LU-SGS matrix-free approach for multiple GPUs is implemented very efficiently using block-coloring method (block-matrix system is considered and for each GPU red-black block-coloring is solved). Unfortunately, such approach becomes less efficient for unstructured grids and hypersonic flows, where graph coloring leads to the change form Gauss–Seidel method to weighted Jacobi-type method and benefits of Gauss–Seidel preconditioning are diminished. This paper is the continuation of our work [5], where we introduced coupled solver and tested implicit method with ILU and LU-SGS preconditioners for a single GPU. Now a LU-SGS preconditioning is adopted for multiple GPUs.
The paper is laid out as follows. First of all, we are considering a formulation of the Newton’s method for stationary and non-stationary solution strategies and linear systems for the desired strategy is constructed. Then we discuss the method of LU-SGS preconditioner formulation for multiple GPUs. Finally, we are considering a hypersonic benchmark problem of a flow around the sphere-cone shaped object.

2. Problem formulation

The governing equations for compressible viscous perfect gas are defined as

\[
\begin{align*}
(\rho)_t + \nabla \cdot (\rho \mathbf{u}) &= 0, \\
(\rho \mathbf{u})_t + \nabla \cdot (\mathbf{u} \otimes (\rho \mathbf{u})) + \nabla p &= \nabla \cdot \mathbf{G}, \\
(E)_t + \nabla \cdot (\mathbf{u}(E + p)) &= \nabla \cdot \mathbf{G} + \nabla (\mathbf{H} \cdot \mathbf{u}).
\end{align*}
\]

Here, \( t \) is time; \( (\cdot)_t \) is a time derivative; operation \( \cdot \) is a dot product, \( \mathbf{u} \) is a velocity vector; \( \rho \) is a gas density; \( E = \mathbf{u}^2/2 + e \) is a total gas energy, \( e = C_v T \) is an internal gas energy; \( p = \rho RT \) is a pressure; \( T \) is a gas temperature; \( R \) is a gas constant; \( \mathbf{H} \) is a viscous stress tensor with dynamic viscosity defined by the Sutherland’s law; \( \mathbf{G} = k \nabla T \) is a heat flux; \( k \) is gas heat conductivity coefficient; \( C_v \) is the heat capacity at constant volume; operation \( \otimes \) designates tensor product.

The computational domain is designated as \( \Omega \).

Assume that the discretization procedure is applied to (1) and equations are translated into the discrete form for the given tessellation of \( \Omega \). This procedure (including used numerical schemes and Riemann solvers) is presented in [6]. Let us rewrite discrete analog of equations (1) for all mesh elements in the vector form, grouping conservative variables into vector \( \mathbf{U} \), inviscid and viscous parts into vector \( \mathbf{H} \) (operator \( \nabla_h \)):

\[
\mathbf{U}_t - \nabla_h \cdot (\mathbf{H}(\mathbf{U})) = 0,
\]

and we use notation \( \mathbf{F}(\mathbf{U}) \triangleq \nabla_h \cdot (\mathbf{H}(\mathbf{U})) \) for brevity. Sign \( \triangleq \) means “equal to by definition”.

Time explicit method for solving the problem (2) is described in [6] and implicit Euler’s method for a single GPU in [5].

For time dependent calculations we use implicit second order time stepping method with time step size designated as \( \tau \in \mathbb{Q} \) and time slices \( n \) and \( n + 1 \). Then (2) is rewritten as

\[
\mathbf{U}^{n+1} - \mathbf{U}^n = \tau/(2) (\mathbf{F}(\mathbf{U}^{n+1}) + \mathbf{F}(\mathbf{U}^n)).
\]

We introduce nonlinear iterations \( r \) as

\[
\mathbf{U}^{n+1,r+1} = \mathbf{U}^{n+1,r} + \delta \mathbf{U}.
\]

Plugging (4) into (3) and linearization near \( \mathbf{U}^{n+1,r} \) leads to the following Newton–Raphson iterations:

\[
\begin{align*}
\mathbf{U}^{n+1,r+1} &= \mathbf{U}^{n+1,r} + \delta \mathbf{U}, \\
\delta \mathbf{U} &= \left( \mathbf{U}^n - \mathbf{U}^{n+1,r} \right)/(\tau) + (\mathbf{F}(\mathbf{U}^{n+1,r}) + \mathbf{F}(\mathbf{U}^n))/(2),
\end{align*}
\]

where \( \mathbf{E} \) is an identity matrix and \( \mathbf{J} \triangleq (\partial \mathbf{F}(\mathbf{U}))/\partial \mathbf{U} \bigr|_{\mathbf{U}^{n+1,r}} \) is the Jacobi matrix.

Let the tolerance parameter \( 0 < \varepsilon \ll 1 \) is provided. Iterations stop after a certain criterion is met, e.g., \( \|\delta \mathbf{U}\| \leq \varepsilon \) and \( \|\mathbf{U}^n - \mathbf{U}^{n+1,r} + \tau/(2) \left( \mathbf{F}(\mathbf{U}^{n+1,r}) + \mathbf{F}(\mathbf{U}^n) \right)\| \leq \varepsilon \) or a maximum number of iterations is reached. If the latter criterion is triggered, than we decrease the value of \( \tau \) and restart iterations. These convergence strategies are usually used for Newton’s methods, see [5] for example. We use weighted norms here, relative to the initial condition vector norm.

For steady state calculations we use implicit Euler’s method and time step size becomes a homotopy parameter that changes from one iteration to another. Then the discrete system is rewritten as

\[
\mathbf{U}^{n+1} - \mathbf{U}^n = \tau^n \mathbf{F}(\mathbf{U}^{n+1}),
\]
with the following Newton–Raphson process:

\[
\begin{align*}
(E/\tau^n - J) \delta U &= F(U^n), \\
U^{n+1} &= U^n + \delta U, \quad n = 1, 2, \ldots ,
\end{align*}
\]  

(7)

where \( J \triangleq (\partial F(U)) / (\partial U) \) \( U^n \) is the Jacobi matrix. Iterations are finished when \( \| F(U^n) \| \leq \varepsilon \).

Time step size \( \tau^n \) has no physical meaning and is selected to insure faster convergence of iterations.

For both iteration processes we can introduce two stages on every step. The first step is the solution of the linear system. The second step is the update of the conservative vector. Thus the Newton’s iterative method is formed. Let us consider the general form of the linear system

\[ \mathbf{A} \mathbf{x} = \mathbf{b}, \]

(8)

where \( \mathbf{A} \triangleq (E/(\tau^n) - \omega \mathbf{J}^*) \) and \( \mathbf{b} \triangleq \alpha (\mathbf{U}^n - \mathbf{U}^{n+1,r})/(\tau) + \omega \mathbf{F(U}^{n+1,r}) + \omega \mathbf{F(U}^n) \), with \( \omega = 1/2 \), \( \alpha = 1 \) for (5) and \( \omega = 1, \alpha = 0 \) for (7). The Jacobi matrix \( \mathbf{J}^* \) is the simplification of the exact Jacobi matrix in terms that off-diagonal viscous tensor elements are neglected when the matrix is assembled during differentiation. We also need a matrix \( \tilde{\mathbf{A}} \) which is a simplification of the matrix \( \mathbf{A} \) where a first order reconstruction is used in the Jacobian. The latter matrix is used for preconditioning.

### 3. Solution of the linear systems

#### 3.1. Formulation of the linear system and the preconditioner

The solution of the linear system (8) is performed using either biconjugate gradient stabilized method (BiCGStab) or generalized biconjugate gradient stabilized method with Krylov biorthogonal basis size \( l \) (BiCGStab(\( l \))) methods, where a Krylov subspace is formed by calling a matrix–vector product. Each of these methods uses only matrix-vector products or dot product operations that are executed on each GPU and then calculated using message passing interface (MPI) reduce operation. The unpreconditioned Krylov method forms a residual vector on each iteration as

\[ \mathbf{r} = \mathbf{A} \mathbf{x} - \mathbf{b}, \]

(9)

which we call a true residual. However, the residual vector for the preconditioned system is different. On each call to matrix-vector operator of the Krylov method a preconditioned residual \( \mathbf{r} \) is considered. It is done in two steps:

\[
\begin{align*}
\mathbf{z} &= \mathbf{Ax} - \mathbf{b}, \\
\mathbf{r} &= \tilde{\mathbf{A}}^{-1} \mathbf{z}.
\end{align*}
\]

(10)

(11)

This residual is no longer directly related to the original system. In order to make sure that we achieved the convergence of the original problem we use the true residual (9) recalculation every 10-th iteration of the Krylov methods. This allows us to control the true residual and accelerates convergence for single precision floating-point arithmetic.

The main difficulty is the preconditioning step. We apply LU-SGS factorization [7] for the preconditioner matrix which is basically two triangular sparse system solutions, one is for the lower triangular system and one is for the upper triangular system. First we split the matrix on lower triangular, diagonal and upper triangular parts as \( \tilde{\mathbf{A}} = \mathbf{L} + \mathbf{D} + \mathbf{U} \). Observe, that \( \tilde{\mathbf{A}} = (\mathbf{L} + \mathbf{D})\mathbf{D}^{-1}(\mathbf{U} + \mathbf{D}) - \mathbf{LD}^{-1}\mathbf{U} \). We can then apply approximate factorization omitting the \( \mathbf{LD}^{-1}\mathbf{U} \) term (according to the original LU-SGS method [7]):

\[
\begin{align*}
\mathbf{z}^* &= (\mathbf{L} + \mathbf{D})^{-1} \mathbf{z}, \\
\mathbf{r} &= (\mathbf{U} + \mathbf{D})^{-1} \mathbf{D} \mathbf{z}^*.
\end{align*}
\]

(12)

(13)

Thus a preconditioning is formulated.
Table 1. Drag $C_x$ and lift $C_y$ coefficients as functions of Mach number; $C^{\text{exp}}$ is for the explicit scheme [5] and $C^{\text{impl}}$ is for the implicit scheme (7).

| M  | $C_x^{\text{exp}}$ | $C_x^{\text{impl}}$ | $\Delta C_x$, % | $C_y^{\text{exp}}$ | $C_y^{\text{impl}}$ | $\Delta C_y$, % |
|----|------------------|------------------|----------------|------------------|------------------|----------------|
| 20 | 0.1442           | 0.1422           | 1.39           | 2.6700           | 2.6345           | 1.33           |
| 22 | 0.1421           | 0.1399           | 1.55           | 2.6779           | 2.6510           | 1.00           |
| 24 | 0.1478           | 0.1387           | 6.16           | 2.6981           | 2.6734           | 0.92           |
| 26 | 0.1381           | 0.1361           | 1.45           | 2.7081           | 2.6919           | 0.60           |
| 28 | 0.1365           | 0.1337           | 2.05           | 2.7277           | 2.7003           | 1.00           |

3.2. LU-SGS preconditioning on multiple GPUs

The whole method is divided in two stages. Internal GPU computing and external computing associated with boundary elements between grid-separated slices. For this purpose for each mesh slice in each GPU we form $L$, $D$, $U$ matrices in block compressed row storage (CRS) sparse format that can be used in sparse linear algebra (cuSPARSE) library from compute unified device architecture (CUDA) toolkit on NVIDIA GPUs. Then for each GPU we form two sets of elements—internal elements (that include physical boundary conditions) and boundary elements that are connected to the neighbor mesh slices. Then the two CUDA streams are executed. The set of internal elements is solved using CUDA cuSPARSE library in one of CUDA Streams and the other CUDA Stream is used for MPI intercommunication and factorization of the set of boundary elements. Such approach equals to the single thread sequential LU-SGS procedure with minimum reordering. The reordering is only related to the boundary elements. It is observed that graph coloring approach for unstructured heterogeneous grid would result in prohibitively many iterations to achieve convergence of the linear system.

4. Benchmarking results and computational efficiency

To solve our benchmark problems, we use the following criteria. Linear system iterations stop when the following condition for the norm of the weighted residual is satisfied: $||r_k/a_k||_2 \leq 5 \times 10^{-7}$, $k = 1, \ldots, 5$. Here $a$ is the characteristic length of an element multiplied by the characteristic scale of conservative variables for each conservative variable (counted by index $k$). For benchmarks we deliberately use gas dynamic variables that are not included into the vector of conservative variables because the latter is known to be converged when the nonlinear residual is small. This way we can monitor the convergence properties of implicit schemes on additional secondary variables such as temperature and heat fluxes. Besides, these variables have sufficient engineering value for the design of hypersonic vehicles.

4.1. Gas dynamics, heat fluxes comparison

We benchmark explicit versus implicit method accuracy on the sphere-cone test problems. We use the explicit multi GPU method that is described in [5]. First we consider the problem that was used in [5, 6, 8] with the hypersonic flow over a sphere-cone of length 1 m with 0.01 m radius sphere and the cone angle of 8° and with the zero angle of attack. Inflow gas parameters correspond to the height of 10 km with various Mach numbers considered. We used previously generated unstructured mesh with $10.2 \times 10^6$ elements. Minimum element length in normal direction to the wall of the body surface is $1.298 \times 10^{-6}$ m.

Comparison results of the stationary problem solution are presented in figure 1 for Mach number $M = 28$ and in table 1 for explicit and implicit methods in wide range of Mach numbers.
Figure 1. Logarithm of the heat flux absolute error distribution between explicit and implicit schemes, \(M = 28\), stationary solution.

Figure 2. Logarithm of the temperature distribution in kelvins over the sphere-cone body, second configuration, \(20^\circ\) angle of attack, nonstationary solution.

We also compare integral properties calculating drag \(C_x\) and lift \(C_y\) coefficients with data from [5]. We can observe in table 1 that integral properties of the obtained results are in good agreement with relative error below 3%. We can observe that the difference in \(C_x\) coefficient for \(M = 24\) is the largest. This is possibly due to the incomplete convergence of the explicit scheme. Heat flux difference distribution is depicted in figure 1. We obtained about 7% difference in heat fluxes using implicit and explicit methods. Maxima of differences are on the pinnacles. We observed that for stationary problem the explicit method converges “from above” while the implicit method converges “from below”. The true heat flux value will be “in between” only for the true stationary solution.

Now we consider the second problem setup that was used in [6] with the hypersonic flow over a sphere-cone of length 1 m with 0.1 m radius sphere and the same base angle. We use the same inflow parameters with Mach number 25 and attack angle of 0° and 20°.

Results of the temperature distribution in the solid body solved by the implicit nonstationary method (5) are presented in figure 2. We observe the formation of the high temperature tail after the body with the two main recirculation vortexes. Qualitative results pf the heat flux distribution along the sphere-cone generatrix are presented in figure 3 for 0° angle of attack and 4 for 20° angle of attack. We can see the same picture as for \(C_x/y\) results where main difference is in pinnacle values. We also observe more smoothed data for the implicit method. We can conclude that the suggested implicit method gives physically adequate results.

4.2. Convergence and computational efficiency
We divide computing efficiency into two parts. First, we compare the stationary solution using scheme (7) for the first problem setup with \(8.5 \times 10^5\) grid elements. The problem is divided between 1–4 GPUs. The total convergence history towards the steady state solution is presented in figure 5. In this setup the Courant number has no physical meaning and is selected to accelerate convergence. Maximum Courant number achieved during this simulation is around 4000.

One can observe that there is no particular degradation of convergence for 1 or 4 GPUs problem used. This indicates that the selected method of LU-SGS reordering has close algebraic properties of the original preconditioner matrix \(\tilde{A}\).
Figure 3. Heat flux along the body surface cone generatrix with zero attack angle for $M = 25$ for the second problem setup, explicit vs implicit schemes.

Figure 4. Heat flux along the body surface cone generatrix with $20 ^\circ$ attack angle for $M = 25$ for the second problem setup, explicit vs implicit schemes.

Second, we compare wall time ratio of the steady state and time dependent gas dynamics problem solution by the explicit method to the implicit method. Results for the steady state are provided in figure 6. We can notice that the fully implicit method demonstrates good acceleration compared to the explicit method for both problems setups and for both targets—stationary and nonstationary. The time step $\tau$ is selected as maximum possible for the linear
Figure 5. Convergence history of the first problem with $8.5 \times 10^5$ grid elements divided between 1–4 GPUs reaching stationary solution. B(5) stands for BiCGStab(5) linear solver method. Residual C-norm stands for $\|F(U^n)\|_{L_\infty}$ on the $n$-th iteration.

Figure 6. Ratio of wall time values needed to solve the first problem to the steady state and the second problem for physical time of 1 s by the explicit method and the best implicit one with BiCGStab(5).

Figure 7. Implicit method GPU scaling depending on the problem for $8.5 \times 10^5$ grid elements.

solver to converge. Usually we obtain such time step, that Courant number is up to 3000–5000. Scaling between GPUs is shown in figure 7. This scaling is not perfect with about 30% loss of efficiency. However there results are presented for the relatively small mesh of $8.5 \times 10^5$ grid elements that fits to one GPU. Increase of each GPU load will improve this result, since the ratio of boundary communication elements to the internal elements on each GPU decreases.
5. Conclusion
In the paper, we considered the updated version of the implicit method that is used on multiple GPUs for the hypersonic gas dynamics solution method. We used LU-SGS preconditioner approach that can be efficiently used in multiple GPU setup as preconditioning for the Krylov solvers. We demonstrated the validity of our approach on the solution of high speed hypersonic flow over the sphere-cone setup that we used in our previous work. It turned out that the suggested approach does not greatly decrease preconditioning properties of the factored matrix for multiple GPUs. The method demonstrated successful application to the physically challenging problems. This implicit method is added to the complex solver that has being developed for the solution of coupled heat and mass transfer problems on hypersonic regimes.

Acknowledgments
Computations were performed on the high performance computing mini-cluster owned by one of the authors (N M E).

References
[1] Jameson A and Turkel E 1981 Math. Comput. 37 385–97
[2] Sun Y, Wang Z J, Liu Y and Chen C L 2007 AIAA paper no. 007-313 313
[3] Sharov D, Luo H, Baum J and Loehner R 2000 AIAA paper no. 2000-927 927
[4] Borisov V E, Davydov A A, Kudryashov I Yu, Lutsky A E and Men’shov I S 2015 Math. Model. Comput. Simul. 7 222–32
[5] Bocharov A N, Bityurin V A, Golovin N N, Evstigneev N M, Fortov V E, Petrovskiy V P, Ryabkov O I, Solomonov Yu S, Shustov A A and Teplyakov I O 2016 J. Phys.: Conf. Ser. 774 012157
[6] Bocharov A N, Evstigneev N M and Ryabkov O I 2015 J. Phys.: Conf. Ser. 653 012119
[7] Jameson A and Yoon S 1987 AIAA J. 25 929–35
[8] Bocharov A N, Balakirev B A, Bityurin V A, Gryaznov V K, Golovin N N, Iosilevskiy I L, Evstigneev N M, Medin S A, Naumov N D, Petrovskiy V P, Ryabkov O I, Solomonov Yu S, Tatarinov A V, Teplyakov I O, Tikhonov A A and Fortov V E 2015 J. Phys.: Conf. Ser. 653 012070