DOI: 10.14529/jsfi250105

Application-Specific Parallel Linear Solver for Nonlinear Harmonics Method with Implicit Time Integration

Alexey P. Duben[1], *Andrey V. Gorobets*[1]

© The Authors 2025. This paper is published with open access at SuperFri.org

The present paper covers specific parallel implementation details of the nonlinear harmonics (NLH) method within an implicit time integration framework. The NLH method plays an important role in industrial turbomachinery applications as it accounts for unsteady effects in modelling of compressors and turbines on a base of low-cost stationary approaches: the flow is modelled using the Reynolds-Averaged Navier–Stokes approach, the mixing plane method is used for the rotor-stator interface, and only one periodic sector of a blade passage per row is considered. The main focus is on the adaptation of the linear solver used in the Newtonian process of the implicit scheme. The goal of this work is to significantly reduce memory consumption and improve performance. This goal is achieved by using a specialized block sparse matrix storage format, adapted linear solver preconditioners with approximate inverse diagonal blocks, and a combination of single- and double-precision real number formats.

Keywords: turbomachinery, rotor-stator interaction, non-linear harmonics method, unstructured mesh, parallel CFD, supercomputer.

Introduction

The nonlinear harmonics (NLH) method, first proposed in [1] and later applied to realistic turbomachines [2, 3], accounts for unsteady effects in simulations of compressors and turbines using low-cost stationary approaches. The flow is modeled on a base of the Reynolds-Averaged Navier–Stokes (RANS) approach. Only one periodic sector of blade passage per row is considered, which is crucial due to the typically large number of blades in each row, from tens to hundreds. In particular, the NLH is incorporated [4, 5] into the commercial solver Cadence Fidelity Fine Turbo, well known as one of the most efficient simulation tools for turbomachinery.

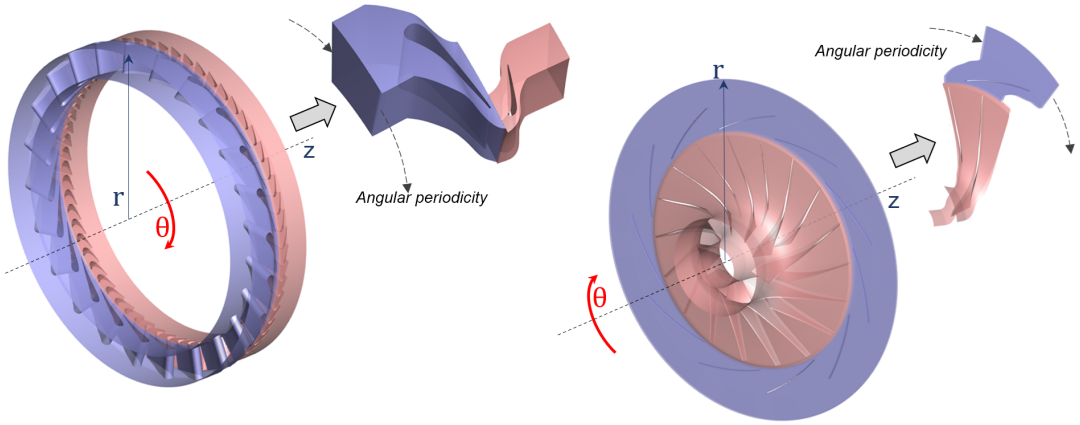


Figure 1. Angular periodicity

The mixing plane (MP) method [6, 7] is used for rotor-stator interfaces, which assumes uniformity of the flow in the circumferential direction at the interface. The NLH method allows capturing the unsteady interaction between adjacent rows by transmitting the perturbations related to the blade passing frequency through the mixing-plane interface.

[1]Keldysh Institute of Applied Mathematics, RAS, Moscow, Russian Federation

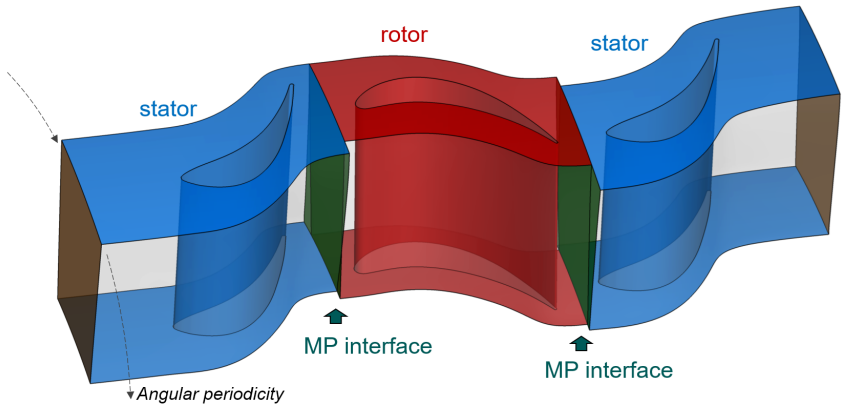


Figure 2. MP rotor-stator interfaces

The NLH technology was successfully implemented [8, 9] within a higher-accuracy numerical algorithm for unstructured meshes in the NOISEtte [10] code. The code has multilevel heterogeneous parallelization [11] by means of MPI, OpenMP and OpenCL parallel standards.

The NLH method needs to solve for each harmonic a supplementary system of transport equations for complex harmonic amplitudes. Thus, each harmonic adds twice as many variables per cell (real and imaginary parts) as in the baseline stationary RANS simulation. What is worse, since the time integration is implicit, the block size of the Jacobian matrix increases from 5 to 10, which increases the storage size and computation cost of the solution by about 4 times. Therefore, the present study is focused on reduction of this undesirable memory consumption.

The rest of the paper is organized as follows. Section 1 briefly describes the NLH technology. Section 2 is devoted to the adaptation of the linear solver by means of a customized sparse block matrix format. The results of memory consumption reduction and performance improvement are presented in Section 3. Finally, the conclusions are summarized.

1. Non-Linear Harmonics Method

1.1. Mathematical Model

The NLH is based on solving the Reynolds-Averaged Navier–Stokes (RANS) equations for simulating a turbulent compressible viscous flow:

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathcal{F}^{\mathrm{C}}(\mathbf{Q}) - \nabla \cdot \mathcal{F}^{\mathrm{D}}(\mathbf{Q}) = 0, \tag{1}$$

where $\mathbf{Q} = (\rho,\ \mathbf{m},\ E)^{\intercal}$ is the vector of averaged conservative variables, ρ is the density, $\mathbf{m} = \rho\mathbf{u}$, where $\mathbf{u} = \{u, v, w\}$ is the velocity vector, $E = \rho(e + \mathbf{u}^2/2)$ is the total energy, e is the specific internal energy. The convective and diffusive fluxes read:

$$\mathcal{F}^{\mathrm{C}}(\mathbf{Q}) = \begin{pmatrix} \mathbf{m} \\ \mathbf{u} \otimes \mathbf{m} + p\mathbf{I} \\ (E + p)\mathbf{u} \end{pmatrix}, \quad \mathcal{F}^{\mathrm{D}}(\mathbf{Q}) = \begin{pmatrix} 0 \\ \boldsymbol{\sigma} \\ \boldsymbol{\sigma} \cdot \mathbf{u} - \mathbf{q} \end{pmatrix}, \tag{2}$$

where $\mathbf{I}$ is the unit tensor, $\boldsymbol{\sigma}(\mathbf{u}) = \{\sigma_{jk}\}(\mathbf{u}) = \mu_{\mathrm{eff}}(\nabla_j u_k + \nabla_k u_j - \delta_{jk}\mathrm{div}\mathbf{u})$ is the stress tensor, δ_{jk} is the Kronecker symbol, $\mathbf{q} = -\frac{\gamma\mu}{\mathrm{Pr}}\nabla e$ is the heat flux vector. $\gamma = c_P/c_V$ is the specific heat ratio, c_P and c_V are the specific heat capacities at constant pressure and constant volume, respectively. $\mathrm{Pr} = c_P\mu/\kappa$ is the Prandtl number, κ is the thermal conductivity coefficient. The

perfect gas equation of state is assumed: $p = (\gamma - 1)\rho e$. $\mu_{\text{eff}} = \mu + \mu_t$, μ is the dynamic viscosity, $\mu_t = \rho\nu_t$, ν_t is the turbulent viscosity defined by a particular turbulence model.

The conservative variables $\mathbf{Q}$ within the NLH method are split into the time-averaged part $\overline{\mathbf{Q}}(x)$ and periodic pulsations $\mathbf{Q}'(t,x)$:

$$\mathbf{Q}(t,x) = \overline{\mathbf{Q}}(x) + \mathbf{Q}'(t,x). \tag{3}$$

The pulsation part $\mathbf{Q}'(t,x)$ is represented as a sum of complex harmonic components:

$$\mathbf{Q}'(t,x) = \frac{1}{2}\sum_{k=1}^{\infty}\left[\widetilde{\mathbf{Q}}_k e^{ik\omega_0 t} + \widetilde{\mathbf{Q}}_{-k} e^{-ik\omega_0 t}\right], \tag{4}$$

where i is the imaginary unit, $\widetilde{\mathbf{Q}}_k = \widetilde{\mathbf{Q}}_{a,k} + i\widetilde{\mathbf{Q}}_{b,k}$, $\widetilde{\mathbf{Q}}_{a,k} = \Re\left(\widetilde{\mathbf{Q}}_k\right)$ and $\widetilde{\mathbf{Q}}_{b,k} = \Im\left(\widetilde{\mathbf{Q}}_k\right)$ are the real and imaginary parts of the amplitude for k-th harmonic, respectively, $\widetilde{\mathbf{Q}}_{-k} = \widetilde{\mathbf{Q}}_{a,k} - i\widetilde{\mathbf{Q}}_{b,k}$ is the complex-conjugate for $\widetilde{\mathbf{Q}}_k$. The number of harmonics N_h defines how accurate the approximation of $\mathbf{Q}'(t,x)$ in (4) is, the more harmonics, the more accurate it is, but at a higher computational cost. The harmonic indices k are defined by the blade passing frequencies (BPF) Ω_{BPF} of adjacent rows: $k = n \cdot k_{\text{BPF}}$, $n = 1, 2, ..., N_h$. k_{BPF}, in turn, is defined as $k_{\text{BPF}} = \Omega_{\text{BPF}}/\Omega_{\text{rot}}$, where Ω_{rot} is the rotation frequency ($k_{\text{BPF}} \in \mathbb{N}$ by definition). Thus, every domain (either rotor or stator) has two sets of harmonic indices k, which are proportional to k_{BPF} from adjacent domains.

According to (3), the RANS system (1) is decomposed into the system for averaged variables and the set of systems for amplitudes of each harmonic. The system for averaged variables $\overline{\mathbf{Q}}$ reads:

$$\frac{\partial \overline{\mathbf{Q}}}{\partial t} + \nabla \cdot \overline{\mathcal{F}}^{C}\left(\overline{\mathbf{Q}}, \mathbf{Q}'\right) - \nabla \cdot \overline{\mathcal{F}}^{\text{D}}(\overline{\mathbf{Q}}, \mathbf{Q}') = 0, \tag{5}$$

$\overline{\mathcal{F}}^{C}\left(\overline{\mathbf{Q}}, \mathbf{Q}'\right) = \mathcal{F}^{\text{C}}(\overline{\mathbf{Q}}) + \mathcal{F}^{\text{C}}_{\text{NLH}}(\mathbf{Q}')$, $\overline{\mathcal{F}}^{\text{D}}(\overline{\mathbf{Q}}, \mathbf{Q}') = \mathcal{F}^{\text{D}}(\overline{\mathbf{Q}}) + \mathcal{F}^{\text{D}}_{\text{NLH}}(\mathbf{Q}')$. Nonlinear deterministic stress contributions of harmonic amplitudes are of the form $\mathcal{F}^{\text{C}}_{\text{NLH}}(\mathbf{Q}') = (0, \overline{\mathbf{m}' \otimes \mathbf{u}'}, \overline{(\mathbf{E}+\mathbf{p})'\mathbf{u}'})^{\intercal}$ and $\mathcal{F}^{\text{D}}_{\text{NLH}}(\mathbf{Q}') = (0, 0, 0, 0, \overline{\boldsymbol{\sigma}' \cdot \mathbf{u}'})^{\intercal}$, where $\boldsymbol{\sigma}' = \boldsymbol{\sigma}(\mathbf{u}')$. The system (5) is distinguished from the system (1) by the presence of the nonlinear terms $\mathcal{F}^{\text{C}}_{\text{NLH}}$ and $\mathcal{F}^{\text{D}}_{\text{NLH}}$, which are defined by $\mathbf{Q}'$.

For brevity, the index k of harmonic amplitude for ω_k frequency will be omitted from now on when referring to harmonic amplitude vectors $\widetilde{\mathbf{Q}}_k$. Thus, the system for a harmonic amplitude vector $\widetilde{\mathbf{Q}}$ is

$$\frac{\partial \widetilde{\mathbf{Q}}}{\partial t} + \nabla \cdot \widetilde{\mathcal{F}}^{C}\left(\widetilde{\mathbf{Q}}, \overline{\mathbf{Q}}\right) + i\omega\widetilde{\mathbf{Q}} - \nabla \cdot \widetilde{\mathcal{F}}^{\text{D}}(\widetilde{\mathbf{Q}}, \overline{\mathbf{Q}}) = 0, \tag{6}$$

$$\widetilde{\mathcal{F}}^{\text{C}} = \begin{pmatrix} \widetilde{\mathbf{m}} \\ \widetilde{\mathbf{u}} \otimes \overline{\mathbf{m}} + \overline{\mathbf{u}} \otimes \widetilde{\mathbf{m}} + \widetilde{p}\mathbf{I} \\ \left(\widetilde{E} + \widetilde{p}\right)\overline{\mathbf{u}} + (\overline{E} + \overline{p})\widetilde{\mathbf{u}} \end{pmatrix}, \quad \widetilde{\mathcal{F}}^{\text{D}} = \begin{pmatrix} 0 \\ \widetilde{\boldsymbol{\sigma}} \\ \widetilde{\boldsymbol{\sigma}} \cdot \overline{\mathbf{u}} + \overline{\boldsymbol{\sigma}} \cdot \widetilde{\mathbf{u}} - \widetilde{\mathbf{q}} \end{pmatrix}, \tag{7}$$

where $\widetilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}(\widetilde{\mathbf{u}})$, $\widetilde{\mathbf{q}} = -\frac{\gamma\mu}{\text{Pr}}\nabla\widetilde{e}$. $\overline{\mu}_{\text{eff}} = \overline{\mu} + \overline{\mu}_t$, $\overline{\mu}_t$ is defined by a turbulence model closing the equations for averaged variables.

Linear systems of equations for harmonic amplitudes (6)–(7) are independent of each other due to orthogonality of harmonics. The equations for averaged variables and harmonic amplitudes are closed by several relations (see [8] for more details), including the following ones: $\overline{f\,g} = \overline{f}\,\overline{g} + \overline{f'g'}$; $(fg)' = f'\,\overline{g} + \overline{f}\,g'$; $\overline{f'g'} = 0.5\sum_{k=1}^{N_h}\left[\Re(\widetilde{f}_k)\Re(\widetilde{g}_k) + \Im(\widetilde{f}_k)\Im(\widetilde{g}_k)\right]$.

1.2. Discretization

The convective terms for harmonic amplitudes $\nabla \cdot \widetilde{\mathcal{F}}^{\mathrm{C}}$ in (6) are discretized in space on an unstructured mesh using the edge-based reconstruction (EBR) [12] (for smooth solutions) and shock-capturing EBR-TVD or EBR-WENO [13] schemes with the scheme [8] based on the Roe [14] scheme for solving the Riemann problem. According to it, the numerical flux $\mathbf{F}_{ij}$ at the surface associated with the edge ij, which connects nodes i and j, is defined as

$$\mathbf{F}_{ij} = \frac{1}{2}\left[\mathcal{F}(\widetilde{\mathbf{Q}}_{ij}, \overline{\mathbf{Q}}_i) + \mathcal{F}(\widetilde{\mathbf{Q}}_{ji}, \overline{\mathbf{Q}}_j)\right] \cdot \mathbf{n}_{ij} - \frac{1}{2}\mathbf{S}_{ij}|\mathbf{\Lambda}_{ij}|\mathbf{S}_{ij}^{-1}\left[\widetilde{\mathbf{Q}}_{ji} - \widetilde{\mathbf{Q}}_{ij}\right], \tag{8}$$

where $\widetilde{\mathbf{Q}}_{ij}$ and $\widetilde{\mathbf{Q}}_{ji}$ are the pre-decay values of conservative harmonic amplitudes, calculated using a EBR-based scheme, at the nodes i and j, respectively. $\mathbf{n}_{ij} = \mathbf{s}_{ij}/|\mathbf{s}_{ij}|$ is the unit vector aligned with $\mathbf{s}_{ij} = -\mathbf{s}_{ji}$ – the surface associated with edge ij. Considering moving of the control volume with the speed $\mathbf{V} = \mathbf{\Omega}_{\mathrm{rot}} \times \mathbf{r}$ ($\mathbf{\Omega}_{\mathrm{rot}}$ is the rotation vector, $\mathbf{r} = (x, y, z)$ is the radius vector of the node), the flux function $\mathcal{F}(\widetilde{\mathbf{Q}}_{ij}, \overline{\mathbf{Q}}_i)$ at the node i can be written as

$$\mathcal{F}(\widetilde{\mathbf{Q}}_{ij}, \overline{\mathbf{Q}}_i) = \begin{pmatrix} \widetilde{\mathbf{m}}_{ij} \\ \widetilde{\mathbf{u}}_{ij} \otimes \overline{\mathbf{m}}_i + \overline{\mathbf{u}}_i \otimes \widetilde{\mathbf{m}}_{ij} + \widetilde{p}_{ij}\mathbf{I} \\ \left(\widetilde{E}_{ij} + \widetilde{p}_{ij}\right)\overline{\mathbf{u}}_i + (\overline{E}_i + \overline{p}_i)\widetilde{\mathbf{u}}_{ij} \end{pmatrix} - \widetilde{\mathbf{Q}}_{ij} \cdot \mathbf{V}. \tag{9}$$

The $\mathcal{F}(\widetilde{\mathbf{Q}}_{ji}, \overline{\mathbf{Q}}_j)$ has the form similar to (9). $\mathbf{\Lambda}_{ij}$ is the diagonal matrix of the eigenvalues of $A_{ij} = d(\mathcal{F} \cdot \mathbf{n}_{ij})/d\widetilde{\mathbf{Q}}$ [14]. $\mathbf{S}_{ij}$ is the matrix of the corresponding eigenvectors. We should emphasize that both $\mathbf{\Lambda}_{ij}$ and $\mathbf{S}_{ij}$ depend only on the averaged variables $\overline{\mathbf{Q}}_i$ and $\overline{\mathbf{Q}}_j$. For the viscous terms $\nabla \cdot \widetilde{\mathcal{F}}^{\mathrm{D}}$ in (6), the method of averaged element splitting (AES) [15] is used.

Since averaged variables and harmonic amplitudes do not depend on time, a pseudo-time iterative process is used to obtain a stationary solution. To do so, the implicit time integration backward differentiation formula (BDF) with a quasi-Newton linearization is used. As already mentioned above, the system (6) differs from (1) by the presence of $\mathcal{F}^{\mathrm{C}}_{\mathrm{NLH}}$ and $\mathcal{F}^{\mathrm{D}}_{\mathrm{NLH}}$ terms. We consider them as source terms independent of $\overline{\mathbf{Q}}$, so (6) is solved using the same numerical scheme as for usual RANS without considering nonstationary perturbations. Also, it enables to perform pseudo-time advancing for the system of averaged variables (5) and harmonic amplitudes (6) separately. At each time step, first a sub-step for averaged variables is computed with harmonic amplitudes frozen, then a sub-step for harmonic amplitudes is carried out with updated averaged variables.

A semidiscrete approximation of (6) for a harmonic amplitude can be formulated as

$$\frac{d\widetilde{\mathbf{Q}}}{dt} - \widetilde{\mathbf{\Phi}}(\widetilde{\mathbf{Q}}, \overline{\mathbf{Q}}, \omega) = 0,$$

where $\widetilde{\mathbf{\Phi}} = -\nabla \cdot \widetilde{\mathcal{F}}^{C}\left(\widetilde{\mathbf{Q}}, \overline{\mathbf{Q}}\right) - i\omega\widetilde{\mathbf{Q}} + \nabla \cdot \widetilde{\mathcal{F}}^{\mathrm{D}}(\widetilde{\mathbf{Q}}, \overline{\mathbf{Q}})$. The implicit scheme reads

$$\left(\frac{\mathbf{I}}{\tau^n} + \widetilde{J}\right)(\widetilde{\mathbf{Q}}^{n+1} - \widetilde{\mathbf{Q}}^n) = \widetilde{\mathbf{\Phi}}\left(\widetilde{\mathbf{Q}}^n, \overline{\mathbf{Q}}^n, \omega\right), \tag{10}$$

where τ^n is the timestep at the n-th pseudo-time step, $\widetilde{J} \approx d\widetilde{\mathbf{\Phi}}/d\widetilde{\mathbf{Q}}$ – an approximated flux Jacobian. We should emphasize that the Jacobian matrix $\widetilde{J}$ for both convective $\widetilde{\mathcal{F}}^{C}$ and diffusive $\widetilde{\mathcal{F}}^{D}$ fluxes depends only on averaged variables $\overline{\mathbf{Q}}$, while a contribution from the source term $i\omega\widetilde{\mathbf{Q}}$ depends on ω only.

The parallel preconditioned BiCGStab solver [16] is applied for the Jacobi linear system. Gauss–Seidel method-based parallel preconditioners [17] for block sparse matrices are used.

Each system of equations for harmonic amplitudes includes the coupled between each other real and imaginary parts. Thus, the blocks of the sparse block matrices for harmonic amplitudes are 10×10, while the system for averaged variables has blocks 5×5. This means that the NLH method requires 4 times more memory to store the matrix and 4 times more computational cost of the sparse matrix-vector product. Fortunately, only one matrix is stored for all the harmonics, since the harmonics are solved one by one.

The boundary conditions for harmonic amplitudes, inlet, outlet, and solid surfaces, are implemented in the same way as for the averaged variables.

The mixing plane (MP) method [7] is used for the rotor-stator interface. The boundary conditions for harmonic amplitudes at the rotor-stator interface are implemented similarly to [2] basing on the MP functionality. For the sector periodicity on the mixing-plane interface, the NLH method uses generalized periodic conditions with phase shift for the closure of harmonic amplitudes with different number of blades in adjacent rows, as in [18]. Further details on the numerical method can be found in [8].

2. Linear Solver Adaptation

The NOISEtte code has an in-house heterogeneous parallel linear solver that uses the preconditioned BiCGStab [16] iterative method with several preconditioners based on the Gauss–Seidel method [17]. It supports multilevel MPI+OpenMP+OpenCL parallelization for running on numerous CPUs and GPUs. The solver is a multi-system solver for systems with block sparse matrices that share the same portrait. The matrices are stored in a block CSR (Compressed Sparse Row) format, with dense blocks of coefficients placed linearly in memory in a row-by-row order. Obviously, sharing one portrait across multiple matrices saves memory for storing portraits. In addition, the group solution for several systems at once allows to reduce the network latency overhead in a distributed parallel mode. MPI messages in the common iterative process are grouped, including halo update operations in sparse matrix-vector products (SpMV) and reduction exchanges in dot products, which reduces the number of messages by several times. As soon as the solution of one of the systems satisfies the residual criterion, it is excluded from the solution process by setting the corresponding convergence flag. Typically, in a basic stationary RANS simulation, there is one system with 5×5 blocks for the 5 main variables (density, 3 velocity vector components, pressure) and from 1 to 4 additional systems for the turbulent model variables (with block sizes 1×1 or 2×2), depending on a particular RANS model.

In the NOISEtte code, a mixed-precision approach is used, in which the Jacobian matrix is stored in single-precision floating-point format, while the mesh functions and discrete operators coefficients are represented in double-precision format (see [11] for details). The linear solver thus operates mainly in single-precision format: vectors and matrices are in single-precision, while BiCGStab scalar coefficients are in double-precision, as well as some intermediate values, such as sum accumulators in dot products, etc. For stability reasons, inversion of diagonal blocks can also be performed in double precision, the result of which is then converted to single precision.

This baseline solver can be directly used for the NLH method as well, but with a matrix block size of 10×10, which is very wasteful. In order to reduce memory consumption and computational cost, the solver has been upgraded with a customized matrix storage format and dedicated matrix-vector product functions. The preconditioner has also been significantly redesigned for

the new matrix format, including the matrix diagonal blocks inversion function and inverted diagonal blocks storage.

2.1. Customized Matrix Format for Harmonics Matrices

The matrix portrait corresponds to nodal adjacency via mesh edges. In the matrices for harmonics, each 10×10 matrix block corresponds to 5 real and 5 imaginary parts of complex variables in nodes. If the variables in blocks are ordered accordingly, first real parts, then imaginary, then the diagonal and off-diagonal 10×10 blocks have the following structure, respectively:

$$\mathbf{A}_{ii} = \begin{pmatrix} \mathcal{A}_{ii} & -\omega v_i \mathcal{I} \\ \omega v_i \mathcal{I} & \mathcal{A}_{ii} \end{pmatrix}, \quad \mathbf{A}_{ij} = \begin{pmatrix} \mathcal{A}_{ij} & 0 \\ 0 & \mathcal{A}_{ij} \end{pmatrix}, \quad i \neq j, \tag{11}$$

where $\mathcal{A}_{ii}$ are dense 5×5 blocks, ω is the frequency to which these harmonic amplitudes correspond (ω_k, actually), v_i is the volume of i-th cell, and $\mathcal{I}$ is the 5×5 identity matrix.

This specific matrix structure allows to easily reduce memory consumption for matrix storage by about 4 times. Instead of 100 values, 26 values are stored: 25 values of the 5×5 block $\mathcal{A}_{ii}$ and one more value ωv_i.

Furthermore, the matrices for harmonics depend only on averaged variables except diagonal coefficients, which contain contributions defined by the source terms $i\omega\widetilde{\mathbf{Q}}$. Thus, once the Jacobian matrix is filled for the averaged variables, it can be reused for the harmonics with only the diagonal elements of diagonal blocks being updated. To implement this memory-saving strategy, apart from minor updates in the matrix format, all the necessary operations involving products with matrix blocks must be provided for this customized block representation. In case of the BiCGStab solver, only the SpMV operation needs to be updated, and the preconditioner may also require changes.

2.2. Preconditioner with Block-Diagonal Inversion

Situation with the preconditioner based on the block Jacobi method, $\mathbf{x}_i^{m+1} = {\mathbf{A}_{ii}}^{-1}\left(\mathbf{b}_i - \sum_{j\neq i}\mathbf{A}_{ij}\mathbf{x}_j^m\right)$, or the block Gauss–Seidel method, $\mathbf{x}_i^{m+1} = {\mathbf{A}_{ii}}^{-1}\left(\mathbf{b}_i - \sum_{j=1}^{i-1}\mathbf{A}_{ij}\mathbf{x}_j^{m+1} - \sum_{j=i+1}^{n}\mathbf{A}_{ij}\mathbf{x}_j^m\right)$, needs more attention, because inverted diagonal blocks are required to obtain solution on the next iteration, $m+1$, of the inner iterative process. Inverted diagonal blocks do not necessarily follow the original block structure, and storage for full 10×10 blocks is required (and about 8 times more arithmetic operations for the inversion compared to 5×5 blocks)

The matrices for harmonics could be treated as block matrices of 5×5 blocks with a portrait 4 times larger. In this case, only one 5×5 diagonal block, $\mathcal{A}_{ii}$, needs to be inverted and stored for each 10×10 block $\mathbf{A}_{ii}$. This requires less computational effort to invert the diagonal blocks and 4 times less memory to store it. On the other hand, the Gauss-Seidel block method would require a double pass for each 10×10 block row then (since it is considered as two 5×5 block rows), which is inefficient enough to overcome all the profit. So we approximate the inverted 10×10 blocks in such a way that only the 5×5 diagonal blocks are stored, while still dealing with the 10×10 block rows in the Jacobi and Gauss–Seidel methods, respectively: $\mathbf{x}_i^{m+1} = {\mathbf{A}_{ii}}^{-1}\left(\mathbf{b}_i - \sum_{j\neq i}\mathbf{A}_{ij}\mathbf{x}_j^m - \mathbf{\Omega}\mathbf{x}_i^m\right)$,

$\mathbf{x}_i^{m+1} = \tilde{\mathbf{A}}_{ii}^{-1}\left(\mathbf{b}_i - \sum_{j=1}^{i-1}\mathbf{A}_{ij}\mathbf{x}_j^{m+1} - \sum_{j=i+1}^{n}\mathbf{A}_{ij}\mathbf{x}_j^{m} - \mathbf{\Omega}\mathbf{x}_i^{m}\right)$, where

$$\tilde{\mathbf{A}}_{ii}^{-1} = \begin{pmatrix} {\mathcal{A}_{ii}}^{-1} & 0 \\ 0 & {\mathcal{A}_{ii}}^{-1} \end{pmatrix}, \quad \mathbf{\Omega} = \begin{pmatrix} 0 & -\omega v_i \mathcal{I} \\ \omega v_i \mathcal{I} & 0 \end{pmatrix}.$$

Using this approximate inversion, complemented with the minor modification by including the extra term $\mathbf{\Omega}\mathbf{x}_i^m$, allows saving a lot of computational effort and memory on dealing with inverted diagonal blocks. However, it is necessary that such an approximation does not spoil the solver convergence process (otherwise, the solver may need many more iterations, which will overcome all the gains). To verify this, a direct comparison is made with the original variant with full storage of 10×10 inverted blocks, results are demonstrated in the following section.

3. Performance Analysis

3.1. Cases Description

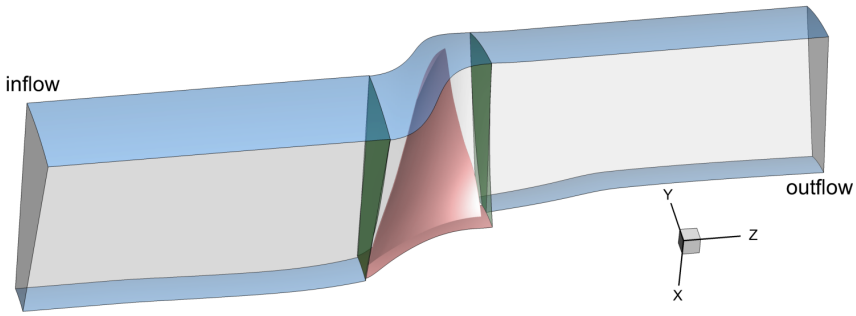
Two model cases are used to demonstrate the performance of the modified implicit solver. Simulations for both test cases were performed using the Menter SST [21] turbulence model. The hexahedral meshes for the test cases are built using the TurboR&D.Mesher software [22].

The first case is an isolated rotor with 22-blade impeller, NASA Rotor-67 [19], which is widely used for validation in turbomachinery applications. Although there is no stator, the computational domain is divided into three parts, as shown in Fig. 3, so that the static subdomains conduct the perturbations generated by the rotating impeller, which are captured by the NLH method. The previously obtained head-capacity characteristics of the Rotor-67 are presented in [9]. Here, the maximum efficiency regime of the rotor is considered for the performance tests. The hexahedral computational mesh contains 3.58 million nodes (single-blade sector with periodic boundary conditions). The same number of periodicity sectors is applied to all three subdomains.

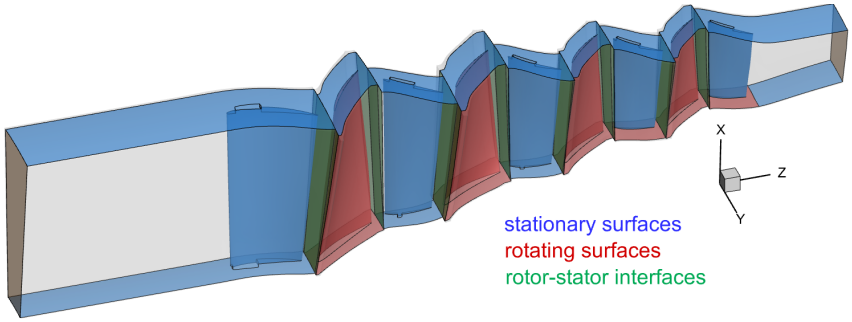
The second case is more computationally intensive, it is part of a model axial compressor of a gas turbine engine with a scaled number of blades (see Fig. 3, and further details can be found in [20]). The configuration considered consists of an inlet guide vane and four stages of rotor and stator, which in total represents 8 rotor-stator interfaces. The mesh for this configuration contains 11.9 million nodes. As for the Rotor-67, head-capacity characteristics can be found in [9]. The highest efficiency regime is considered here for performance tests.

3.2. Demonstration of Performance Improvement

Reduced memory consumption and increased performance are demonstrated in comparison with the previous NLH method implementation [9] on the Rotor-67 model test case. The hexahedral mesh contains about 3.6 million nodes. The total memory consumption depending on the number of harmonics is shown in Fig. 4. It can be seen that the additional memory consumption for storing the matrices for harmonics has become negligible. The performance ratio depends on the number of harmonics, since it is the solver for harmonic systems that has been accelerated. For instance, in the typical case of using three harmonics, a speedup of 1.6 times has been achieved.

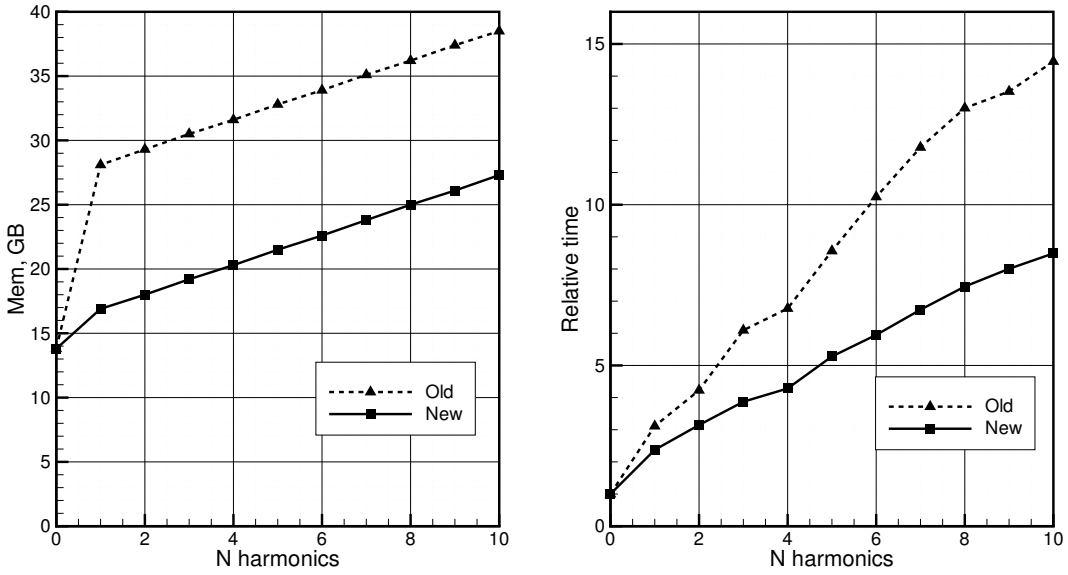


(a) Rotor-67 model rotor



(b) Axial multi-stage model compressor

Figure 3. Computational domains of the cases considered



(a) Memory consumption

(b) Computation time

Figure 4. Memory consumption and computation time relative to the no-harmonics simulation for the Rotor-67 test case as a function of the number of harmonics, comparing the previous implementation [9] (Old) and the improved one (New)

3.3. Performance of Modified Preconditioners

Now let us separately consider the acceleration of preconditioners of the linear iterative solver. A comparison of the baseline preconditioners with full storage of 10×10 inverted blocks

and the modified preconditioners, which use reduced 5×5 inverted blocks, is performed on example of the Rotor-67 test case with 3 harmonics.

To maximize the possible negative effect on the BiCGStab solver convergence from the diagonal inversion approximation, a single iteration of the Jacobi method with zero initial guess is used, which corresponds to the block-diagonal preconditioner. The solver tolerance is set several orders of magnitude smaller than what is typically used in practice to more accurately measure the difference in the number of iterations, with the iteration counts averaged over many time steps. The difference appeared to be negligible, within 2%. Then, Gauss–Seidel (GS) and symmetric GS (SGS) preconditioners were tested, and no notable difference in the resulting number of BiCGStab iterations was observed neither.

Finally, the SGS preconditioner with the approximate diagonal blocks inversion was tested in practical conditions. Comparative testing showed that diagonal block inversion is accelerated by 2.8 times, and the total time for solving linear systems for harmonics has become 1.17 times shorter. The preconditioner solution stage, which takes most of the solver time, is now 1.15 times faster. The overall speedup achieved for the simulation due to the preconditioner upgrade is about 8%. Memory consumption for storing the inverted diagonal of harmonic systems is reduced by 4 times, which corresponds to 6.5% reduction of the total memory used.

3.4. Parallel Performance

Since the computation time was reduced and the data exchanges remained the same, the parallel speedup could have notably degraded. To make sure this did not happen, parallel speedup measurements were performed on a cluster using more than 600 cores. The cluster nodes are equipped with two 16-core Intel Xeon CPUs (8 DDR4-3200 memory channels each). The tests for the Rotor-67 and model multi-stage axial compressor are performed using 3 harmonics. The speedup plots obtained are shown in Fig. 5. No notable degradation of parallel efficiency has been observed compared to the previous version [9], which is about 1.6 times slower. For instance, on 16 nodes, the parallel efficiency (acceleration at 16 nodes relative to one node, divided by the number of nodes and multiplied by 100%) of the multi-stage compressor simulation was 82% and now is 83%, the difference seems to be just within the measurement accuracy. Parallel efficiency of 77% has been obtained on the Rotor-67 case when having about 6 thousand nodes per core.

Single-node parallel performance is demonstrated on the Rotor-67 case using 3 harmonics. The speedup relative to sequential execution is presented in Tab. 1 for various parallel execution modes with different ratios of MPI processes and OpenMP threads. On a single 16-core CPU, the OpenMP speedup is about 10 times. Using simultaneous multithreading with 2 threads per core gives about 6% speedup. This can be explained by mitigating the effect of memory latency on cache misses, since when one thread gets stuck on a memory transaction, another can proceed. It should also be noted that the SGS predonditioner is parallelized by blocks: the matrix is sliced among threads, and each thread applies the preconditioner to its diagonal block, at the interface the unknowns are taken from the previous iteration, as in the Jacobi method. Therefore, the solver convergence may degrade as the number of threads increases. To evaluate this effect, the average number of solver iterations was measured when setting the solver tolerance several orders of magnitude lower than is typically used in practice. The solver with 32 threads needs on average about 12% more iterations than with 1 thread (however, the impact on the overall convergence of the solution to the nonlinear problem is less significant).

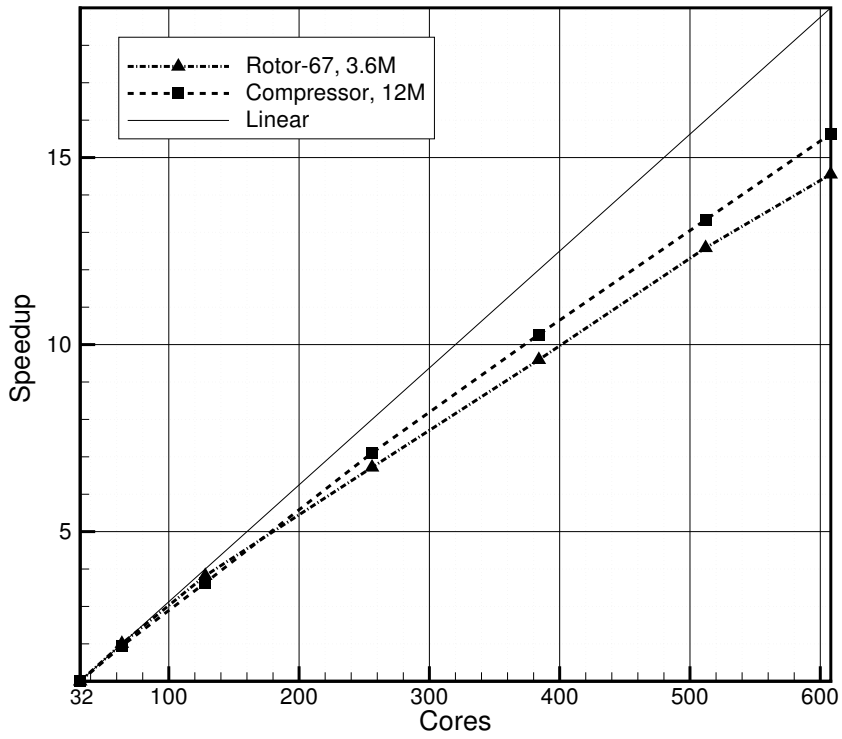


Figure 5. Parallel speedup on a cluster system for the Rotor-67 and model multi-stage axial compressor using 3 harmonics

On two CPUs running one MPI process per CPU (to avoid NUMA factor), the speedup is about 22 times. Considering that the numerical algorithm is rather memory-bound (low arithmetic intensity per unit of memory traffic, especially for the sparse linear solver) and the number of memory channels is 16, this acceleration can be viewed as high enough.

Table 1. Single-node performance on the Rotor-67 case using 3 harmonics

Cores	CPUs	MPI	OpenMP	Time, s	Speedup
1	1	1	1	53.3	1.0
16	1	1	16	5.37	9.9
16	1	1	32	5.08	10.5
32	2	1	64	3.36	15.9
32	2	2	32	2.42	22
32	2	32	1	2.32	23
32	2	32	2	2.23	23.9

Conclusion

The NLH method reproduces unsteady effects in turbomachinery applications simulated using low-cost RANS approaches. It allows for the modeling of only one periodic sector of a single blade passage per row in compressors and turbines and the transmission of non-stationary perturbations through a mixing-plane interface. An important problem with this method is the memory consumption, which grows with the number of harmonics. In case of an implicit

scheme, the Jacobian sparse block matrix for complex variables has a block size twice as large and, accordingly, needs four times more memory. Reducing memory consumption is especially critical for using the NLH method on GPUs, which have very limited memory size. In the present work, this reduction has been achieved by using a specialized block sparse matrix storage format and approximate inverse diagonal blocks in Gauss–Seidel based preconditioners. Compared to our previous NLH implementation [9], the proposed modifications have reduced the memory footprint of the Jacobian matrix for harmonics by a factor of 4, resulting in an overall reduction in memory consumption by about 1.5 times, as well as a speedup of about 1.5×. At the same time, parallel performance in shared and distributed memory was maintained at a fairly high level, showing about 80% parallel efficiency when running on hundreds of CPU cores with a payload per core of about 6 thousand mesh nodes.

The achieved reduction in memory consumption allows us to begin using GPUs in simulations with the NLH method. Our further work will be aimed at porting the NLH computational algorithm, including new solver kernels, to GPUs. This should not be very difficult, since the computational algorithm of the NLH method implemented in the NOISEtte framework is fully compatible with the stream processing paradigm and can rely on the existing GPU computing infrastructure of the code. It is also planned to incorporate the NLH method for use with the full approximation multigrid method [23], which is expected to provide a significant convergence speedup.

Acknowledgements

Our simulations have been carried out using the equipment of the Shared Resource Center of KIAM RAS (http://ckp.kiam.ru) and the shared research facilities of HPC computing resources at Lomonosov Moscow State University [24]. The authors thankfully acknowledge these institutions.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. He, L., Ning, W.: Efficient approach for analysis of unsteady viscous flows in turbomachines. AIAA J. 36(11), 2005–2012 (1998). https://doi.org/10.2514/2.328

2. Chen, T., Vasanthakumar, P., He, L.: Analysis of unsteady blade row interaction using nonlinear harmonic approach. J. Propuls. Power 17(3), 651–658 (2001). https://doi.org/10.2514/2.5792

3. He, L., Chen, T., Wells, R.G., *et al.*: Analysis of Rotor-Rotor and Stator-Stator Interferences in Multi-Stage Turbomachines. J. Turbomach. Trans. ASME 124(4), 564–571 (2002). https://doi.org/10.1115/1.1508382

4. Vilmin, S., Lorrain, E., Hirsch, C., *et al.*: Unsteady flow modeling across the rotor/stator interface using the non-linear harmonic method. In: Proceedings of the ASME Turbo Expo 2006: Power for Land, Sea, and Air. Volume 6: Turbomachinery, Parts A and B. pp. 1227–1237. ASME (2006). https://doi.org/10.1115/GT2006-90210

5. Vilmin, S., Lorrain, E., Tartinville, B., *et al.*: The Nonlinear Harmonic Method: From single stage to multi-row effects. Int. J. Comput. Fluid Dyn. 27(2), 88–99 (2013). https://doi.org/10.1080/10618562.2012.752074

6. Burgos, M.A., Contreras, J., Corral, R.: Efficient Edge-Based Rotor/Stator Interaction Method. AIAA J. 49(1), 19–31 (2011). https://doi.org/10.2514/1.44512

7. Duben, A., Gorobets, A., Soukov, S., *et al.*: Supercomputer Simulations of Turbomachinery Problems with Higher Accuracy on Unstructured Meshes. In: Voevodin, V., Sobolev, S., Yakobovskiy, M., Shagaliev, R. (eds.) RuSCDays 2022. LNCS, vol. 13708, pp. 356–367. Springer (2022). https://doi.org/10.1007/978-3-031-22941-1

8. Duben, A.P., Zagitov, R.A., Shuvaev, N.V.: Nonlinear harmonics method for supercomputer simulations of fluid dynamics in turbomachines with higher accuracy on unstructured meshes. Lobachevskii J. Math. 45(7), 3007–3016 (2024). https://doi.org/10.1134/S1995080224603916

9. Duben, A.P., Zagitov, R.A., Shuvaev, N.V., Marakueva, O.V.: Towards an Adaptation of the Nonlinear Harmonics Method Realized in an Unstructured Flow Solver for Simulation of Turbomachinery Problems on Supercomputers. In: Voevodin, V., Antonov, A., Nikitenko, D. (eds.) Supercomputing. RuSCDays 2024. LNCS, vol. 15406, pp. 253–266. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-78459-0_19

10. Abalakin, I.V., Bakhvalov, P.A., Bobkov, V.G., *et al.*: NOISEtte CFD&CAA Supercomputer Code for Research and Applications. Supercomputing Frontiers and Innovations 11(2), 78–101 (2024). https://doi.org/10.14529/jsfi240206

11. Gorobets, A., Bakhvalov, P.: Heterogeneous CPU+GPU parallelization for high-accuracy scale-resolving simulations of compressible turbulent flows on hybrid supercomputers. Computer Physics Communications 271, 108231 (2022). https://doi.org/10.1016/j.cpc.2021.108231

12. Bakhvalov, P.A., Abalakin, I.V., Kozubskaya, T.K.: Edge-based reconstruction schemes for unstructured tetrahedral meshes. Int. J. Numer. Methods Fluids 81(6), 331–356 (2016). https://doi.org/10.1002/fld.4187

13. Bakhvalov, P.A., Kozubskaya, T.K.: EBR-WENO scheme for solving gas dynamics problems with discontinuities on unstructured meshes. Comput. Fluids 157, 312–324 (2017). https://doi.org/10.1016/j.compfluid.2017.09.004

14. Roe, L.: Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. J. Comput. Phys. 43, 357–372 (1981).

15. Bakhvalov, P., Surnachev, M.: Method of averaged element splittings for diffusion terms discretization in vertex-centered framework. J. Comput. Phys. 450, 110819 (2022). https://doi.org/10.1016/j.jcp.2021.110819

16. van der Vorst, H.: Bi-CGSTAB: A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. 13, 631–644 (1992). https://doi.org/10.1137/0913035

17. Magomedov, A.R., Gorobets, A.V.: Heterogeneous Implementation of Preconditioners Based on GaussSeidel Method for Sparse Block Matrices. Computational Mathematics and Modeling 33, 438–442 (2022). https://doi.org/10.1007/s10598-023-09585-2

18. Gerolymos, G.A., Michon, G.J., Neubauer, J.: Analysis and Application of Chorochronic Periodicity in Turbomachinery Rotor/Stator Interaction Computations. J. Propuls. Power 18(6), 1139–1152 (2002). https://doi.org/doi:10.2514/2.6065

19. Strazisar, A.J., Wood, J.R., Hathaway, M.D., Suder, K.L.: Laser Anemometer Measurements in a Transonic Axial-Flow Fan Rotor. NASA TP-2879, 1989.

20. Voroshnin, D.V., Marakueva, O.V., Muraveiko, A.S.: Modeling Unsteady Phenomena in an Axial Compressor. Math. Models Comput. Simul. 12, 413–421 (2020). https://doi.org/10.1134/S2070048220030187

21. Menter, F.R., Kuntz, M., Langtry, R.: Ten Years of Industrial Experience with the SST Turbulence Model. In: Proceedings of the 4th International Symposium on Turbulence, Heat and Mass Transfer, Begell House Inc., West Redding, pp. 625–632 (2003). https://doi.org/10.2514/3.12149

22. Zagitov, R.A., Salnikov, S.D., Shuvaev, N.V.: Automatic Block-Structured Grid Generation in Turbo Machine Blade Passages by TurboR&D.Mesher Software. Math. Models Comput. Simul. 16, 112–122 (2024). https://doi.org/10.1134/S2070048224010125

23. Gorobets, A.V., Soukov, S.A., Magomedov A.R.: Heterogeneous Parallel Implementation of a Multigrid Method with Full Approximation in the Noisette Code. Math. Models Comput. Simul. 16, 609–619 (2024). https://doi.org/10.1134/S2070048224700261

24. Voevodin, V., Antonov, A., Nikitenko, D., *et al.*: Supercomputer Lomonosov-2: Large scale, deep monitoring and fine analytics for the user community. Supercomput. Front. Innov. 6(2), 4–11 (2019). https://doi.org/10.14529/jsfi190201