Optimization of the numerical algorithms of the ADREA-I mesoscale prognostic meteorological model for real-time applications

Ivan V. Kovalets a,*, Spyros Andronopoulos b, Alexander G. Venetsanos b, John G. Bartzis c

a Institute of Mathematical Machines and Systems Problems, National Academy of Sciences of Ukraine, Department of Environmental Modelling, Prospect Glushkova-42, 03187 Kiev, Ukraine
b National Centre for Scientific Research (NCSR) “Demokritos”, Institute of Nuclear Technology and Radiation Protection, Environmental Research Laboratory, 15310 Aphia Paraskevi Attikis, Greece

Received 16 June 2006; received in revised form 19 April 2007; accepted 6 May 2007
Available online 17 July 2007

Abstract

The real-time applicability of the ADREA-I prognostic mesoscale meteorological model was enhanced by applying the preconditioned BiCGSTAB method for the numerical solution of the pressure equation in combination with increasing the magnitude of the time steps up to the values allowed by the Courant number. The ILU, MILU, ILUT and ILUM preconditioning methods with different ordering strategies were used. The implementation was developed for arbitrarily complex geometries. The application of MILU(1) preconditioning and ILUT preconditioning with red–black ordering of the unknowns (RB + ILUT) has resulted in up to six times shorter overall computational time in comparison to the previously implemented line relaxation (LR) method. The feasibility of increasing the time steps has been proved by comparing the results of the 24-h ADREA-I forecasts with the observations during a real sea breeze case in Attiki, Greece: decreasing the time steps by a factor of 10 in comparison with the values allowed by the Courant number leads to decrease of the statistical error indicators by only 1–5%.

Keywords: Preconditioning; Ordering; BiCGSTAB; SIMPLER; Sea breeze; Time step; Courant number

1. Introduction

Real-time Emergency Response Systems (ERSs) are widely used for assessing the consequences and for decision support in cases of hazardous pollutants’ accidental and/or deliberate releases into the atmosphere. Examples of such systems are the RODOS system for nuclear emergency management in Europe (Raskob and Ehrhardt, 1999), the Copenhagen ERS (Baklanov et al., 2006), the TEAP system (José et al., 2007) and others. The quality of the results of these systems clearly depends on the input meteorological data. The latter usually include gridded meteorological fields, calculated by a Numerical Weather Prediction (NWP) model, operated externally in the National Weather Services.

For the case of calculations of local scale atmospheric dispersion in complex terrain (distances of the order of 10–100 km) downscaling of the NWP data is required to account for the meso- and microscale features of the flow. The downscaling procedure involves calculations on nested grids with increased spatial resolution. Since the positions of the nested domains depend on the site of interest the possibility of including a mesoscale NWP model as a part of the ERS to produce calculations on user-specified nested domains might be considered. Previously this approach was not used because of the significant computational resources required by an NWP model to calculate forecast. Nowadays this approach is used, for instance, in the Copenhagen ERS (Baklanov et al., 2006) and in the TEAP system (José et al., 2007). However, optimization of the mesoscale NWP models is still required for their successful integration with ERSs.
The ADREA-I code (Bartzis et al., 1991, 1999) has been developed in NCSR “Demokritos”. It is a three-dimensional mesoscale prognostic meteorological model especially designed for calculations of atmospheric flows in complex terrain (Sotiropoulou et al., 2004; Andronopoulos et al., 2000; Vlachogiannis et al., 2000; Varvayanni et al., 1998, 1993). The ADREA-I model is based on the numerical solution of the three-dimensional non-hydrostatic fully compressible hydrodynamic equations of the turbulent atmospheric flow. The model equations are discretized on a Cartesian grid with the finite volume method (Patankar, 1980). ADREA-I has been implemented for testing in the previous version of the RODOS system locally installed in NCSR “Demokritos” for prognoses of the meteorological situations in cases of complex terrain. However, it has not been operationally used for real-time applications of the RODOS system.

Governing equations of the mesoscale models involves fewer simplifications than the equations of large scale weather forecast models. Therefore the mesoscale models describe in more detail the physical processes in the atmosphere. Those processes result from non-adiabatic, non-hydrostatic and compressibility effects on the flow (breezes, slope flows, urban heat islands, and complex topography). The terms that account for these effects in the hydrodynamic equations impose different restrictions on the time steps \( \tau \) of the numerical integration required by stability when explicit finite-difference schemes are used. The most severe among these restrictions arise in the case of the fully compressible equations: \( \tau < \frac{1}{\max \left( \min (h_x, h_y, h_z) / c_s \right)} \) (Roache, 1972), where \( h_x, h_y, h_z \) are mesh sizes in the corresponding directions, \( c_s \approx 300 \text{ m/s} \) is the sound speed. If the sound and gravity waves are filtered by the Boussinesq and hydrostatic approximations the Courant number restriction remains: \( \tau < \frac{1}{\max \left( \sum a \frac{h_a}{u_a} \right)} \), where \( G \) is the computational domain, \( u_a \) is the velocity component in \( a \)-th direction. The latter is much less restrictive than in the compressible case, since in the atmosphere the Mach number always: \( M = U/c_s \ll 1 \) (\( U \) is the magnitude of wind velocity).

To avoid the restrictions on the time steps, the fully implicit finite-difference schemes can be used, which are unconditionally stable. However, due to nonlinearity of the operators of the hydrodynamic equations, the time steps are still restricted by the accuracy of the results. The problem of the “optimal” choice of the time step for the numerical integration of a nonlinear model remains unsolved. Therefore the time steps are selected based on empirical experience and different physical considerations. In some works (e.g., Thomas and Browning, 2001) it has been claimed that the restrictions on the time steps by the Courant number for the fully implicit schemes are usually sufficient to achieve an accurate representation of the mesoscale flow features. At the same time in some operational NWP models more severe restrictions on the time step selection appear. For instance, recommended time step in MM5 prognostic model is (Grell et al., 1994): \( \tau < 0.003 \min (h_x, h_y) \approx \min (h_x, h_z) / c_s \). That restriction is natural because the numerical scheme of MM5 is explicit in horizontal directions. In the present work the possibility to increase the time steps of numerical integration of the mesoscale model equations in the implicit schemes up to the values defined by the Courant number has been also considered.

The preconditioned Krylov subspace methods have been widely used to enhance the computational performance of different environmental models (e.g., Rao and Medina, 2006). The older successive overrelaxation (SOR) and line relaxation (LR) methods (see e.g., Ratto et al., 1994) or direct methods (Flasak and Moussiopoulos, 1988) are traditional for the meteorological applications. Still each of these methods has its advantages and difficulties. The successful application of the SOR method needs estimation of the optimal value of the relaxation parameter, which is not always possible. Direct methods, based on fast Fourier transform, can be very efficient (Flasak and Moussiopoulos, 1988), though their use is limited to the uniform horizontal grids. As for the Krylov subspace methods, despite some successful attempts (Thomas et al., 2003) in meteorological applications their use is restricted by significant grid anisotropy, which causes degradation of many preconditioning techniques (Notay, 1999). However, as it will be shown in the present work, with increasing time steps the preconditioned Krylov subspace methods get definite advantage.

Thus, the aim of the present paper was to develop an optimization method for the enhancement of ADREA-I model in view of its possible integration within the real-time mode of operation of the RODOS system. The developed method combines the preconditioned BiCGSTAB technique (Van der Vorst, 2003) with the strategy of increasing time steps up to the values defined by the Courant number. These methods are used for numerical solution of the pressure equation that arises in the ADREA/SIMPLER algorithm (Bartzis et al., 1991). In the following sections the choice of the optimization method is justified and described in detail. The performance of the optimized version of the ADREA-I model is demonstrated by computational simulations of a real sea breeze formation event in Attiki, Greece in June 2005.

2. Numerical approach

2.1. Discretization of the pressure equation

The current implementation of the ADREA-I model employs an implicit scheme for the solution of the fully compressible system of hydrodynamic equations, describing the turbulent atmospheric flow (Bartzis et al., 1991, 1999). The equations are discretized on a staggered Cartesian grid with the finite volume approach. The iterative ADREA/SIMPLER algorithm (Bartzis et al., 1991) is used to calculate the pressure and velocity fields at each time level. This algorithm leads to an iteration cycle in which the equation for the next approximation of the pressure field is solved using the variables from the current iteration level. The derivation of the ADREA/SIMPLER algorithm from the numerical approximations of the governing equations of the ADREA-I is presented in Appendix. As it is shown there, at each iteration step of the algorithm...
the following equation for pressure is to be solved (see also Eq. (23)):

\[
\frac{\rho_{ij}^n P_{ij}^{n+1} - \rho_{ij}^n}{\tau} - \tau A_x \alpha_x^r A_x P_{ij}^{n+1} - \tau A_y \alpha_y^r A_y P_{ij}^{n+1} - \tau A_z \alpha_z^r A_z P_{ij}^{n+1} = \Phi_{ijk}
\]

(1)

where \( n \) is the time level, \( s \) is the iteration number, \( P \) is the pressure, \( P^s = \frac{P^{n+1,s}}{\tau} \), \( \rho \) is the density, \( \alpha_x, \alpha_y, \alpha_z \) are positive nondimensional coefficients depending on velocities, \( A_x, A_y \) are operators approximating corresponding spatial derivatives by the forward and backward differences (see Appendix), \( \Phi = \rho^s / P^s \), and \( \Phi \) is the right hand side that depends on the variables from the previous iteration level and from the previous time step. Note that, as presented in Appendix, the ADREA/SIMPLER algorithm and Eq. (1) result from the numerical approximation of the governing system of model equations through the ADREA-I method, where Dirichlet (constant value) BC are imposed for continuity, momentum, internal energy and water mass fraction, and the following equation for pressure is to be solved (see also Appendix).

2.2. Line relaxation (LR) method

The problems (1) and (2) lead to a matrix equation. The structure of the matrix depends on the ordering of the unknowns. The natural ordering is most frequently used. In the case of rectangular domains it is defined by the relationship:

\[
l = N_x N_y (j-1) + N_y (i-1) + k = \text{ORD}_0(i,j,k).
\]

(3)

Here \( j, i, k \) are the indices in \( y, x, z \) directions correspondingly, \( N_x, N_y, N_z \) are the sizes (number of cells) of the domain in \( x \) and \( z \) directions. The numbering (3) corresponds to natural ordering with the JIK order of changing indices (i.e., index K changing first, index J last). Then Eq. (1) leads to the following system of the algebraic equations in interior domain:

\[
(A_0 \tilde{x})_i = (b_0)_i
\]

\[
= AP(l)x(l) + AU(l)x(l + 1) + AD(l)x(l - 1) + AE(l)x(l + N_x) + AW(l)x(l - N_y) + AN(l)x(l + N_y) + AS(l)x(l - N_z).
\]

(4)

Here the notation of Patankar (1980) is used, in which coefficients \( AU(l), AD(l), AW(l), AE(l), AN(l), AS(l) \) link the given node \( l \) with its upper, down, west, east, north and south neighbours correspondingly. Thus, the matrix \( A_0 \) is seven diagonal with \( AU, AD, AN \) representing its upper part and \( AD, AW, AE, AS \) its lower part. The following conditions hold in the domain interior:

\[
\begin{align*}
\text{AU}(l) &= AD(l + 1) \\
\text{AE}(l) &= AW(l + N_x) \\
\text{AN}(l) &= AS(l + N_y)
\end{align*}
\]

(5)

which follow from the conservative form of Eqs. (1) and (2) and imply that the matrix \( A_0 \) is symmetric. However, asymmetry can be introduced by the boundary conditions. For instance, the following hold near the boundaries:

\[
\begin{align*}
\text{AW}(l_0) &= 0 & l_0 &= l(N_y, j, k) \\
\text{AE}(l_0 - N_x) &\neq 0
\end{align*}
\]

(6)

when the term in square brackets of Eq. (2) is equal to zero. Prior to numerical solution Eq. (4) is normalised, so that \( \text{AP}(l) = 1, \forall l \).

For Eq. (1) the following relationship holds:

\[
\begin{align*}
0 &< -(\text{AU}(l) + \text{AD}(l) + \text{AE}(l) + \text{AW}(l) + \text{AN}(l) + \text{AS}(l)) \\
&< \text{AP}(l), \text{AU}(l) \leq 0, \text{AD}(l) \leq 0, \text{AE}(l) \leq 0, \text{AW}(l) \leq 0, \text{AN}(l) \leq 0, \text{AS}(l) \leq 0, \forall l
\end{align*}
\]

(7)

Following Theorem 2.4.14 from Orthega and Rheinboldt (1970) inequalities (7) guarantee that the matrix \( A_0 \) is strongly diagonally dominant \( M \)-matrix.

When domains with complex geometries are discretized by Cartesian rectangular finite volumes, with constant number of control volumes in each direction as is the case for ADREA-I, blocked cells appear which fall under the ground surface. The blocked cells are excluded from the solution vector and the ordering (3) is modified in the following way:

\[
\text{nout} = 0 \\
do(j,i,k)
\begin{cases}
 l = \text{ORD}_0(i,j,k) - \text{nout}, (i,j,k) \notin \text{blocked} \\
 \text{nout} = \text{nout} + 1; (i,j,k) \in \text{blocked}
\end{cases}
\]

(8)

The matrix \( A \) of the modified system \( A \tilde{x} = b \) has not any more seven-diagonal structure. However, it is easy to see that renumbering does not change either the symmetry of matrix or the order of the nonzero elements in a given row. The \( M \)-property of the matrix is also preserved. Therefore for the sake of simplicity the discussion below will refer to the unmodified ordering.

2.2. Line relaxation (LR) method

The previously implemented in ADREA-I method of “line relaxations” resembles the method of successive overrelaxation. It has been frequently used in diagnostic and prognostic
meteorological models (e.g., Thomas et al., 2003). If written in matrix form it can be expressed as follows:

\[
(L + D + U) x^{l+1} = -U x^l + b
\]  

(9)

Here matrix \( L \) is the lower triangular part of the matrix \( A \), matrix \( D \) is the diagonal part of \( A \), \( U_1 \) is a matrix that includes the first upper diagonal of \( A \), \( U(l, l+1) = AU(l) \). Matrix \( U_2 \) contains all other upper diagonals of \( A \). Implementation of the iteration step (9) is easily transformed to the solution of \( N_hN_v \) systems of equations with three-diagonal matrix of the size \( N_h \).

Since the “lines” are oriented vertically, the rate of convergence of the LR method can become high for anisotropic grids, when vertical links are much stronger than horizontal. Indeed, the relative value of the matrix coefficients is approximately:

\[
\begin{align*}
\frac{\alpha_{j}S_{xz}}{\alpha_{j}h_{j}h_{z}} & \sim \frac{\alpha_{j}h_{j}^2}{\alpha_{j}h_{z}^2} \\
& = \frac{R}{N} 
\end{align*}
\]  

(10)

Here \( S_{xz} \) is the \( xz \) cell surface area, \( h_{j}, h_{z} \) are the vertical and horizontal mesh sizes. When \( h_{j}/h_{z} \ll 1 \), then \( R \ll 1 \) and the LR method will obviously converge fast.

However, the LR method loses its efficiency when the time integration step \( \tau \) increases. An example of sharp increase in computational time needed for solution of the pressure equation with increasing time step is shown in Fig. 2. As it is clearly seen from this figure, the time needed for solution of pressure equation is growing rapidly: from 0.1 s to 60 s with \( \tau \) changing from 1 s to 400 s. Such rapid increase in computational time indicates the rise of the condition number of the system \( \kappa = \lambda_{\max}/\lambda_{\min} \) with rising \( \tau \) (\( \lambda_{\max} > 0, \lambda_{\min} > 0 \) are maximum and minimum eigenvalues of the system). For the rest of the variables (humidity, temperature, and turbulent kinetic energy) the solution time raises much slower reaching at most 1 s. Due to such rapid increase in computational time with increasing \( \tau \) very small time steps are to be used when LR method is applied. For instance, in the test case presented below \( \tau = 16 \) s which is much less than the time step imposed by the Courant number. For such small time steps the LR method converged in about 30 iterations. This is very fast convergence for the number of active cells \( \approx 5 \times 10^4 \). And even in that case, as follows from Fig. 2 the pressure equation demands the majority of computational time in comparison with the other variables. Thus, the obvious strategy to optimise the ADREA-I model is to enhance the solution of the pressure equation together with increasing time steps.

2.3. Preconditioned BiCGSTAB method

The preconditioned conjugate gradients (CG) method of numerical solution of the matrix equations is one of the most popular methods for the case of Symmetric Positive Definite (SPD) \( M \)-matrices (Van der Vorst, 2003). For the case of non-symmetric PD matrices the BiCGSTAB method has been developed by Van der Vorst (2003). It has been widely recognized as one of the best methods for such matrices. As it was mentioned above, asymmetry can be introduced by the boundary conditions, however, it is easy to see that the matrix remains positive definite and preserves the \( M \)-property. Thus the BiCGSTAB method was chosen to accelerate the solution of the pressure equation under large time steps.

As in the CG method, in BiCGSTAB, the residual vectors, the basis vectors and the next approximation for the solution vector are built recursively at each iteration step in the way that minimizes the residual in certain sense. One iteration step consists of several matrix–vector multiplications. Thus the number of operations in one iteration step is \( O(N) \). In each iteration step of the preconditioned BiCGSTAB algorithm the auxiliary system of equations with the matrix \( K = A \) is to be solved twice (preconditioning step). The matrix \( K \) is called “preconditioner” and its choice is critical for the convergence of the BiCGSTAB method. The preconditioning step reduces the condition number of the system and can greatly increase convergence.

Effective preconditioners for \( M \)-matrices can be constructed with the Incomplete LU (ILU) factorization method (Van der Vorst, 2003). The ILU preconditioner is defined as the approximation \( K = L \) of the matrix \( A \), with the upper and bottom triangular matrices \( U \) and \( L \). Matrix \( L \) has the unit diagonal, and in both matrices \( U \) and \( L \), nonzero elements appear only in positions defined by the nonzero stencil:

\[
K = L \quad U, \quad l(i,j) \neq 0, u(i,j) \neq 0 \iff i, j \in \text{Stenc}
\]  

(11)

The choice of the nonzero stencil is critical for the quality of preconditioner. If the nonzero stencil includes arbitrary pairs of indices, then an exact LU factorization is obtained. The latter requires \( O(N^3) \) operations, thus the nonzero stencil should be sparse enough to be practical. The simplest choice of sparse nonzero stencil is the coincidence with the nonzero stencil of the original matrix \( A \). This choice leads to ILU(0) preconditioner (stencil of level zero). Adding some new nonzero entries in stencil could make ILU more exact and thus more effective. At the same time increasing the number of nonzero entries in the stencil increases the number of operations performed per iteration in the BiCGSTAB algorithm. Thus, only positions containing elements of relatively large size should be chosen. The question to locate these positions was studied intensively and the concept of the “level of fill in” has been introduced. The latter works well for the \( M \)-matrices and is defined through the graph representation of matrix (Van der Vorst, 2003). The nonzero stencil based on the “level of fill in” \( N_{\text{lev}} \) is defined as the set of such column numbers \( j \) in the row \( i \), that the minimum length of the path from the node \( j \) to the node \( i \) in the matrix graph is not higher than \( N_{\text{lev}} \). The preconditioners, in which the nonzero stencil was defined by the “level of fill in” were called as ILU(\( N_{\text{lev}} \)).

With the given nonzero stencil the construction of the \( L \) and \( U \) factors is achieved with a version of the Gaussian elimination algorithm. In the present work the so-called IKJ version of the LU factorization algorithm (Saad, 2003) was implemented. In this algorithm the coefficients of the \( L \) and
U factors are calculated from the matrix coefficients in the following way:

1. For $i = 2, N$ Do :
2. For $k = 1, i - 1$, and if $(i, k) \in \text{Stenc}$, Do :
3. $a_{ik} = a_{ik}/a_{kk}$
4. For $j = k + 1, N$ and if $(i, j) \in \text{Stenc}$, Do :
5. $a_{ij} = a_{ij} - a_{ik}a_{kj}$
6. EndDo
7. EndDo
8. EndDo

The first cycle in the algorithm (12) is cycle by rows of matrix (i.e., in vertical direction). The other two cycles are cycles by columns (i.e., in horizontal from the left to the right direction). Thus, in this version of the Gaussian Elimination algorithm when the values in the current row are modified, they depend only on the values in the upper rows (previously processed). The stencil in the algorithm (12) can correspond either to the stencil of the given level $N_{\text{lev}}$, when ILU($N_{\text{lev}}$) is constructed, or to any other nonzero stencil.

In the Modified ILU factorization MILU($N_{\text{lev}}$) (Gustaffson, 1996), when current row is processed in the algorithm (12) the elements of the $L$ and $U$ factors are first calculated for the stencil of the $N_{\text{lev}} + 1$ level. Then the elements in positions, not coinciding with the stencil of the $N_{\text{lev}}$-th level are summarized, subtracted from the main diagonal and then set to zero. In this way the constructed $L$ and $U$ factors have the following property: $A = L U e$ (Saad, 2003). In other words the MILU factorization is exact for unit vectors.

The rate of convergence of ILU and MILU methods for general matrices had been studied in Axelsson and Lu (1997). Convergence for matrices arising from approximations of two-dimensional elliptic problems with Dirichlet boundary conditions had been studied in Gustaffson (1996). The rate of convergence of ILU preconditioner for such problems is: $N_{\text{oper}} \sim O(N^{1.5})$ where $N_{\text{oper}}$ is the number of operations.
needed for convergence. For the MILU the convergence is faster: \( N_\text{oper} \sim O(N^{1.25}) \) (Gustaffson, 1996). That estimation is close to the “grid-independent” convergence: \( N_\text{oper} \sim O(N) \). The MILU factorization is stable for the diagonally dominant \( M \)-matrices (Brand, 1992), and thus is feasible for the solution of (1).

Contrary to the above “level of fill in” concept, in the ILUT preconditioning strategy (Saad, 2003), the elements calculated during step 5 of the algorithm (12) are dropped if their absolute values are less than a given fraction of the norm of the currently processed row: \( |a_{ij}| \leq \omega \| \mathbf{a} \| = \omega |a_{ii}| \). Here \( \omega \) is threshold value, and the last equality is due to (7). Thus the nonzero stencil is defined in the ILUT approach dynamically, and the nonzero stencil calculated based on the “level of fill” concept (denoted by “Stenc” in the algorithm (12)) can be considered as the “first approximation” to the nonzero stencil of the ILUT. In the present work the ILUT approach was also combined with the “diagonal modification strategy”, which was described above. In the present study it had been found empirically that the values of \( \omega = 0.01 \) were nearly optimal. However, the performance of ILUT was not very sensitive to the values of \( \omega \). In all studied cases the performance with ILUT reduced not more than by the factor of 2 within the range of \( \omega : 0.001 \leq \omega \leq 0.1 \).

The performance of the ILU preconditioned BiCGSTAB method depends also on the ordering of the unknowns. For the case of vertically stretched anisotropic grids the natural JIK or IJK orderings (horizontal indices changing first) have advantage over the KJI ordering (D’Azevedo et al., 1992).

The red–black ordering of the unknowns (Saad, 2003) leads to the block partitioning of the system of equations:

\[
\begin{pmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{pmatrix}
\begin{pmatrix}
\mathbf{x}_R \\
\mathbf{x}_B
\end{pmatrix}
= 
\begin{pmatrix}
\mathbf{b}_R \\
\mathbf{b}_B
\end{pmatrix}
\]

If red–black ordering is exact \( A_{11} = \mathbf{I}, A_{22} = \mathbf{I} \) (I is identity matrix). In that case red nodes are linked only to black nodes and vice versa. Thus the red unknowns can be eliminated, leading to the system of equations for the black unknowns:

\[
\mathbf{S}_B \mathbf{x}_B = \left( \mathbf{I} - A_{21} A_{12} \right) \mathbf{x}_B = \left( \mathbf{b}_B - A_{21} \mathbf{b}_R \right) = \mathbf{b}_S
\]

The matrix \( \mathbf{S}_B \) is called Schur complement matrix. There is no need to calculate and store matrix \( \mathbf{S}_B \) itself, because in the BiCGSTAB algorithm only matrix–vector products \( \mathbf{w} = \mathbf{S}_B \mathbf{v} \) are needed, which can be calculated in the following way (Saad, 2003):

\[
\mathbf{v}_1 = A_{12} \mathbf{v} \\
\mathbf{w} = \mathbf{v} - A_{21} \mathbf{v}_1
\]

The ILU(0) preconditioner of the matrix \( \mathbf{S} \) can be calculated with the algorithm (12) as the bottom corner block of the ILU(1) factorization of the original matrix — the so-called “induced preconditioner” (Saad, 2003).

The more advanced multilevel approach close to ILUM (Saad, 2003), GILUM (Zhang, 2001) and MRILU (Botta and Wubs, 1999) had been implemented. It is based on the idea of the repeated approximate red–black ordering. In this multilevel approach the corresponding to level 1 nodes of Schur complement matrix (15) are further reordered to form almost independent set of nodes (which are again called “red” and the rest “black”). The modified Greedy algorithm (Zhang, 2001) is used to find those almost independent sets of nodes.

### Table 1

<table>
<thead>
<tr>
<th>Case no.</th>
<th>Case definition</th>
<th>Horizontal mesh size (m)</th>
<th>Time step ( \tau ) (s) for MILU(1) and RB + ILUT</th>
<th>Computational time (min) of different methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20 June 2005, Attiki</td>
<td>4000</td>
<td>200</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>20 June 2005, Attiki</td>
<td>2000</td>
<td>100</td>
<td>285</td>
</tr>
<tr>
<td>3</td>
<td>Same conditions as case 1 but with increased inlet wind velocities</td>
<td>4000</td>
<td>160</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>Same conditions as case 1 but with reduced inlet wind velocities</td>
<td>4000</td>
<td>250</td>
<td>30</td>
</tr>
</tbody>
</table>

Calculations were performed with Pentium-IV, 3 GHz.
If the two sets of nodes are not completely independent the diagonal blocks in the matrix (13) are not anymore identity matrices. However, when the links between the nodes in the first ("red") set are weak the $A_{11}$ block is strongly diagonally dominant matrix and can be approximated with the diagonal matrix (as it is done in the MRILU approach). Then the Schur complement of the second level is calculated. This recursive procedure is repeated up to the maximum specified level. For large CFD problems that maximum level should not be too high (Weijer et al., 2003) and in the present study three levels were used. The obtained approximate block factorization of the original matrix is then used as preconditioner.

The described BiCGSTAB algorithm, together with preconditioners (ILU($N_{lev}$), MILU($N_{lev}$), ILUT, and ILUM), and the different orderings (natural JIK, KJI, and red–black) have been implemented in a software library. It uses the Illpack–Ellpack format (Saad, 2003) for the storage of sparse matrices. This format is very effective when working with matrices

Fig. 3. Calculated and measured time histories of the wind velocity and wind direction for three meteorological stations. Symbols — measurements: (1), "—" $h_x = h_y = 4$ km, $t = 16$ s; (2), "- - - -" $h_x = h_y = 4$ km, $t = 200$ s; (3), "—" $h_x = h_y = 2$ km, $t = 16$ s; and (4), "- - - -" $h_x = h_y = 2$ km, $t = 100$ s.
having an approximately fixed number of nonzero elements in one row. However, for the multilevel preconditioners this format appeared to be too much memory consuming and the more economical Compressed Sparse Row (CSR) format (Saad, 2003) will be used in the next developments.

3. Results of calculations

To evaluate the performance of the newly implemented methods in ADREA-I, computational simulations of a real case sea breeze formation event in the area of Attiki, Greece, have been performed. The date of the event was 20 June 2005. The computational domain is shown in Fig. 1 together with the locations of the ground meteorological stations that provided the measurements used for comparison purposes. The x- and y-axis were taken along the west to east and south to north directions, respectively, and z-axis vertically upwards. Four sets of calculations were performed, which are summarized in Table 1. In cases 1, 3 and 4 with a horizontal grid resolution of $4 \times 4$ km$^2$ the computational domain was discretized by $46 \times 46 \times 29$ cells in x, y, and z directions, respectively. In case 2 with a finer ($2 \times 2$ km$^2$) horizontal grid resolution the number of grid cells in horizontal directions was increased to 92. Rawinsonde measurements were used to initialize the vertical profiles of wind, temperature and humidity. Initial vertical profiles of turbulent kinetic energy were initialized with the stationary solution of the one-dimensional problem, describing the vertical turbulent momentum transport in the atmosphere. The average climatic values for the specific season were used to initialize the sea-surface and land-surface temperatures and the soil humidity. In case 3 the initial profile of the magnitude of the wind velocity was increased by the constant value $\delta U = U_{10} = 3.8$ m/s equal to the measured 10-m wind speed. Thus, the winds in the surface-layer were increased by the factor of 2. On the contrary, in case 4 the initial profile of the magnitude of the wind velocity was reduced by the constant value $\delta U = -0.5U_{10} = -1.9$ m/s, decreasing the winds in the surface-layer by the factor of 2. Though in cases 3 and 4 comparisons with measurements were impossible, they allowed more comprehensive testing of the developed algorithms and, as it will be seen below, provided more support to the strategy of increasing the time steps up to the Courant number.

In all cases runs with “small” and “large” (i.e., defined by the Courant limit) time steps were performed. In all cases the “small” time step was the same: $\tau = 16$ s. The “large” time steps were defined by the Courant limit: $\tau = \min_G(\sum_i h_i/|u_i|)$. The corresponding values of $\tau$ for cases 1—4 are presented in Table 1 (column 4).

The specific day was characterized by low surface wind speed $U_{10} = 3.8$ m/s and a pronounced sea breeze development during the daytime had been observed. The calculations started at 00:00 hours local time. The effect of the time step on the performance of the LR and the different types of preconditioned BiCGSTAB methods is shown in Fig. 2. It is obvious from Fig. 2 that for small time steps $\tau < 25$ s all methods perform worse than the line relaxation (LR) method. This is because for small time steps line relaxation takes great

![Fig. 4. Calculated and measured time histories of the temperature for three meteorological stations. Symbols — measurements; (1), "- - - -" $h_x = h_y = 4$ km, $\tau = 16$ s; (2), "- - - - -" $h_x = h_y = 4$ km, $\tau = 200$ s; (3), "- - - -" $h_x = h_y = 2$ km, $\tau = 16$ s; and (4), "- - - - -" $h_x = h_y = 2$ km, $\tau = 100$ s.](image)
advantage of the coefficients anisotropy in vertical direction, as it was discussed above. However, with increasing time steps the effect of ill-conditioning becomes more pronounced and the preconditioned BiCGSTAB methods perform much faster (up to 30 times for $\tau = 400$ s) than the LR method. Asymptotically the fastest methods are MILU(1) with natural KJI ordering and ILUT with red–black ordering. The ILU(2), MILU(2) and the ILUM methods (not shown in Fig. 2) performed several times worse than the MILU(1), requiring time for solution of pressure equation about 8 s when $\tau = 100$ s. Therefore they were not used in further calculations. The degradation of the MILU(2) performance happens because of the increased number of the multiplications with increasing the level of fill. The poor performance of ILUM was possibly due to the lacks of the Greedy algorithm, which both theoretically (Saad, 2003) and practically cannot find the set of independent nodes of the maximum possible size.

The total computational times needed by ADREA-I for the 24-h simulation with the new methods (MILU(1) and RB + ILUT) using the large time steps and with the old method (LR) using the small time steps are compared in Table 1. As it can be seen from Table 1 an overall level of improvement by a factor of $5-6$ was achieved in cases 1, 3 and 4 on the coarse grid and by a factor of $3.5-4.5$ in case 2 on the fine grid. In all cases, presented in Table 1 the RB + ILUT method was used with the same value of $\omega = 0.01$. The same levels of improvement, achieved with that method support the above assumption that RB + ILUT is not very sensitive to the values of $\omega$.

In case 2 increasing the spatial resolution by a factor of 2 in comparison with case 1 leads to increase in the size of the solution vector by a factor of 4: $N_2 \approx 4N_1$. As it is seen from Table 1 in case 2 the total computational time with the MILU(1) and RB + ILUT methods increased by a factor of 7.5: $T_{calc2} \approx 7.5T_{calc1}$. Thus the number of operations needed to solve the pressure Eq. (1) per one time step increased by a factor of $N_{oper2}/N_{oper1} = (T_{calc2}/T_{calc1}) \approx 3.8 \approx N_2/N_1$. Thus, almost grid independent convergence rate is observed here. This is consistent with the abovementioned theoretical results proved by Gustaffson (1996) for idealized problems.

The possibility to increase the time steps was verified by comparisons of the calculation results (in cases 1 and 2) with the meteorological measurements for the specific day. The comparisons were performed against the data from the eight meteorological stations, which locations are shown in Fig. 1. At all the stations wind speed, wind direction and temperature were measured. The comparisons of the predicted vs. measured data (wind velocities, wind directions, and temperatures) for three stations are shown in Figs. 3 and 4. Calculations with the small time steps are shown as the solid lines, while the calculations with the time steps, defined by the Courant limit, are shown as the dashed lines. Both observed and predicted time histories of wind demonstrate pronounced sea breeze development. The calculated wind speeds and temperatures are close enough to the measured values. Figs. 3 and 4 also demonstrate improvement of the calculation agreement with the measurements with increasing grid resolution.

Fig. 5 shows the calculated surface wind field in the domain of calculations, which also reveals characteristic sea breeze flow features in daytime. As it can be seen from Figs. 3–5, increasing the time steps in both cases—fine and coarse resolution — has negligible effect on the quality of the calculated results.

The statistical indicators (root mean square deviations of the wind velocities and air temperatures) of the errors of calculations in comparison with the measurements are presented in Table 2. As it is seen from Table 2, under the influence of the increased time steps all statistical characteristics of error increase only by 1–5% of the values obtained with the small time steps.
Table 2
Statistical comparisons between calculations and measurements

<table>
<thead>
<tr>
<th>Case No.</th>
<th>rmsu (m/s)</th>
<th>rmst (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>With (\tau = 16) s</td>
<td>With large (\tau)</td>
</tr>
<tr>
<td>1</td>
<td>1.69</td>
<td>1.72</td>
</tr>
<tr>
<td>2</td>
<td>1.26</td>
<td>1.22</td>
</tr>
</tbody>
</table>

\(\text{rmsu} = \text{root mean square deviation of wind velocity; } \text{rmst} = \text{root mean square deviation of temperature; Case no. corresponds to Table 1; large values of } \tau \text{ correspond to column 4 of Table 1.}\)

Table 3 presents the root mean square differences in velocity magnitude \(\sigma_u\) and temperature \(\sigma_T\) between results, calculated with the small and large time steps in the runs 1–4. As it is seen from Table 3 in all cases \(\sigma_u \leq 0.2\) m/s, \(\sigma_T \leq 0.28\) °C. Both values are essentially smaller than the corresponding root mean square errors, presented in Table 2. Thus, from the above it can be concluded that the strategy of increasing the time steps up to the values restricted by the Courant number is justified by the accuracy of the obtained results.

4. Conclusions

In the present work the improvement in computational speed of the ADREA-I mesoscale prognostic meteorological model was considered. The way of achieving this aim was to use the preconditioned BiCGSTAB method for the solution of the pressure equation together with increasing the time steps up to the values restricted by the Courant number. Different preconditioning strategies (ILU, MILU, ILUT, and ILUM) and different kinds of orderings in combination with the BiCGSTAB method were implemented in the software library, especially designed for calculations in conjunction with CFD codes like ADREA-I. The implemented methods were tested by performing computational simulations of a real sea breeze formation event in Attiki, Greece, on June 25th, 2005, and by comparing the calculations results with measured meteorological data for the specific day. Increasing the time steps was justified by the accuracy of the results achieved. Using time steps defined by the Courant limit leads to the increase of the root mean square errors of the wind velocity and temperature only by 1–5% in comparison with the case, when calculations were done with significantly (by the factor of 5–10) smaller time steps.

The previously implemented line relaxation method appeared to be inapplicable when larger time steps were used, since it led to very high computational times. The preconditioned BiCGSTAB method significantly improved the situation. The best levels of improvement were achieved for the case of MILU(1) preconditioning and ILUT preconditioning combined with the red—black orderings of unknowns. The overall levels of improvement achieved with the preconditioned BiCGSTAB method and time steps restricted by the Courant limit in comparison with the old LR method and small time steps were by a factor of 5–6 in the case of the coarser (4 km) grid and by a factor of 3.5–4.5 in the case of the finer (2 km) grid. The overall time needed for the calculation of the 24-h forecast with the new methods was about 30 min in the case of the 4-km grid. This appears to be close to the requirements of the real-time applicability of the ADREA-I model for emergency response.

Acknowledgements

The present work has been fully supported by the European Commission through the EURATOM grant RODOS/METADM, contract No. 516492 (FI6R). Authors thank Nikos Gounaris for making Fig. 1 and four anonymous referees for their useful comments. The first author thanks all staff of Environmental Research Laboratory due to very fruitful discussions on the presented topic and all kinds of support during his stay as a visiting scientist there.

Appendix. Governing equations and the ADREA/SIMPLER algorithm

Here some essential details concerning the numerical approximation of the model equations and the ADREA/SIMPLER algorithm are given. More details concerning the derivation of the governing equations can be found in Houssadas et al. (1991) and Bartzis et al. (1999), and concerning the numerical scheme in Bartzis et al. (1991). The governing system of equations of the ADREA-I model consists of the equations for the moist air—liquid mixture mass, momentum, internal energy and water mass fraction together with the state equation for the ideal gas. In the tensor notation those equations are:

\[
\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_\beta}{\partial x_\beta} = 0 \quad (16)
\]

\[
\frac{\partial u_\beta}{\partial t} + \frac{1}{\rho} \frac{\partial P}{\partial x_\beta} = F_{\text{uf}} - \frac{1}{\rho} \frac{\partial \rho q_i (1 - q_i) u_{\alpha 2} u_{\alpha 3}}{\partial x_\alpha} + g_\beta \quad (17)
\]

\[
\frac{\partial e}{\partial t} = -u_\alpha \frac{\partial e}{\partial x_\alpha} - \frac{P}{\rho} \frac{\partial u_\alpha}{\partial x_\alpha} + \frac{1}{\rho} \frac{\partial \rho q_i (1 - q_i) u_{\alpha 2} (e_i - e_l)}{\partial x_\alpha} + \frac{P}{\rho} \frac{\partial q_i u_{\alpha 2}}{\partial x_\alpha} \quad (18)
\]

Table 3
Root mean square differences in prediction of the wind velocity \((\sigma_u)\) and temperature \((\sigma_T)\) with the “large” and “small” time steps

<table>
<thead>
<tr>
<th>Case no.</th>
<th>(\sigma_u) (m/s)</th>
<th>(\sigma_T) (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.15</td>
<td>0.24</td>
</tr>
<tr>
<td>2</td>
<td>0.09</td>
<td>0.11</td>
</tr>
<tr>
<td>3</td>
<td>0.18</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>0.2</td>
<td>0.28</td>
</tr>
</tbody>
</table>

Case descriptions and corresponding time steps are given in Table 1.
Here $v_1$, $v_2$, $v_3$ are the coordinates, $u_a = (u, v, w)$, $g_{ax}$, $u_{ax}$, $K_{ax}$ are the components of wind velocity, gravity acceleration, slip velocity, momentum eddy viscosity in $a$-th direction ($\alpha = 1, 2, 3$), $\rho$ is the density, $e, c_i$ are the specific moist air internal energy and water substance internal energy, $q_a, q_1, q_2$ are the water substance, water substance liquid and water substance vapour mass fractions, $T$ is the temperature, $P$ is the pressure, $c_p$ is the moist air specific heat capacity, $\omega_\theta$ is the component of the angular velocity of the Earth rotation velocity vector. The governing equations are complemented with the turbulence parameterization, involving the transport equation for the turbulent kinetic energy, heat conduction equation in the surface soil layer, parameterizations of the basic physical processes such as heat exchange with Earth surface, water condensation, rainfall velocity, and solar radiation (Bartzis et al., 1999).

Note, that despite Eqs. (17)–(19) are presented in non-conservative form, the conservative form of those equations is used for the numerical approximation. However, using the approach of Patankar (1980), the resulting approximation is equivalent to some particular approximation of the non-conservative form of Eqs. (17)–(19). For the presentation of the ADREA/SIMPLER algorithm the non-conservative form is preferable and therefore used hereafter.

Eqs. (16)–(19) are approximated with the finite volume method on the staggered Cartesian grid. Thus all the scalar variables are defined on the same set of grid nodes. The grid for the $u$ component of wind velocity is shifted in $x$ direction with respect to the original scalar grid. Analogously for $v$ and $w$ components the grids are shifted in $y$ and $z$ directions, respectively, with respect to the scalar grid.

In the regular cells of the computational domain (rectangular cells without solid obstacles inside) Eq. (16) is approximated as:

$$
\begin{align*}
\rho_{ijk}^{n+1} - \rho_{ijk}^n &= \frac{\tau}{h_x} \left[ \rho_{ijk+1/2} u_{ijk+1/2}^{n+1} - \rho_{ijk-1/2} u_{ijk-1/2}^{n+1} + \rho_{ijk+1/2}^s v_{ijk+1/2}^{n+1} - \rho_{ijk-1/2}^s v_{ijk-1/2}^{n+1} \right] \\
&+ \frac{\tau}{h_y} \left[ \rho_{ijk+1/2}^s v_{ijk+1/2}^{n+1} - \rho_{ijk-1/2}^s v_{ijk-1/2}^{n+1} \right] \\
&+ \frac{\tau}{h_z} \left[ \rho_{ijk+1/2}^s w_{ijk+1/2}^{n+1} - \rho_{ijk-1/2}^s w_{ijk-1/2}^{n+1} \right] \\
&= \frac{\tau}{h_x} \rho_{ijk} \frac{\partial u_{ijk}^{n+1}}{\partial x} + \frac{\tau}{h_y} \rho_{ijk} \frac{\partial v_{ijk}^{n+1}}{\partial y} + \frac{\tau}{h_z} \rho_{ijk} \frac{\partial w_{ijk}^{n+1}}{\partial z} \\
&+ A_x \left( \rho u_{ijk}^{n+1} \right)_{ijk+1/2} - A_x \left( \rho u_{ijk}^{n+1} \right)_{ijk-1/2} = 0
\end{align*}
$$

where $s$ is the iteration number, $n$ is the time layer, $\rho_{ijk}^{n+1}, A_x, A_y, A_z$ are operators, approximating the derivative by forward difference: $A_x \phi_{ijk} = (\phi_{i+1,j,k} - \phi_{ijk})/h_x$. The values of the scalar variable between nodes: $\rho_{ijk}$ are obtained by the first order interpolation method. The approximation for the momentum Eq. (17) is ($\beta = 1$):

$$
\begin{align*}
\frac{u_{ijk}^{n+1} - u_{ijk}^n}{\tau} &= -\frac{1}{\rho_{ijk}} A_x \frac{P_{ijk}}{\rho} \\
&= F_{u,ijk} - \delta_{ijk} u_{ijk}^{n+1} \\
&= \frac{\tau}{h_x} \rho_{ijk} \frac{\partial u_{ijk}^{n+1}}{\partial x} - \frac{\tau}{h_y} \rho_{ijk} \frac{\partial u_{ijk}^{n+1}}{\partial y} - \frac{\tau}{h_z} \rho_{ijk} \frac{\partial u_{ijk}^{n+1}}{\partial z} + F_{u,ijk} \\
&= \frac{\tau}{h_x} \rho_{ijk} \frac{\partial u_{ijk}^{n+1}}{\partial x} - \frac{\tau}{h_y} \rho_{ijk} \frac{\partial u_{ijk}^{n+1}}{\partial y} - \frac{\tau}{h_z} \rho_{ijk} \frac{\partial u_{ijk}^{n+1}}{\partial z} + F_{u,ijk}
\end{align*}
$$

Here $F_{u,ijk}$ is numerical approximation of the right hand side of Eq. (17), depending on the values of the state vector at both $s$ and $s + 1$ iteration levels; $F_{u,ijk}$ is part of $F_{u,ijk}$, containing variables only at $s$ iteration level. $A_x$ is operator, approximating the derivative by the backward difference: $A_x \phi_{ijk} = (\phi_{i-1,j,k} - \phi_{ijk})/h_x$. The values of $\delta_{ijk}$ are equal to the diagonal elements of the matrix representing the numerical approximation of the diffusion–convection operator: $(\rho_0 \partial u/\partial x_a) - (1/\rho) (\partial x_a \rho K_{ax} (\partial u/\partial x_a))_{i,j,k} \approx (\Theta_{ij,k} \delta_{ijk})$. Here the index $l(ijk)$ has the same meaning as in (3), matrix $\Theta$ represents numerical approximation of convection–diffusion operator and vector $\gamma$ - approximation of $u$ component of velocity on the computational grid. The monotonic approximation of convection–diffusion operator implies that the matrix $\Theta$ is $M$-matrix therefore its diagonal values (and therefore values of $\delta$ in Eq. (22)) are always positive.

Approximations for $\beta = 2, 3$ are analogous. Substituting $u_{ijk}^{n+1},$ $v_{ijk+1/2}^{n+1}, w_{ijk+1/2}$ from those approximations together with the relationship: $\rho_{ijk}^{n+1} = \rho_{ijk}^{n+1}$ in (21) yields:

$$
\begin{align*}
\sum_{l=1}^{L_{ijk}} \gamma_l \rho_{ijk}^{n+1} &= \frac{\tau}{h_x} \rho_{ijk} ^{n+1} \frac{\partial u_{ijk}^{n+1}}{\partial x} - \frac{\tau}{h_y} \rho_{ijk} ^{n+1} \frac{\partial u_{ijk}^{n+1}}{\partial y} - \frac{\tau}{h_z} \rho_{ijk} ^{n+1} \frac{\partial u_{ijk}^{n+1}}{\partial z} + \gamma_1 \rho_{ijk}^{n+1} F_{u,ijk} \\
&= -A_x (\alpha_x' \rho u_{ijk}^{n+1})_{ijk+1/2} + A_x (\alpha_x' \rho u_{ijk}^{n+1})_{ijk-1/2} - A_y (\alpha_y' \rho v_{ijk}^{n+1})_{ijk+1/2} \\
&- A_y (\alpha_y' \rho v_{ijk}^{n+1})_{ijk-1/2} - A_z (\alpha_z' \rho w_{ijk}^{n+1})_{ijk+1/2} + A_z (\alpha_z' \rho w_{ijk}^{n+1})_{ijk-1/2} = \Phi_{l(ijk)},
\end{align*}
$$

where $\alpha_x' = 1/(1 + \tau_1 \delta_{ijk}), \alpha_y' = 1/(1 + \tau_1 \delta_{ijk}), \alpha_z' = 1/(1 + \tau_1 \delta_{ijk})$ are nondimensional and positive since the values of $\delta$ are positive as described above.

Consider now the boundary conditions at the inlet boundary. Let the inlet boundary coincide with the subset of nodes of one of the velocity components: $\{i = 1/2, 1 \leq j \leq N_y, 1 \leq k \leq N_z\}$. At the inlet boundary constant values of velocity components, temperature and water substance mass fraction, together with zero pressure gradients normal to the boundary are assumed. Substituting those conditions to the continuity equation in the nodes of the nearest plane to the inlet boundary (1jk) will lead to the following modification of Eq. (23):

$$
\begin{align*}
\frac{\tau}{h_x} \rho_{ijk}^{n+1} &= \frac{\tau}{h_x} \rho_{ijk}^{n+1} \frac{\partial u_{ijk}^{n+1}}{\partial x} - \frac{\tau}{h_y} \rho_{ijk}^{n+1} \frac{\partial u_{ijk}^{n+1}}{\partial y} - \frac{\tau}{h_z} \rho_{ijk}^{n+1} \frac{\partial u_{ijk}^{n+1}}{\partial z} + \gamma_1 \rho_{ijk}^{n+1} F_{u,ijk} \\
&= \Phi_{l(ijk)}
\end{align*}
$$

Let the outlet boundary coincide with the subset of nodes of the scalar variables: $\{i = N_y, 1 \leq j \leq N_y, 1 \leq k \leq N_z\}$. The
boundary conditions at the outlet boundary are of the Neumann type: $\frac{\partial \rho}{\partial n} = 0$, where $\varphi = \{ \rho, u_g, c, q_{av} \}$, $\mathbf{n}$ is normal vector to the boundary. Substituting these conditions to the numerical approximation of the continuity Eq. (16) in the nodes of the outlet boundary $(N, (j, k))$ obtain the following modification of Eq. (23):

$$
\frac{\xi}{\tau} P_{Nj,k}^{t+1} - \rho_{Nj,k}^{t+1} - \tau A_1 \alpha_c^l A_2 P_{Nj,k}^{t+1} - \tau A_1 \alpha_c^l A_2 D_{Nj,k}^{t+1} = \Phi_{Nj,k}^{t+1}
$$

As follows from Eq. (25), values of pressure at the outlet boundary are not linked (during one iteration step) to the values of pressure in interior domain. This feature makes boundary conditions (25) in a certain sense similar to the constant pressure boundary condition, which is typically applied at the outlet boundaries for modeling of subsonic compressible flows (Wesseling, 2001, Section 12.4; Chung, 2002, Table 13.6.1). However, in contrast to that more frequently used approach, in case of (25) constant pressure is applied only in one specified node of the outlet boundary plane. This allows preserving full mass conservation of the flow in the computational domain without additional corrections of the outlet velocity field which are to be applied if constant pressure were used as boundary condition for the whole outlet boundary plane (Eq. (9.6) from Versteeg and Malalasekera, 1995).

Thus, the overall flow of the ADREA/SIMPLER algorithm, based on the SIMPLER algorithm of Patankar (1980), consists of two steps per iteration step: (a) solve Eq. (23) using the values from the previous iteration step; (b) solve Eqs. (17)–(19) using the corrected pressure values, and update all other variables, including the variable $\xi^t$ needed for next iteration of the algorithm.

Note, that the derivation above was performed for the regular cells. The same kind of derivation could be performed also for the irregular cells. However, in that case the operators $A_1, A_2$ and other should be treated as generalized operators, approximating the corresponding derivatives, which follow from the finite volume discretization of the original equations.

References

tion and transport of pollutants over a Mediterranean coastal area: the in-
fluence of biogenic volatile organic compound emissions on ozone con-
Axelsson, O., Lu, H., 1997. A survey of some estimates of eigenvalues and con-
dition numbers for certain preconditioned matrices. Journal of Computa-
tional and Applied Mathematics 80, 241–264.
ological modelling for nuclear emergency preparedness. Journal of Environ-
mental Radioactivity 85 (2–3), 154–170.
Bartzis, J.G., Venetansos, A.G., Varvayanni, M., Catsaros, N., Megaritus, A.,
1991. ADREA-I. A three-dimensional transient transport code for complex
Bartzis, J.G., Catsaros, N., Varvayanni, M., Venetansos, A., Andronopoulos, S.,
Volumes, Mesoscale Meteorological Prognostic Model. Report RO-
DOS(WG2)-TN(99)-02. Available from: <www.rodos.fzk.de>
Botta, E.F.F., Wubs, F.W., 1999. Matrix renumbering ILU: an effective alge-
braic multilevel ILU preconditioner for sparse matrices. SIAM Journal
on Matrix Analysis Applications 20 (4), 1007–1026.
ILU preconditioner with high level fill. BIT Numerical Mathematics 32,
442–463.
Flasch, Th., Moussiopoulos, N., 1988. Direct solution of the Helmholtz equa-
tion using Fourier analysis on the CYBER 205. Environmental Software 3
(1), 12–16.
Grell, G.A., Dudhia, J., Stauffer, D.R., 1994. A Description of the Fifth-gen-
eration Penn State/NCAR Mesoscale Model (MMS), NCAR Technical
Gustaffson, I., 1996. An incomplete factorisation preconditioning method
based on modification of element matrices. BIT Numerical Mathematics
36 (1), 86–100.
precipitation using Cartesian coordinates and a single prognostic equation
José, R.S., Pérez, J.L., González, R.M., 2007. An operational real-time air
quality modelling system for industrial plants. Environmental Modelling
Ortega, J.M., Rheinboldt, W.C., 1970. Iterative Solution of Nonlinear Equa-
Publishing Corporation.
large-scale free surface flows. Environmental Modelling and Software 21
(1), 98–106.
Raskob, W., Ehhardt, J. (Eds.), 1999. The RODOS System: Decision Support
For Nuclear Off-site Emergency Management in Europe. RODOS report
consistent models for wind field over complex terrain: the state of the
art. Environmental Software 9, 247–268.
Saad, Y., 2003. Iterative Methods for Sparse Linear Systems. SIAM,
Philadelphia.
Sotiropoulou, R.P., Tagaris, E., Pilinis, C., Andronopoulos, S., Sfetsos, A.,
in Athens and Marseille greater areas. Journal of Geophysical Research
109 (DS5), D05205.1–D05205.16.
Thomas, S.J., Browning, G.L., 2001. The accuracy and efficiency of semi-implic-
tite time-stepping for mesoscale storm dynamics. Journal of the Atmo-
spheric Sciences 58, 3053–3063.
for non-hydrostatic atmospheric models. Monthly Weather Review 131,
2464–2478.
Sim-
ulation of the sea breeze under opposing synoptic conditions. Environmen-
tal Software 8 (1), 19–27.
Varvayanni, M., Catsaros, N., Konte, P., Statharas, J., Bartzis, J.G., 1998. De-
velopment and interaction of thermally driven flows over Attiki peninsula
under northerly background wind — a case study. Atmospheric Environ-
ment 32 (12), 2291–2311.
Cambridge University Press.
Versteeg, H.K., Malalasekera, W., 1995. An Introduction to Computational

