A Massively Parallel Algorithm for the Three-Dimensional Navier-Stokes-Boussinesq Simulations of the Atmospheric Phenomena

Maciej Paszyński[1(✉)], Leszek Siwik[1], Krzysztof Podsiadło[1], and Peter Minev[2]

[1] Department of Computer Science, AGH University of Science and Technology, Krakow, Poland
paszynsk@agh.edu.pl

[2] Applied Mathematics Institute, Mathematical and Statistical Sciences, University of Alberta, Edmonton, Canada

Abstract. We present a massively parallel solver using the direction splitting technique and stabilized time-integration schemes for the solution of the three-dimensional non-stationary Navier-Stokes-Boussinesq equations. The model can be used for modeling atmospheric phenomena. The time integration scheme utilized enables for efficient direction splitting algorithm with finite difference solver. We show how to incorporate the terrain geometry into the simulation and how to perform the domain decomposition. The computational cost is linear $\mathcal{O}(N)$ over each sub-domain, and near to $\mathcal{O}(N/c)$ in parallel over 1024 processors, where N is the number of unknowns and c is the number of cores. This is even if we run the parallel simulator over complex terrain geometry. We analyze the parallel scalability experimentally up to 1024 processors over a PROMETHEUS Linux cluster with multi-core processors. The weak scalability of the code shows that increasing the number of subdomains and processors from 4 to 1024, where each processor processes the subdomain of $49 \times 49 \times 99$ internal points ($50 \times 50 \times 100$ box), results in the increase of the total computational time from 120 s to 178 s for a single time step. Thus, we can perform a single time step with over 1,128,000,000 unknowns within 3 min. The number of unknowns results from the fact that we have three components of the velocity vector field, one component of the pressure, and one component of the temperature scalar field over 256,000,000 mesh points. The computation of the one time step takes 3 min on a Linux cluster. The direction splitting solver is not an iterative solver; it solves the system accurately since it is equivalent to Gaussian elimination. Our code is interfaced with the mesh generator reading the NASA database and providing the Earth terrain map. The goal of the project is to provide a reliable tool for parallel, fully three-dimensional computations of the atmospheric phenomena.

Keywords: Massive parallel computations · Alternating direction solver · Navier-Stokes Boussinesq · Finite difference method

V. V. Krzhizhanovskaya et al. (Eds.): ICCS 2020, LNCS 12137, pp. 102–117, 2020.
https://doi.org/10.1007/978-3-030-50371-0_8

1 Introduction

Air pollution is receiving a lot of interest nowadays. It is visible, especially in the Kraków area in Poland (compare Fig. 1), as this is one of the most polluted cities in Europe [1]. People living there are more and more aware of the problem, which causes the raising of various movements that are trying to improve air quality. Air pollution grows because of multiple factors, including traffic, climate, heating in the winter, the city's architecture, etc. The ability to model atmospheric phenomena such as thermal inversion over the complicated terrain is crucial for reliable simulations of air pollution. Thermal inversion occurs when a layer of warm air stays over a layer of cool air, and the warm air holds down the cool air and it prevents pollutants from rising and scattering.

Fig. 1. Pollution with fog and thermal inversion over the same area near Kraków between October 2019 and January 2020 (photos by Maciej Paszyński)

We present a massively parallel solver using the direction splitting technique and stabilized time-integration schemes for the solution of the three-dimensional non-stationary Navier-Stokes-Boussinesq equations.

The Navier-Stokes-Boussinesq system is widely applied for modeling the atmospheric phenomena [2], oceanic flows [3] as well as the geodynamics simulations [4]. The model can be used for modeling atmospheric phenomena, in particular, these resulting in a thermal inversion. It can be used as well for modeling several other important atmospheric phenomena [5,6]. It may even be possible to run the climate simulation of the entire Earth atmosphere using the

approach presented here. The time integration scheme utilized results in a Kronecker product structure of the matrices, and it enables for efficient direction splitting algorithm with finite difference solver [7], since the matrix is a Kronecker product of three three-diagonal matrices, resulting from discretizations along x, y, and z axes. The direction splitting solver is not an iterative solver; it is equivalent to the Gaussian elimination algorithm.

We show how to extend the alternating directions solver into non-regular geometries, including the terrain data, still preserving the linear computational cost of the solver. We follow the idea originally used in [8] for sequential computations of particle flow. In this paper, we focus on parallel computations, and we describe how to compute the Schur complements in parallel with linear cost, and how to aggregate them further and still have a tri-diagonal matrix that can be factorized with a linear computational cost using the Thomas algorithm. We also show how to modify the algorithm to work over the complicated non-regular terrain structure and still preserve the linear computational cost.

Thus, if well parallelized, the parallel factorization cost is near to $\mathcal{O}(N/c)$ in every time step, where N is the number of unknowns and c is the number of cores. We analyze the parallel scalability of the code up to 1024 multi-core processors over a PROMETHEUS Linux cluster [9] from the CYFRONET supercomputing center. Each subdomain is processed with $50 \times 50 \times 100$ finite difference mesh. Our code is interfaced with the mesh generator [10] reading the NASA database [11] and providing the Earth terrain map. The goal of the project is to provide a reliable tool for parallel fully three-dimensional computations of the atmospheric phenomena resulting in the thermal inversion and the pollution propagation.

In this paper, we focus on the description and scalability of the parallel solver algorithm, leaving the model formulation and large massive parallel simulations of different atmospheric phenomena for future work. This is a challenging task itself, requiring to acquire reliable data for the initial state, forcing, and boundary conditions.

2 Navier-Stokes Boussinesq Equations

The equations in the strong form are

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p + \mathrm{Pr}\Delta \mathbf{u} = g\mathrm{PrRa}T + f \text{ in } \Omega \times (0, T_f] \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega \times (0, T_f] \tag{2}$$

$$\mathbf{u} = 0 \text{ in } \partial\Omega \times (0, T_f] \tag{3}$$

$$\frac{\partial T}{\partial t} + (u \cdot \nabla)T + \Delta T = 0 \text{ in } \Omega \times (0, T_f] \tag{4}$$

$$T = 0 \text{ in } \partial\Omega \times (0, T_f] \tag{5}$$

where u is the velocity vector field, p is the pressure, $Pr = 0.7$ is the Prandt number, $g = (0, 0, -1)$ is the gravity force, $Ra = 1000.0$ is the Rayleigh number, T is the temperature scalar field.

We discretize using finite difference method in space and the time integration scheme resulting in a Kronecker product structure of the matrices.

We use the second-order in time unconditionally stable time integration scheme for the temperature equation and for the Navier-Stokes equation, with the predictor-corrector scheme for pressure. For example we can use the Douglass-Gunn scheme [13], performing an uniform partition of the time interval $\bar{I} = [0, T]$ as

$$0 = t_0 < t_1 < \ldots < t_{N-1} < t_N = T,$$

and denoting $\tau := t_{n+1} - t_n$, $\forall n = 0, \ldots, N-1$. In the Douglas-Gunn scheme, we integrate the solution from time step t_n to t_{n+1} in three substeps as follows:

$$\begin{cases} (1 + \frac{\tau}{2}\mathcal{L}_1)u^{n+1/3} = \tau f^{n+1/2} + (1 - \frac{\tau}{2}\mathcal{L}_1 - \tau\mathcal{L}_2 - \tau\mathcal{L}_3)u^n, \\ (1 + \frac{\tau}{2}\mathcal{L}_2)u^{n+2/3} = u^{n+1/3} + \frac{\tau}{2}\mathcal{L}_2 u^n, \\ (1 + \frac{\tau}{2}\mathcal{L}_3)u^{n+1} = u^{n+2/3} + \frac{\tau}{2}\mathcal{L}_3 u^n. \end{cases} \tag{6}$$

For the Navier-Stokes equations, $\mathcal{L}_1 = \partial_{xx}$, $\mathcal{L}_2 = \partial_{yy}$, and $\mathcal{L}_3 = \partial_{zz}$, and the forcing term represents $g\mathrm{PrRa}T^{n+1/2}$ plus the convective flow and the pressure terms $\left(u^{n+1/2} \cdot \nabla u^{n+1/2}\right) + \nabla \tilde{p}^{n+1/2}$ treated explicitly as well. The pressure is computed with the predictor/corrector scheme. Namely, the predictor step

$$\tilde{p}^{n+\frac{1}{2}} = p^{n-\frac{1}{2}} + \phi^{n-\frac{1}{2}}, \tag{7}$$

with $p^{-\frac{1}{2}} = p_0$ and $\phi^{-\frac{1}{2}} = 0$ computes the pressure to be used in the velocity computations, the penalty steps

$$\begin{cases} \psi - \partial_{xx}\psi = -\frac{1}{\tau}\nabla \cdot u^{n+1}, \\ \xi - \partial_{yy}\xi = \psi, \\ \phi^{n+1/2} - \partial_{zz}\phi^{n+1/2} = \xi, \end{cases} \tag{8}$$

and the corrector step updates the pressure field based on the velocity results and the penalty step

$$p^{n+\frac{1}{2}} = p^{n-\frac{1}{2}} + \phi^{n+\frac{1}{2}} - \chi\nabla \cdot \left(\frac{1}{2}(u^{n+1} + u^n)\right). \tag{9}$$

These steps are carefully designed to stabilize the equations as well as to ensure the Kronecker product structure of matrix, resulting in the linear computational cost solver. The mathematical proofs of the stability of the formulations, motivating such the predictor/corrector (penalty) steps, can be found in [7,12] and the references there.

For the temperature equation, $\mathcal{L}_1 = \partial_{xx}$, $\mathcal{L}_2 = \partial_{yy}$, and $\mathcal{L}_3 = \partial_{zz}$, and the forcing term represents the advection term treated explicitly $\left(u^{n+1/2} \cdot \nabla T^{n+1/2}\right)$.

For mathematical details on the problem formulation and its mathematical properties, we refer to [12].

Each equation in our scheme contains only derivatives in one direction, so they are of the following form

$$\begin{aligned}(1+\alpha\partial_{xx})\,u^{n+1/3} &= RHS_x\\ (1+\alpha\partial_{yy})\,u^{n+2/3} &= RHS_y\\ (1+\alpha\partial_{zz})\,u^{n+1} &= RHS_z\end{aligned} \tag{10}$$

or the update of the pressure scalar field. Thus, when employing the finite difference method, we either endup with the Kronecker product matrices with sub-matrices being three-diagonal, or the point-wise updates of the pressure field

$$\begin{aligned}u_{ijk}^{n+1/3}+\alpha\frac{u_{(i-1)jk}^{n+1/3}-2u_{ijk}^{n+1/3}+u_{(i+1)jk}^{n+1/3}}{dt} &= RHS_x\\ u_{ijk}^{n+2/3}+\alpha\frac{u_{i(j-1)k}^{n+2/3}-2u_{ijk}^{n+2/3}+u_{i(j+1)k}^{n+1/3}}{dt} &= RHS_y\\ u_{ijk}^{n+1}+\alpha\frac{u_{ij(k-1)}^{n+1}-2u_{ijk}^{n+1}+u_{ij(k+1)}^{n+1}}{dt} &= RHS_z,\end{aligned} \tag{11}$$

where $\alpha=\tau/2$ or $\alpha=1$, depending on the equation, which is equivalent to

$$\begin{aligned}\alpha u_{(i-1)jk}^{n+1/3}+(dt-2\alpha)u_{ijk}^{n+1/3}+\alpha u_{(i+1)jk}^{n+1/3} &= dt*RHS_x\\ \alpha u_{i(j-1)k}^{n+2/3}+(dt-2\alpha)u_{ijk}^{n+2/3}+\alpha u_{i(j+1)k}^{n+1/3} &= dt*RHS_y\\ \alpha u_{ij(k-1)}^{n+1}+(dt-2\alpha)u_{ijk}^{n+1}+\alpha u_{ij(k+1)}^{n+1} &= dt*RHS_z,\end{aligned} \tag{12}$$

These systems have a Kronecker product structure $\mathcal{M}=\mathcal{A}^x\otimes\mathcal{B}^y\otimes\mathcal{C}^z$ where the sub-matrices are aligned along the three axis of the system of coordinates, one of these sub-matrices is three-diagonal, and the other two sub-matrices are scalled identity matrices. From the parallel matrix computations point of view, discussed in our paper, it is important that in every time step, we have to factorize in parallel the system of linear equations having the Kronecker product structure.

3 Factorization of the System of Equations Possessing the Kronecker Product Structure

The direction splitting algorithm for the Kronecker product matrices implements three steps, which result is equivalent to the Gaussian elimination algorithm [14], since

$$(\mathcal{M})^{-1}=(\mathcal{A}^x\otimes\mathcal{B}^y\otimes\mathcal{C}^z)^{-1}=(\mathcal{A}^x)^{-1}\otimes(\mathcal{B}^y)^{-1}\otimes(\mathcal{C}^z)^{-1} \tag{13}$$

Each of the three systems is three-diagonal,

$$\begin{bmatrix}A_{11}^x & A_{12}^x & \cdots & 0\\ A_{21}^x & A_{22}^x & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & A_{kk}^x\end{bmatrix}\begin{bmatrix}z_{111} & z_{121} & \cdots & z_{1lm}\\ z_{211} & z_{221} & \cdots & z_{2lm}\\ \vdots & \vdots & \ddots & \vdots\\ z_{k11} & z_{k21} & \cdots & z_{klm}\end{bmatrix}=\begin{bmatrix}y_{111} & y_{121} & \cdots & y_{1lm}\\ y_{211} & y_{221} & \cdots & y_{2lm}\\ \vdots & \vdots & \ddots & \vdots\\ y_{k11} & y_{k21} & \cdots & y_{klm}\end{bmatrix} \tag{14}$$

and we can solve it in a linear $\mathcal{O}(N)$ computational cost. First, we solve along x direction, second, we solve along y direction, and third, we solve along z direction.

4 Introduction of the Terrain

To obtain a reliable three-dimensional simulator of the atmospheric phenomena, we interconnect several components. We interface our code with mesh generator that provides an excellent approximation to the topography of the area [10], based on the NASA database [11]. The resulting mesh generated for the Krakow area is presented in Fig. 2.

In our system of linear equations, we have several tri-diagonal systems with multiple right-hand-sides, factorized along x, y and z directions. Each unknown in the system represents one point of the computational mesh. In the first system, the rows are ordered according to the coordinates of points, sorted along x axis. In the second system, the rows are ordered according to the y coordinates of points, and in the third system, according to z coordinates. When simulating the atmospheric phenomena like the thermal inversion over the prescribed terrain with alternating directions solver and finite difference method, we check if a given point is located in the computational domain. The unknowns representing points that are located inside the terrain (outside the atmospheric domain) are removed from the system of equations. This is done by identifying the indexes of the points along x, y, and z axes, in the three systems of coordinates. Then, we modify the systems of equations, so the corresponding three rows in the three systems of equations are reset to 0, the diagonal is set to 1, and the corresponding rows and columns of the three right-hand-sides are set 0.

For example, if we want to remove point (r, s, t) from the system, we perform the following modification in the first system.

The rows in the first system they follow the numbering of points along x axis. The number of columns corresponds to the number of lines along x axis perpendicular to OYZ plane. Each column of the right-hand side correspond to yz coordinates of a point over OYZ plane. We select the column corresponding to the "st" point. We factorize the system with this column separately, by replacing the row in the matrix by the identity on the diagonal and zero on the right-hand side. The other columns in the first system are factorized in a standard way.

$$\begin{bmatrix} A^x_{11} & A^x_{12} & \cdots & 0 \\ A^x_{21} & A^x_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & A^x_{rr} = 1.0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A^x_{kk} \end{bmatrix} \begin{bmatrix} z_{1st} \\ z_{2st} \\ \vdots \\ z_{rst} \\ \vdots \\ z_{kst} \end{bmatrix} = \begin{bmatrix} y_{1st} \\ y_{2st} \\ \vdots \\ y_{rst} = 0.0 \\ \vdots \\ y_{kst} \end{bmatrix} \tag{15}$$

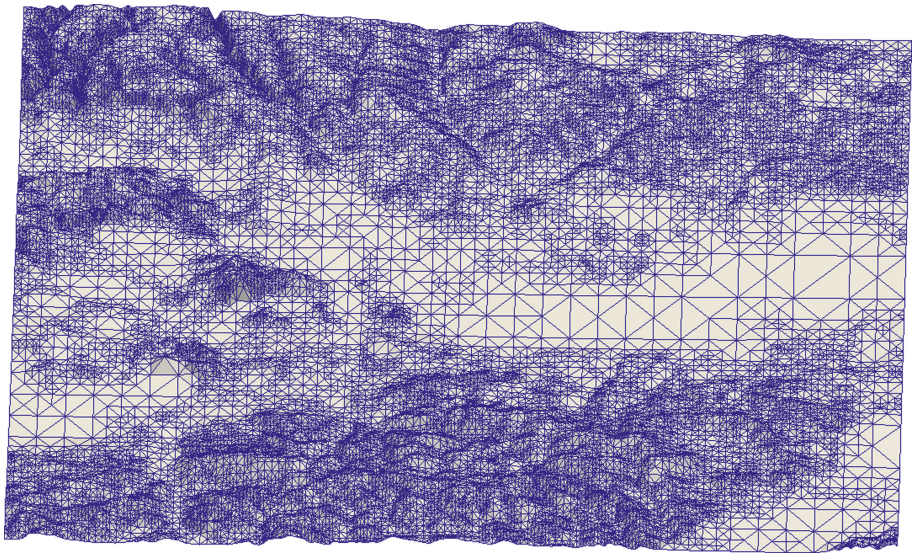

Fig. 2. The computational mesh generated based on the NASA database, representing the topography of the Krakow area.

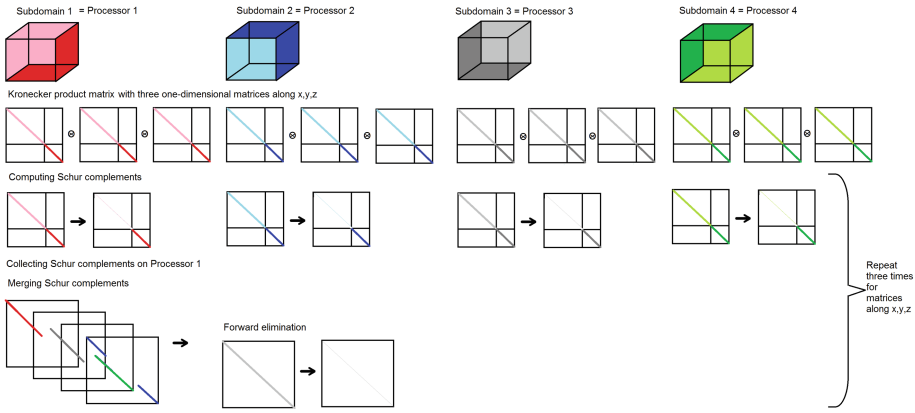

Fig. 3. Illustration on the parallel solver algorithm.

Analogous situation applies for the second system, this time with right-hand side columns representing lines perpendicular to OXZ plane. We factorize the "rt" column in the second system separately, by setting the row in the matrix as the identity on the diagonal, and using 0.0 on the right-hand side. The other columns in the second system are factorized in the standard way.

$$\begin{bmatrix} B^y_{11} & B^y_{12} & \cdots & 0 \\ B^y_{21} & B^y_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & B^y_{ss}=0 & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B^y_{ll} \end{bmatrix} \begin{bmatrix} y_{r1t} \\ y_{r2t} \\ \vdots \\ y_{rst} \\ \vdots \\ y_{rlt} \end{bmatrix} = \begin{bmatrix} z_{r1t} \\ z_{r2t} \\ \vdots \\ z_{rst}=0.0 \\ \vdots \\ z_{rlt} \end{bmatrix} \tag{16}$$

Similarly, in the third system we factorize the "rs" column separately. The other columns in the third system are factorized in a standard way.

$$\begin{bmatrix} C^z_{1,1} & C^z_{1,2} & \cdots & 0 \\ C^z_{2,1} & C^z_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & C^z_{tt}=1.0 & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C^z_{m,m} \end{bmatrix} \begin{bmatrix} x_{rs1} \\ x_{rs2} \\ \vdots \\ x_{rst} \\ \vdots \\ x_{rsm} \end{bmatrix} = \begin{bmatrix} b_{rs1} \\ b_{rs2} \\ \vdots \\ b_{rst}=0.0 \\ \vdots \\ b_{rsm} \end{bmatrix} \tag{17}$$

Using this trick for all the points in the terrain, we can factorize the Kronecker product system in a linear computational cost over the complex terrain geometry.

5 Parallel Factorization with Domain Decomposition Preserving the Linear Computational Cost

The computational domain is decomposed into several cube-shape sub-domains. We generate systems of linear equations over each sub-domain separately, and we enumerate the variables in a way that interface unknowns are located at the end of the matrices. We compute the Schur complement of the interior variables with respect to the interface variables. We do it in parallel over each of the subdomains. The important observation is that the Schur complement matrices will also be three-diagonal matrices. This is because the subdomain matrix is three-diagonal, and the Schur complement computation can be implemented as forward eliminations, performed in the three sub-systems, each of them stopped after processing the interior nodes in the particular systems. Later, we aggregate the Schur complements into one global matrix. We do it by global gather operation. This matrix is also tri-diagonal and can be factorized in a linear cost. Later, we scatter the solution, and we use the partial solutions from the global matrix to backward substitute each of the systems in parallel. These operations are illustrated in Fig. 3.

We perform this operation three times, for three submatrices of the Kronecker product matrix, defined along three axes of the coordinate system. We provide algebraic details below.

Thus, assuming we have $r-1$ rows to factorize in the first system ($r-1$ rows in the interior, and $k-r+1$ rows on the interface), we run the forward elimination over the first matrix, along the x direction, and we stop it before processing the r-th row (denoted by red color). This partial forward elimination stopped at the r-th row ensures that below that row we have the Schur complement of the first $r-1$ rows related with the interior points in the domain with respect to the next $k-r+1$ rows related with the interface points (the Schur complement is denoted by blue color). This Schur complement matrix is indeed tri-diagonal:

$$\begin{bmatrix}
A^x_{11} & A^x_{12} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\
A^x_{21} & A^x_{22} & A^x_{23} & 0 & \cdots & \cdots & \cdots & \cdots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \\
\cdots & 0 & A^x_{(r-1)(r-2)} & A^x_{(r-1)(r-1)} & A^x_{(r-1)r} & 0 & \cdots & \cdots \\
\cdots & \cdots & 0 & A^x_{r(r-1)} & A^x_{rr} & A^x_{r(r+1)} & 0 & \cdots \\
\vdots & \vdots & \ddots & \vdots & \ddots & \ddots & \ddots & \vdots \\
\cdots & \cdots & \cdots & \cdots & 0 & A^x_{(k-1)(k-2)} & A^x_{(k-1)(k-1)} & A^x_{(k-1)k} \\
\cdots & \cdots & \cdots & \cdots & \cdots & 0 & A^x_{k(k-1)} & A^x_{kk}
\end{bmatrix}$$

$$\xrightarrow{\text{Partial forward eliminations}}$$

$$\begin{bmatrix}
\tilde{A}^x_{11} & \tilde{A}^x_{12} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \tilde{A}^x_{22} & \tilde{A}^x_{23} & 0 & \cdots & \cdots & \cdots & \cdots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \\
\cdots & 0 & 0 & \tilde{A}^x_{(r-1)(r-1)} & \tilde{A}^x_{(r-1)r} & 0 & \cdots & \cdots \\
\cdots & \cdots & 0 & 0 & \tilde{A}^x_{rr} & \tilde{A}^x_{r(r+1)} & 0 & \cdots \\
\cdots & \cdots & 0 & 0 & \tilde{A}^x_{r(r+1)} & \tilde{A}^x_{(r+1)(r+1)} & \tilde{A}^x_{(r+1)(r+2)} & \cdots \\
\vdots & \vdots & \ddots & \vdots & \ddots & \ddots & \ddots & \vdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & 0 & \tilde{A}^x_{k(k-1)} & \tilde{A}^x_{kk}
\end{bmatrix} \tag{18}$$

$$\begin{bmatrix}
z_{111} & z_{121} & \cdots & z_{1lm} \\
z_{211} & z_{221} & \cdots & z_{2lm} \\
\vdots & \vdots & \ddots & \vdots \\
z_{(r-1)11} & z_{(r-1)21} & \cdots & z_{(r-1)lm} \\
z_{r11} & z_{r21} & \cdots & z_{rlm} \\
\vdots & \vdots & \ddots & \vdots \\
z_{k11} & z_{k21} & \cdots & z_{klm}
\end{bmatrix} =
\begin{bmatrix}
\tilde{y}_{111} & \tilde{y}_{121} & \cdots & \tilde{y}_{1lm} \\
\tilde{y}_{211} & \tilde{y}_{221} & \cdots & \tilde{y}_{2lm} \\
\vdots & \vdots & \ddots & \vdots \\
\tilde{y}_{(r-1)11} & \tilde{y}_{(r-1)21} & \cdots & \tilde{y}_{(r-1)lm} \\
\tilde{y}_{r11} & \tilde{y}_{r21} & \cdots & \tilde{y}_{rlm} \\
\vdots & \vdots & \ddots & \vdots \\
\tilde{y}_{k11} & \tilde{y}_{k21} & \cdots & \tilde{y}_{klm}
\end{bmatrix} \tag{19}$$

We perform this operation on every sub-domain, and then we gather on processor one the tri-diagonal Schur complements, we aggregate them into one matrix along x direction. The matrix is still a tri-diagonal matrix, and we solve the matrix using linear $\mathcal{O}(N)$ computational cost Gaussian elimination procedure with the Thomas algorithm.

Next, we scatter and substitute the partial solutions to sub-system over subdomains. We do it by replacing the last $r-k+1$ rows by the identity matrix and placing the solutions into the right-hand side blocks. Namely, on the right-hand side we replace rows from $r+1$ (denoted by blue color) by the solution obtained in the global phase, to obtain:

$$\begin{bmatrix} \tilde{A}^x_{11} & \tilde{A}^x_{12} & 0 & \cdots & \cdots & \cdots \\ 0 & \tilde{A}^x_{22} & \tilde{A}^x 23 & 0 & \cdots & \cdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \cdots & 0 & 0 & \tilde{A}^x_{(r-1)(r-1)} & \tilde{A}^x_{(r-1)r} & 0 \\ 0 & \cdots & \cdots & 0 & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \ddots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix} \quad \begin{matrix} \cdots \\ \cdots \\ \\ \cdots \\ 0 \cdots 0 \\ \ddots \vdots \\ 1 \ 0 \\ 0 \ 1 \end{matrix} \tag{20}$$

$$\begin{bmatrix} z_{111} & z_{121} & \cdots & z_{1lm} \\ z_{211} & z_{221} & \cdots & z_{2lm} \\ \vdots & \vdots & \ddots & \vdots \\ z_{(r-1)11} & z_{(r-1)21} & \cdots & z_{(r-1)lm} \\ z_{r11} & z_{r21} & \cdots & z_{rlm} \\ z_{(k-1)11} & z_{(k-1)21} & \cdots & z_{(k-1)lm} \\ z_{k11} & z_{k21} & \cdots & z_{klm} \end{bmatrix} = \begin{bmatrix} \tilde{y}_{111} & \tilde{y}_{121} & \cdots & \tilde{y}_{1lm} \\ \tilde{y}_{211} & \tilde{y}_{221} & \cdots & \tilde{y}_{2lm} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{y}_{(r-1)11} & \tilde{y}_{(r-1)21} & \cdots & \tilde{y}_{(r-1)lm} \\ \hat{z}_{r11} & \hat{z}_{r21} & \cdots & \hat{z}_{rlm} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{z}_{(k-1)11} & \hat{z}_{(k-1)21} & \cdots & \hat{z}_{(k-1)lm} \\ \hat{z}_{k11} & \hat{z}_{k21} & \cdots & \hat{z}_{klm} \end{bmatrix} \tag{21}$$

and running backward substitutions over each subdomain in parallel.

Next, we plug the solutions to the right-hand side of the second system along y axis, and we continue with the partial factorization. Now, we have $s-1$ rows in the interior and $l-s+1$ rows on the interface.

We compute the Schur complements in the same way as for the fist subsystem, thus we skip the algebraic details here. We perform this operation on every sub-domain, then we collect on processor one and aggregate the Schur complements into the global matrix along y directions. The global matrix is three-diagonal, and we solve it with Thomas algorithm. Next, we scatter and substitute the partial solutions to sub-system on each subdomain, and we solve by backward substitutions.

Finally, we plug the solution to the right-hand side of the third system along z axis, and we continue with the partial factorization. Now, we have $t-1$ rows in the interior and $m-t+1$ rows on the interface. The partial eliminations follow the same lines as for the two other directions, thus, we skip the algebraic details.

We repeat the computations for this third direction, computing the Schur complements on every sub-domain, collecting them into one global system, which is still three-diagonal, and we can solve it using the linear computational cost Thomas algorithm.

Next, we substitute the partial solution to sub-systems. We replace the last $t-m+1$ rows by the identity matrix, and place the solutions into the right-hand side, and run the backward substitutions over each subdomain in parallel.

6 Parallel Scalability

The solver is implemented in fortran95 with OpenMP (see Algorithm 1) and MPI libraries used for parallelization. It does not use any other libraries, and it is a highly optimized code. We report in Fig. 4 and Table 1 the weak scalability for three different subdomain sizes, $50 \times 50 \times 100$, $25 \times 100 \times 100$, and $50 \times 50 \times 50$. The weak scalability for the subdomains of $49 \times 49 \times 99$ internal points, shows that increasing the number of processors from 4 to 1024, simultaneously increasing the number of subdomains from 4 to 1024, and the problem size from $50 \times 50 \times 400$ to $800 \times 800 \times 400$, results in the increase of the total computational time from 120 s to 178 s for a single time step. Thus, we can perform a single time step with over 1,128,000,000 unknowns (three components

Algorithm 1. OpenMP loop parallelization

```
!!$OMP PARALLEL DO DEFAULT(PRIVATE) PRIVATE(start)
SHARED(inverse,nrhs,nint) SHARED(s_mult,d)
DO i = 1,nrhs
  start = (i-1)*(nint+2)+2; s_mult(i,1) = d(start)*inverse
END DO
!!$OMP END PARALLEL DO
```

Algorithm 2. Loop unrolling to optimize cache usage

```
loop_end =nrhs/10; i=1; inverse =1.0/(mat%b(n)-cp(n-1)*mat%a(n));
dmult =mat%a(n)
DO j = 1,loop_end
  IF(i<nrhs-10)THEN
    finish = i*(nint+2)-1
    dp(n,i) = (d(finish) - dp(n-1,i) * dmult)*inverse
    dp(n,i+1) = (d(finish+(nint+2)) - dp(n-1,i+1) * dmult)*inverse
    ...
    dp(n,i+9) = (d(finish+(nint+2)*9) - dp(n-1,i+9) * dmult)*inverse
    i=i+10
  END IF
END DO
```

of the velocity vector field, and one component of the pressure and the temperature scalar fields over 256,000,000 mesh points) within 3 min on a cluster. For the numerical verification of the code, we refer to [15].

We report in Figure 5 and Table 2 the strong scalability for six different simulations, each one with box size $50 \times 50 \times 50$, with 8, 16, 32, 64, 128 and 256 subdomains. Since the number of nodes is multiplied by the number of unknowns (three components of the velocity vector field, one component of the pressure scalar field and one component of the temperature scalar field), we obtained between $8 \times 50 \times 50 \times 5 = 5$ millions, to $256 \times 50 \times 50 \times 5 = 160$ millions of unknowns. We can read the superlinear speedup for these plots, which is related to the optimization of cache usage on smaller subdomains, with optimizing the memory transfers to the computational kernel and loop unrolling technique, as illustrated in Algorithm 2.

In Fig. 6, we show some snapshots from the preliminary simulations. In here, we focused on the description and scalability of the parallel solver algorithm,

Table 1. Weak scalability up to 1024 processors (subdomains). Each grid box contains one subdomain with $49 \times 49 \times 99$, $24 \times 99 \times 99$, or $49 \times 49 \times 49$ internal points, respectively, one subdomain per processor.

Subdomains = Processors	Grid	$50 \times 50 \times 50$ Time [s]	$25 \times 100 \times 100$ Time [s]	$50 \times 50 \times 100$ Time [s]
1	(1, 1, 1)	19	58	–
2	(1, 1, 2)	23	63	–
4	(1, 1, 4)	23	66	120
8	(2, 1, 4)	63	85	157
16	(2, 2, 4)	36	97	152
32	(4, 2, 4)	42	100	150
64	(4, 4, 4)	49	115	157
128	(8, 4, 4)	63	129	160
256	(8, 8, 4)	72	144	166
512	(16, 8, 4)	–	–	170
1024	(16, 16, 4)	–	–	178

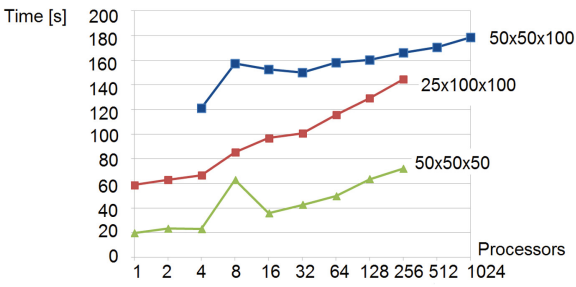

Fig. 4. Weak scalability for subdomains with $49 \times 49 \times 99$, $24 \times 99 \times 99$, and $49 \times 49 \times 49$ internal points, one subdomain per processor, up to 1024 processors (subdomains). We increase the problem size with the number of processors. For the ideal parallel code, the execution time remains constant.

Table 2. Strong scallability up to 256 processors.

Processors ndofs * 1,000,000	4	8	16	32	64	128	256
5	120	63	–	–	–	–	–
10	–	157	36	–	–	–	–
20	–	–	152	42	–	–	–
40	–	–	–	150	49	–	–
80	–	–	–	–	157	63	–
160	–	–	–	–	–	160	72

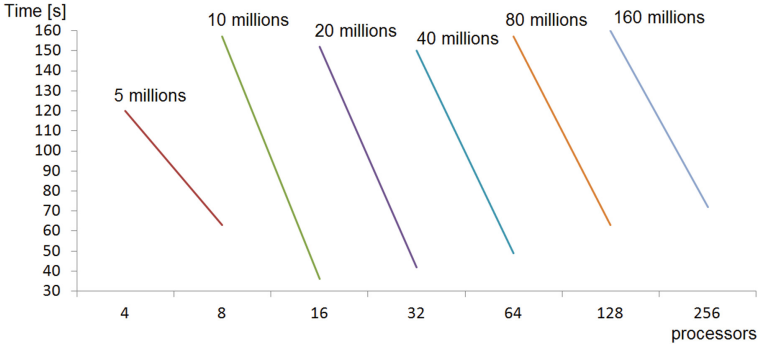

Fig. 5. Strong scallability for meshes with different sizes, for different numbers of processors. For larger meshes, it is only possible to run them on maximum number of processors.

leaving the model formulation and large massive parallel simulations of different atmospheric phenomena for the future work. This will be a challenging task itself, requiring to acquire reliable data for the initial state, forcing, and boundary conditions.

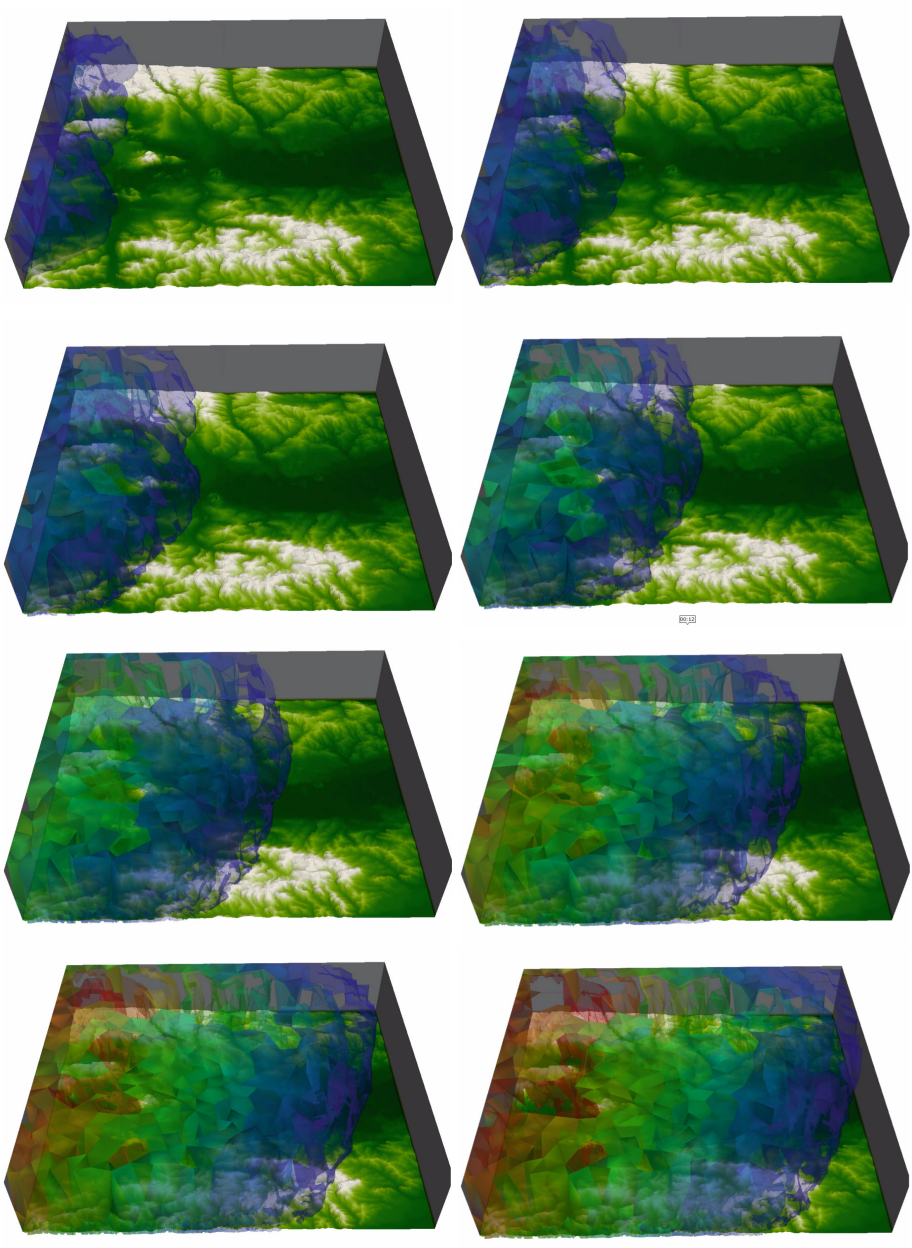

Fig. 6. Snapshots from the simulation

7 Conclusions

We described a parallel algorithm for the factorization of Kronecker product matrices. These matrices result from the finite-difference discretizations of the Navier-Stokes Boussinesq equations. The algorithm allows for simulating over the non-regular terrain topography. We showed that the Schur complements are tri-diagonal, and they can be computed, aggregated, and factorized in a linear computational cost. We analyzed the weak scalability over the PROMETHEUS Linux cluster from the CYFRONET supercomputing center. We assigned a subdomain with $50 \times 50 \times 100$ finite difference mesh to each processor, and we increased the number of processors from 4 to 1024. The total execution time for a single time step increased from 120 s to 178 s for a single time step. Thus, we could perform computations for a single time step with around 1,128,000,000 unknowns within 3 min on a Linux cluster. This corresponds to 5 scalar fields over 256,000,000 mesh points. In future work, we plan to formulate the model parameters, initial state, forcing, and boundary conditions to perform massive parallel simulations of different atmospheric phenomena.

Acknowledgments. This work and the visit of prof. Petar Minev in AGH University is supported by National Science Centre, Poland grant no. 2017/26/M/ ST1/ 00281.

References

1. European Environment Agency: Air Quality in Europe - 2017 report. 13/2017
2. Marras, S., et al.: A review of element-based Galerkin methods for numerical weather prediction: finite elements, spectral elements, and discontinuous Galerkin. Arch. Comput. Methods Eng. **23**, 673–722 (2016)
3. Song, Y., Hou, T.: Parametric vertical coordinate formulation for multiscale, Boussinesq, and nonBoussinesq ocean modeling. Ocean. Model. **11**, 298–332 (2006)
4. Schaeffer, N., Jault, D., Nataf, H.-C., Furnier, A.: Turbulent geodynamo simulations: a leap towards Earth's core. Geophys. J. Int. **211**(1), 1–29 (2017)
5. Zhang, Z., Moore, J.C.: Mathematical and Physical Fundamentals of Climate Change, Chap. 11 - Atmospheric Dynamics, pp. 347–405 (2015)
6. Zeytounian, R.: Asymptotic Modeling of Atmospheric Flows. Springer, Heidelberg (1990). https://doi.org/10.1007/978-3-642-73800-5
7. Guermond, J.-L., Minev, P.D.: High-order time stepping for the Navier-Stokes equations with minimal computational complexity. J. Comput. Appl. Math. **310**, 92–103 (2017)
8. Keating, J., Minev, P.: A fast algorithm for direct simulation of particulate flows using conforming grids. J. Comput. Phys. **255**, 486–501 (2013)
9. Bubak, M., Kitowski, J., Wiatr, K. (eds.): eScience on Distributed Computing Infrastructure. Achievements of PLGrid Plus Domain-Specific Services and Tools, vol. 8500. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10894-0
10. https://github.com/Podsiadlo/terrain
11. Farr, T.G., et al.: The Shuttle Radar Topography Mission, Reviews of Geophysics, vol. 45, no. 2 (2005)

12. Guermond, J.L., Minev, P.D.: A new class of massively parallel direction splitting for the incompressible Navier–Stokes equations. Comput. Methods Appl. Mech. Eng. **200**, 2083–2093 (2011)
13. Douglas, J., Gunn, J.E.: A general formulation of alternating direction methods. Numerische Mathematik **6**(1), 428–453 (1964)
14. Golub, G.H., Van Loan, C.: Matrix Computations, 3rd edn. John Hopkins University Press, Baltimore (1996)
15. A. Takhirov, R. Frolov, P. Minev, Direction splitting scheme for Navier-Stokes-Boussinesq system in spherical shell geometries. arXiv:1905.02300 (2019)