Optimisation 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 & Systems Problems, National Academy of Sciences of Ukraine, Department of Environmental Modelling, pr. Glushkova-42, 03187, Kiev, Ukraine

[b] National Centre for Scientific Research (NCSR) "Demokritos", Institute of Nuclear Technology and Radiation Protection, Environmental Research Laboratory, 15310 Aghia Paraskevi Attikis, Greece

[c] University of Western Macedonia, Department of Engineering and Management of Energy Resources , Bakola and Sialvera Str., 50100 Kozani, Greece

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 of ILUT preconditioning with red-black ordering of the unknowns (RB+ILUT) has resulted in up to 6 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 lead to decrease of the statistical errors indicators by only 1-5%.

Keywords: preconditioning, ordering, BiCGSTAB, SIMPLER, sea breeze, time step, Courant number

published in revised form in *Environmental Modelling and Software*, 2008 (23), 96-108

[*] Corresponding author.
E-mail address: ik@env. kiev.ua, Fax +380-44-5263615

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 in 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 those 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 micro- scale 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 a 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, Varvayanni, et al., 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 involve 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, complex topography). The terms that account for those effects in the hydrodynamic equations impose different restrictions on the time steps τ 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: $\tau < \tau_{\max} \approx \min(h_x, h_y, h_z)/c_s$ (Roache, 1972), where h_x, h_y, h_z, are mesh sizes in the corresponding directions $c_s \approx 300\, m/s$ is the sound speed. If the sound and gravity waves are filtered by the Boussinesq and hydrostatic approximations the Courant number restriction remains: $\tau < \tau_{\max} = \min_G \left(\sum_\alpha h_\alpha / |u_\alpha| \right)$, where G is the computational domain, u_α is the velocity component in α-th direction. The latter is much less restrictive than in the compressible case, since in the atmosphere the Mach number always: $M = U/c_s << 1$ (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 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): $\tau < 0.003 \min\left(h_x, h_y\right) \approx \min\left(h_x, h_y\right)/c_s$. 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 (Flassak and Moussiopoulos, 1988) are traditional for the meteorological applications. Still each of those 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 (Flassak 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. Those 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. Discretisation 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 equation (23)):

$$\frac{\xi_{ijk}^{s} P_{ijk}^{s+1} - \rho_{ijk}^{n}}{\tau} - \tau \Lambda_x \alpha_x^s \Lambda_{\bar{x}} P_{ijk}^{s+1} - \tau \Lambda_y \alpha_y^s \Lambda_{\bar{y}} P_{ijk}^{s+1} - \tau \Lambda_z \alpha_z^s \Lambda_{\bar{z}} P_{ijk}^{s+1} = \Phi_{ijk} \qquad (1)$$

where n is the time level, s is the iteration number, P is the pressure, $P^s = P^{n+1,s}$, ρ is the density, $\alpha_x, \alpha_y, \alpha_z$ are positive nondimensional coefficients depending on velocities, $\Lambda_x, \Lambda_{\bar{x}}$ are operators approximating corresponding spatial derivatives by the forward and backward differences (see Appendix), $\xi^s = \rho^s / P^s$,and Φ 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 equation (1) result from the numerical approximation of the governing system of model equations - continuity, momentum, internal energy and water mass fraction, which originally does not include the pressure equation. The boundary conditions (BC) for equation (1), are derived from the BC for density and velocities to assure mass and momentum conservation in boundary cells (see Appendix). For

instance, if $i=1$ is the plane nearest to the inflow boundary, where Dirichlet (constant value) BC are imposed for velocity components, temperature and water substance mass fraction, then, the pressure equation will be:

$$\frac{\xi^s_{1jk}}{\tau}P^{s+1}_{1jk}-\tau\Lambda_y\alpha^s_y\Lambda_{\bar{y}}P^{s+1}_{1jk}-\tau\Lambda_z\alpha^s_z\Lambda_{\bar{z}}P^{s+1}_{1jk}-\frac{\tau}{h_x}\left[\alpha^s_x\Lambda_x P^{s+1}_{1jk}\right]=\frac{1}{\tau}\rho^n_{1jk}+\Phi_{1jk} \tag{2}$$

Analogous BC appear at the other boundary planes and corners. In BC (2) the term in the square brackets is zero at the outflow boundary planes where the Neumann BC are imposed ($\partial\varphi/\partial\mathbf{n}=0$, φ - variable, $\mathbf{n}$ normal vector to the boundary).

The problem (1)-(2) leads 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:

$$l=N_xN_z(j-1)+N_z(i-1)+k=ORD_0\left(i,j,k\right). \tag{3}$$

Here j,i,k – are the indices in y,x,z directions correspondingly, N_x,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 the equation (1) leads to the following system of the algebraic equations in interior domain:

$$\begin{aligned}&\left(\underline{\underline{A_0}}\,\underline{x}\right)_l=\left(\underline{b_0}\right)_l=\\&AP(l)x(l)+AU(l)x(l+1)+AD(l)x(l-1)+\\&AE(l)x(l+N_z)+AW(l)x(l-N_z)+\\&+AN(l)x(l+N_xN_z)+AS(l)x(l-N_zN_z)\end{aligned} \tag{4}$$

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 $\underline{\underline{A}}_0$ is seven diagonal with *AU*, *AE*, *AN* representing its upper part and *AD*, *AW*, *AS* – its lower part. The following conditions hold in the domain interior:

$$\begin{aligned}&AU(l)=AD(l+1)\\&AE(l)=AW(l+N_z)\\&AN(l)=AS(l+N_zN_x)\end{aligned} \tag{5}$$

which follow from the conservative form of the equations (1)-(2) and imply, that the matrix $\underline{\underline{A}}_0$ is symmetric. However, asymmetry can be introduced by the boundary conditions. For instance, the following hold near the boundaries:

$$\begin{aligned}&AW\left(l_0\right)=0,\ l_0=l\left(N_x,j,k\right)\\&AE\left(l_0-N_k\right)\neq 0\end{aligned} \tag{6}$$

when the term in square brackets of equation (2) is equal to zero. Prior to numerical solution the equations (4) are normalised, so that $AP(l) = 1, \forall l$.

For equation (1) the following relationship holds:

$$\begin{aligned} &0 < -\left(AU(l) + AD(l) + AE(l) + AW(l) + AN(l) + AS(l) \right) < AP(l), \\ &AU(l) \le 0, AD(l) \le 0, AE(l) \le 0, AW(l) \le 0, AN(l) \le 0, AS(l) \le 0, \forall l \end{aligned} . \tag{7}$$

Following Theorem 2.4.14 from Orthega and Rheinboldt, (1970) inequalities (7) guarantee that the matrix $\underline{\underline{A}}_0$ 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:

$$\begin{aligned} &nout = 0 \\ &do(j,i,k) \\ &\begin{cases} l = ORD_0(i,j,k) - nout, & (i,j,k) \notin blocked \\ nout = nout + 1; & (i,j,k) \in blocked \end{cases} \\ &enddo \end{aligned} \tag{8}$$

The matrix $\underline{\underline{A}}$ of the modified system $\underline{\underline{A}}\underline{x} = \underline{b}$ has not any more 7-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 over-relaxation. 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:

$$\left(\underline{\underline{L}} + \underline{\underline{D}} + \underline{\underline{U}}_1 \right) \underline{x}^{s+1} = -\underline{\underline{U}}_2 \underline{x}^s + \underline{b} \tag{9}$$

Here matrix $\underline{\underline{L}}$ is the lower triangular part of the matrix $\underline{\underline{A}}$, matrix $\underline{\underline{D}}$ is the diagonal part of $\underline{\underline{A}}$, $\underline{\underline{U}}_1$ is a matrix that includes the first upper diagonal of $\underline{\underline{A}}$: $U_1(l, l+1) = AU(l)$. Matrix $\underline{\underline{U}}_2$ contains all other upper diagonals of $\underline{\underline{A}}$. Implementation of the iteration step (9) is easily transformed to the solution of $N_x N_y$ systems of equations with three-diagonal matrix of the size N_z.

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:

$$\frac{AE(l)}{AU(l)} \sim \frac{AS(l)}{AU(l)} \sim \frac{\alpha_x S_{xz} h_z}{\alpha_z h_x S_{xy}} \sim \frac{\alpha_x h_z^2}{\alpha_z h_x^2} = R \tag{10}$$

Here S_{xz} is the xz cell surface area, h_z, h_x are the vertical and horizontal mesh sizes. When $h_z / h_x << 1$, then $R<<1$ and the LR method will obviously converge fast.

However, the LR method looses its efficiency when the time integration step τ increases. An example of sharp increase in computational time needed for solution of the pressure equation with increasing time step is shown in the Figure 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 τ 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 τ ($\lambda_{\max} > 0, \lambda_{\min} > 0$ are maximum and minimum eigenvalues of the system). For the rest of the variables (humidity, temperature, turbulent kinetic energy) the solution time raises much slower reaching at most 1 s. Due to such rapid increase in computational time with increasing τ very small time steps are to be used when LR method is applied. For instance, in the test case presented below $\tau = 16s$ 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 $\approx 5 \cdot 10^4$. And even in that case, as follows from the Figure 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 Bi-CGSTAB 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 $\underline{\underline{K}} \approx \underline{\underline{A}}$ is to be solved twice (preconditioning step). The matrix $\underline{\underline{K}}$ 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) factorisation method (van der Vorst, 2003). The ILU preconditioner is defined as the approximation $\underline{\underline{K}} = \underline{\underline{L}}\,\underline{\underline{U}}$ of the matrix $\underline{\underline{A}}$, with the upper and bottom triangular matrices $\underline{\underline{U}}$ and $\underline{\underline{L}}$. Matrix $\underline{\underline{L}}$ has the unit diagonal, and in both matrices $\underline{\underline{U}}$ and $\underline{\underline{L}}$, nonzero elements appear only in positions defined by the nonzero stencil:

$$\begin{aligned} &\underline{\underline{K}} = \underline{\underline{L}}\,\underline{\underline{U}}, \\ &l(i,j) \neq 0, u(i,j) \neq 0 \Leftrightarrow, i,j \in Stenc \end{aligned} \tag{11}$$

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 $O\left(N^3\right)$ 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 $\underline{\underline{A}}$. 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_{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 then N_{lev}. The preconditioners, in which the nonzero stencil was defined by the "level of fill in" were called $ILU(N_{lev})$.

With the given nonzero stencil the construction of the $\underline{\underline{L}}$ and $\underline{\underline{U}}$ 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. This algorithm proceeds as follows:

```
1.      For i=2,N Do:
2.          For k=1,i-1, and if (i,k) ∈ Stenc, Do:
3.              a_ik=a_ik/a_kk
4.                  For j=k+1,N and if (i,j) ∈ Stenc, Do:
5.                  a_ij=a_ij-a_ik a_kj                                    (12)
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 2 cycles are cycles by columns (i.e., in horizontal from the left to 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_{lev}, when $ILU(N_{lev})$ is constructed, or to any other nonzero stencil.

In the Modified ILU factorization $MILU(N_{lev})$ (Gustaffson, 1996), when current row is processed in the algorithm (12) the elements of the $\underline{\underline{L}}$ and $\underline{\underline{U}}$ factors, are first calculated for the stencil of the $N_{lev}+1$ level. Then the elements in positions, not coinciding with the stencil of the N_{lev}-th level are summarized, subtracted from the main diagonal and then set to zero. In this way the constructed $\underline{\underline{L}}$ and $\underline{\underline{U}}$ factors have the following property: $\underline{\underline{A}}\underline{e} = \underline{\underline{L}}\underline{\underline{U}}\underline{e}$, where $\underline{e}$ is unit vector (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 Gustafsson (1996). The rate of convergence of ILU preconditioner for such problems is: $Noper \sim O\left(N^{1.5}\right)$ where $Noper$ is number of operations needed for convergence. For the MILU the convergence is faster: $Noper \sim O\left(N^{1.25}\right)$, (Gustaffson, 1996). That estimation is close to the "grid-independent" convergence: $Noper \sim O(N)$. The MILU factorization is stable for the diagonally dominant M-matrices (Brand, 1996), 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 the 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: $\left|a_{ij}\right| \le \omega \left\|\underline{a}_i\right\| = \omega \left|a_{ii}\right|$. Here ω 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 ω. In all studied cases the performance with ILUT reduced not more then by the factor of 2 within the range of ω: $0.001 \le \omega \le 0.1$.

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:

$$\begin{pmatrix} \underline{\underline{A}}_{11} & \underline{\underline{A}}_{12} \\ \underline{\underline{A}}_{21} & \underline{\underline{A}}_{22} \end{pmatrix} \begin{pmatrix} \underline{x}_R \\ \underline{x}_B \end{pmatrix} = \begin{pmatrix} \underline{b}_R \\ \underline{b}_B \end{pmatrix} \tag{13}$$

If red-black ordering is exact $\underline{\underline{A}}_{11} = \mathbf{I}$, $\underline{\underline{A}}_{22} = \mathbf{I}$, ($\mathbf{I}$ 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:

$$\underline{\underline{S}}\underline{x}_B = \left(\mathbf{I} - \underline{\underline{A}}_{21}\underline{\underline{A}}_{12}\right)\underline{x}_B = \left(\underline{b}_B - \underline{\underline{A}}_{21}\underline{b}_R\right) = \underline{b}_S \tag{14}$$

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

$$\begin{aligned} &Compute\ \underline{v}^1 = \underline{\underline{A}}_{12}\underline{v} \\ &Compute\ \underline{w} = \underline{v} - \underline{\underline{A}}_{21}\underline{v}^1 \end{aligned} \tag{15}$$

The ILU(0) preconditioner of the matrix $\underline{\underline{S}}$ 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. 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 $\underline{\underline{A}}_{11}$ 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 3 levels were used. The obtained approximate block factorization of the original matrix is then used as preconditioner.

The described BiCGSTAB algorithm, together with preconditioners ($ILU(N_{lev})$, $MILU(N_{lev})$, $ILUT$, $ILUM$), and the different orderings (natural JIK, KJI, 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 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 Figure 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 the z-axis vertically upwards. Four sets of calculations were performed, which are summarized in Table 1. In the cases 1, 3, 4 with a horizontal grid resolution of 4×4 km^2 the computational domain was discretized by 46 × 46 × 29 cells in x, y, and z directions respectively. In the case 2 with a finer (2 × 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 initialise the sea-surface and land-surface temperatures and the soil humidity. In the 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-meter wind speed. Thus, the winds in the surface-layer were increased by the factor of two. On the contrary, in the case 4 the initial profile of the magnitude of the wind velocity was reduced by the constant value $\delta U = -0.5 U_{10} = -1.9 m/s$, decreasing the winds in the surface-layer by the factor of two. Though in the 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: $\tau = 16s$. The “large” time steps were defined by the Courant limit: $\tau \approx \min_G \left(\sum_\alpha h_\alpha / |u_\alpha| \right)$. The corresponding values of τ for the 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 h local time. The effect of the time step on the performance of the LR and of the different types of preconditioned BiCGSTAB methods is shown on Figure 2. It is obvious from Figure 2 that for small time steps $\tau < 25s$ all methods perform worse than the line relaxation (LR) method. This is because for small time steps line relaxation takes great 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 $\tau = 400s$) 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 Figure 2) performed several times worse than the MILU(1), requiring time for solution of pressure equation about 8 s when $\tau = 100s$. 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 24h 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 the cases 1,3,4 on the coarse grid and by a factor of 3.5-4.5 in the 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 ω .

The total computational times needed by ADREA-I for the 24h 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 the levels of improvement by a factor of 5-6 were achieved in the cases 1, 3 and 4 on the coarse grid and by a factor of 3.5-4.5 in the case 2 on the fine grid.

In case 2 increasing the spatial resolution by a factor of 2 in comparison with case 1 lead 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: $Tcalc_2 \approx 7.5 Tcalc_1$. Thus the number

of operations needed to solve the pressure equation (1) per one time step increased by a factor of $Noper_2 / Noper_1 \approx (Tcalc_2 \tau_2) / (Tcalc_1 \tau_1) \approx 3.8 \approx N_2 / N_1$. 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 calculations results (in the cases 1 and 2) with the meteorological measurements for the specific day. The comparisons were performed against the data from the 8 meteorological stations, which locations are shown in Figure 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, temperatures) for three stations are shown in Figures 3-4. Calculations with the small time steps are shown with the solid lines, while the calculations with the time steps, defined by the Courant limit are shown with the dashed lines. Both observed and predicted time histories of wind demonstrate pronounced sea breeze development. The simulated by the model wind speeds and temperatures are close enough to the measured values. Figures 3-4 also demonstrate improvement of the calculations agreement with the measurements with increasing grid resolution. Figure 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 the Figures 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 3 presents the root mean square differences in velocity magnitude σ_U and temperature σ_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 $\sigma_U \le 0.2\, m/s$, $\sigma_T \le 0.28\, ^\circ C$. 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, 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 lead 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 to 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 of 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 № 516492 (FI6R). Authors thank Nikos Gounaris for making Figure 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 Housiadas et al., (1991), and in 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:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_\alpha}{\partial x_\alpha} = 0 \tag{16}$$

$$\frac{\partial u_\beta}{\partial t} + \frac{1}{\rho}\frac{\partial P}{\partial x_\beta} = F_{u_\beta} = -u_\alpha \frac{\partial u_\beta}{\partial x_\alpha} + \frac{1}{\rho}\frac{\partial}{\partial x_\alpha}\left(\rho K_{m\alpha}\frac{\partial u_\beta}{\partial x_\alpha}\right) - 2e_{\beta\alpha\gamma}\omega_\alpha u_\gamma + \frac{1}{\rho}\frac{\partial \rho q_l (1-q_l) u_{s\beta} u_{s\alpha}}{\partial x_\alpha} + g_\beta \tag{17}$$

$$\frac{\partial e}{\partial t} = -u_\alpha \frac{\partial e}{\partial x_\alpha} - \frac{P}{\rho}\frac{\partial u_\alpha}{\partial x_\alpha} + \frac{1}{\rho}\frac{\partial}{\partial x_\alpha}\left(\rho c_p \frac{K_{m\alpha}}{\sigma_h}\frac{\partial T}{\partial x_\alpha}\right) + \frac{1}{\rho}\frac{\partial \rho q_l (1-q_l) u_{s\alpha}\left(e_g - e_l\right)}{\partial x_\alpha} + \frac{P}{\rho}\frac{\partial q_l u_{s\alpha}}{\partial x_\alpha} \tag{18}$$

$$\frac{\partial q_w}{\partial t} = -u_\alpha \frac{\partial q_w}{\partial x_\alpha} + \frac{1}{\rho}\frac{\partial}{\partial x_\alpha}\left(\rho \frac{K_{m\alpha}}{\sigma_h}\frac{\partial q_w}{\partial x_\alpha}\right) - \frac{1}{\rho}\frac{\partial \rho q_l u_{s\alpha}(1-q_w)}{\partial x_\alpha} \tag{19}$$

$$P = \rho / \xi(e, q_v). \tag{20}$$

Here $x_\alpha = (x, y, z)$ are the coordinates, $u_\alpha = (u, v, w)$, g_α, $u_{s\alpha}$, $K_{m\alpha}$, are the components of wind velocity , gravity acceleration, slip velocity, momentum eddy viscosity in α^{th} direction ($\alpha = 1,2,3$), ρ is the density, e, e_l are the specific moist air internal energy and water substance internal energy, q_w, q_l, q_v 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, $e_{\alpha\beta\gamma}$ is the antisymmetric symbol, ω_β 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, solar radiation (Bartzis et al., 1999).

Note, that despite the equations (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 equations (17)-(19). For the presentation of the ADREA/SIMPLER algorithm the non-conservative form is preferable and therefore used here.

The equations (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 with respect to the scalar grid.

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

$$\frac{\rho_{ijk}^{s+1}-\rho_{ijk}^{n}}{\tau}+\frac{\rho_{i+1/2jk}^{s}u_{i+1/2jk}^{s+1}-\rho_{i-1/2jk}^{s}u_{i-1/2jk}^{s+1}}{h_x}+\frac{\rho_{ij+1/2k}^{s}v_{ij+1/2k}^{s+1}-\rho_{ij-1/2k}^{s}v_{ij-1/2k}^{s+1}}{h_y}+\frac{\rho_{ijk+1/2}^{s}w_{ijk+1/2}^{s+1}-\rho_{ijk-1/2}^{s}w_{ijk-1/2}^{s+1}}{h_z}=$$
$$=\frac{\rho_{ijk}^{s+1}-\rho_{ijk}^{n}}{\tau}+\Lambda_x\left(\rho^s u^{s+1}\right)_{i-1/2jk}+\Lambda_y\left(\rho^s v^{s+1}\right)_{ij-1/2k}+\Lambda_z\left(\rho^s w^{s+1}\right)_{ijk-1/2}=0 \qquad (21)$$

where s is the iteration number, n is the time layer, $\varphi^s \underset{def}{=} \varphi^{s,n+1}$, $\Lambda_x, \Lambda_y, \Lambda_z$ are operators, approximating the derivative by forward difference: $\Lambda_x\varphi_{ijk}=\left(\varphi_{i+1jk}-\varphi_{ijk}\right)/h_x$. The values of the scalar variable between nodes: $\rho_{i\pm1/2jk}$ are obtained by the first order interpolation method. The approximation for the momentum equation (17) is ($\beta=1$):

$$\frac{u_{i-1/2jk}^{s+1}-u_{i-1/2jk}^{n}}{\tau}+\frac{1}{\rho_{i-1/2jk}^{s}}\Lambda_{\bar{x}}P_{ijk}^{s+1}=F_{u,i-1/2jk}=\tilde{F}_{u,i-1/2jk}-\delta_{u,i-1/2jk}^{s}u_{i-1/2jk}^{s+1} \qquad (22)$$

Here $F_{u,i-1/2jk}$ is numerical approximation of the right hand side of equation (17),depending on the values of the state vector at both s and s+1 iteration levels; $\tilde{F}_{u,ijk}$ is part of $F_{u,i-1/2jk}$, containing variables only at s iteration level, $\Lambda_{\bar{x}}$ is operator, approximating the derivative by the backward difference: $\Lambda_{\bar{x}}\varphi_{ijk}=\left(\varphi_{ijk}-\varphi_{i-1jk}\right)/h_x$. The values of $\delta_{u,i-1/2jk}^{s}$ are equal to the diagonal elements of the matrix representing the numerical approximation of the diffusion-convection operator: $\left.\left(u_\alpha\frac{\partial u}{\partial x_\alpha}-\frac{1}{\rho}\frac{\partial}{\partial x_\alpha}\rho K_{m\alpha}\frac{\partial u}{\partial x_\alpha}\right)\right|_{x(i),y(j),z(k)}\approx\left(\underline{\underline{\Theta}}\,\underline{u}\right)_{l(ijk)}$. Here the index $l(ijk)$ has the same meaning as in (3), matrix $\underline{\underline{\Theta}}$ represents numerical approximation of convection-diffusion operator and vector $\underline{u}$ - approximation of u component of velocity on the computational grid. The monotonic approximation of convection-diffusion operator implies that the matrix $\underline{\underline{\Theta}}$ is M-matrix therefore its diagonal values (and therefore values of δ in equation (22)) are always positive. Approximations for $\beta=2,3$ are analogous. Substituting $u_{i-1/2jk}^{s+1}$, $v_{ij-1/2k}^{s+1}$, $w_{ijk-1/2}^{s+1}$ from those approximations together with the relationship: $\rho^{s+1}\approx P^{s+1}\xi^s$ in (21) yields:

$$\frac{\xi^s P_{ijk}^{s+1}-\rho_{ijk}^{n}}{\tau}-\tau\Lambda_x\alpha_x^s\Lambda_{\bar{x}}P_{ijk}^{s+1}-\tau\Lambda_y\alpha_y^s\Lambda_{\bar{y}}P_{ijk}^{s+1}-\tau\Lambda_z\alpha_z^s\Lambda_{\bar{z}}P_{ijk}^{s+1}=$$
$$=-\Lambda_x(\alpha_x^s\rho^s u^s)_{i-1/2jk}-\Lambda_y(\alpha_y^s\rho^s v^s)_{ij-1/2k}-\Lambda_z(\alpha_z^s\rho^s w^s)_{ijk-1/2} \qquad (23)$$
$$-\tau\Lambda_x(\alpha_x^s\rho^s\tilde{F}_u^s)_{i-1/2jk}-\tau\Lambda_y(\alpha_y^s\rho^s\tilde{F}_v^s)_{ij-1/2k}-\tau\Lambda_z(\alpha_z^s\rho^s\tilde{F}_w^s)_{ijk-1/2}=\Phi_{ijk}$$

where $\alpha_x^s=1/\left(1+\tau\delta_{u,ijk}^s\right)$, $\alpha_y^s=1/\left(1+\tau\delta_{v,ijk}^s\right)$, $\alpha_z^s=1/\left(1+\tau\delta_{w,ijk}^s\right)$ are non-dimensional and positive since the values of δ 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: $\{i = 1/2, 1 \le j \le N_y, 1 \le k \le N_z\}$. 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 equation (23):

$$\frac{\xi^s P_{1jk}^{s+1} - \rho_{1jk}^n}{\tau} - \frac{\tau}{h_x}\alpha_x^s \Lambda_x P_{1jk}^{s+1} - \tau\Lambda_y \alpha_y^s \Lambda_{\bar{y}} P_{1jk}^{s+1} - \tau\Lambda_z \alpha_z^s \Lambda_{\bar{z}} P_{1jk}^{s+1} = \Phi_{1jk} \tag{24}$$

Let the outlet boundary coincide with the subset of nodes of the scalar variables: $\{i = N_x, 1 \le j \le N_y, 1 \le k \le N_z\}$. The boundary conditions at the outlet boundary are of the Neumann type: $\partial\varphi/\partial\mathbf{n} = 0$, where $\varphi = \{\rho, u_\beta, e, q_w\}$, $\mathbf{n}$ is normal vector to the boundary. Substituting these conditions to the numerical approximation of the continuity equation (16) in the nodes of the outlet boundary $(N_x jk)$ obtain the following modification of equation (23):

$$\frac{\xi^s P_{N_x jk}^{s+1} - \rho_{N_x jk}^n}{\tau} - \tau\Lambda_y \alpha_y^s \Lambda_{\bar{y}} P_{N_x jk}^{s+1} - \tau\Lambda_z \alpha_z^s \Lambda_{\bar{z}} P_{N_x jk}^{s+1} = \Phi_{N_x jk} \tag{25}$$

As follows from equation (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, sec. 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 the equation (23) using the values from the previous iteration step; b) solve the equations (17)-(19) using the corrected pressure values, and update all other variables, including the variable ξ^s 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 $\Lambda_x, \Lambda_{\bar{x}}$ and other should be treated as generalized operators, approximating the corresponding derivatives, which follow from the finite volume discretization of the original equations.

References

1. Axelsson, O., Lu, H., 1997. A survey of some estimates of eigenvalues and condition numbers for certain preconditioned matrices. Journal of Computational and Applied Mathematics 80 241-264.
2. Andronopoulos, S., Passamichali, A., Gounaris, N., Bartzis, J.G., 2000. Evolution and transport of pollutants over a Mediterranean coastal area: the influence of biogenic volatile organic compound emissions on ozone concentrations. Journal of Applied Meteorology 39 (4) 526-545.
3. Baklanov, A., Sørensen, J.H., Hoe, S.C., Amstrup, B., 2006. Urban meteorological modelling for nuclear emergency preparedness. Journal of Environmental Radioactivity 85 (2-3) 154-170.
4. Bartzis, J.G., Venetsanos, A.G., Varvayanni, M., Catsaros, N., Megaritou, A., 1991. ADREA-I. A three-dimensional transient transport code for complex terrain and other applications. Nuclear Technology 94 (2) 135-148.
5. Bartzis, J.G., Catsaros, N., Varvayanni, M., Venetsanos, A., Andronopoulos, S., Vlachogiannis, D., Konte, K., 1999. ADREA-I: A three-dimensional finite volumes, mesoscale meteorological prognostic model. RODOS Report RODOS(WG2)-TN(99)-02 (available at www.rodos.fzk.de).
6. Housiadas, C., Amanatidis, G.T., Bartzis, J.G., 1991, Prediction of orographic precipitation using Cartesian coordinates and a single prognostic equation for the water substance. Boundary Layer Meteorology 56: 245-260.
7. Botta, E.F.F., Wubs, F.W., 1999. Matrix renumbering ILU: an effective algebraic multilevel ILU preconditioner for sparse matrices. SIAM J. Matrix Anal. Appl. 20 (4) 1007-1026.
8. Brand, C., 1992. An incomplete-factorisation preconditioning using repeated red-balck ordering. Numerical Mathematics 61 433-454.
9. Chung T.J., 2002. Computational Fluid Dynamics. Cambridge University Press, Cambridge.
10. D'Azevedo, E.F., Forsyth, P.A., Tang, W.P., 1992. Towards a cost effective ILU preconditioner with high level fill. BIT 32 442-463.
11. Grell, G. A., Dudhia, J., Stauffer, D. R., 1994. A description of the fifth-generation Penn State/NCAR mesoscale model (MM5). NCAR Technical Note, NCAR/TN-398+STR 117 pp (available at: http://www.mmm.ucar.edu/mm5/mm5-home.html)
12. Gustaffson, I., 1996. An incomplete factorisation preconditioning method based on modification of element matrices. BIT 36 (1) 86-100.
13. José, R.S., Pérez, J.L., González, R.M., 2007. An operational real-time air quality modelling system for industrial plants. Environmental Modelling and Software 22 (3) 297-307.
14. Flassak, Th., Moussiopoulos, N., 1988, Direct solution of the Helmholz equation using Fourier analysis on the CYBER 205. Environmental Software 3 (1) 12-16.
15. Notay, Y., 1999. A multilevel block incomplete factorisation preconditioner. Applied Numerical Mathematics 31 209-225.

16. Orthega J.M., Rheinboldt W. C., 1970. Iterative solution of nonlinear equations in several variables. Academic Press, New York and London.
17. Patankar, S.V., 1980. Numerical Heat Transfer and Fluid Flow. Hemisphere Publishing Corporation.
18. Rao, P., Medina, M.A., 2006. Enhanced TABS-MDS model for simulating large-scale free surface flows. Environmental Modelling and Software 21 (1) 98-106.
19. Raskob, W., Ehrhardt, J. (eds.), 1999. The RODOS System: Decision Support For Nuclear Off-Site Emergency Management In Europe. RODOS report WG_EN_TN99_02, http://www.rodos.fzk.de.
20. Ratto, C.F., Festa, R., Romeo, C., Frumento, O.A., Galluzzi, M., 1994. Mass consistent models for wind field over complex terrain: The state of the art. Environmental Software 9 247-268.
21. Roache, P., 1972. Computational Fluid Dynamics. Hermosa Publishers.
22. Sotiropoulou, R.E.P., Tagaris, E., Pilinis, C., Andronopoulos, S., Sfetsos, A., Bartzis, J.G., 2004. The BOND Project: Biogenic Aerosols and Air-Quality in Athens and Marseille Greater Areas. Journal of Geophysical Research D: Atmospheres. 109 (5) D05205 1-16.
23. Thomas, S.J., Browning, G.L., 2001. The accuracy and efficiency of semi-implicit time-stepping for mesoscale storm dynamics. Journal of the Atmospheric Sciences 58 3053-3063.
24. Thomas, S.J., Hacker, J. P., Smolarkievicz, P.K., 2003. Spectral preconditioners for non-hydrostatic atmospheric models. Monthly Weather Review 131 2464-2478.
25. Saad, Y. 2003. Iterative Methods for Sparse Linear Systems, SIAM, Philadelfia.
26. Varvayanni, M., Bartzis, J.G., Helmis, C. G. and Asimakopoulos, D. N., 1993. Simulation of the sea breeze under opposing synoptic conditions. Environmental Software 8 (1), 19-27.
27. Varvayanni, M., Catsaros, N., Konte, P., Statharas, J., Bartzis, J.G., 1998. Development and interaction of thermally driven flows over Attiki peninsula under northerly boackground wind – a case study. Atmospheric Environment 32 (12) 2291-2311.
28. Van der Vorst, H., 2003. Iterative Krylov Methods for Large Linear Systems. Cambridge University Press.
29. Versteeg H.K., Malalasekera W., An Introduction to Computational Fluid Dynamics. Longman Scientific & Technical, Essex, England.
30. Vlachogiannis, D., Andronopoulos, S., Passamichali, A., Gounaris, N., Bartzis, J.G., 2000. A three-dimensional model study of the impact of AVOC and BVOC emissions on ozone in an urban area of the eastern Spain. Environmental Monitoring and Assessment 65 (1-2) 41-48.
31. Wesseling P., 2001. Principles of Computational Fluid Dynamics. Springer-Verlag, Berlin.
32. Weijer, W., Dijkstra, H., Öksüzoglu, H.A., Wubs, F.W., de Nier, A.C., 2003. A fully-implicit model of the global ocean circulation. Journal of Computational Physics 192 452-470.

33. Zhang, J., 2001. A grid based multilevel incomplete LU factorisation preconditioning technique for general sparse matrices. Applied Mathematics and Computation 124 (1) 95-115.

Figures

Figure 1. Computational domain with topography contours, centred on the Attiki (Greece) region. Positions of surface meteorological stations are marked with dots.

Figure 2. Dependence of computational time for solving the pressure equation per time step on time step magnitude, for the different methods: LR, ILU(0), MILU(1) with natural KJI ordering and ILUT+RB. Computational time for solving the temperature equation with the Gauss-Seidel method is also shown.

Figure 3. Calculated and measured time histories of the wind velocity and wind direction for three meteorological stations. Symbols– measurements; 1), "_____" - $h_x = h_y = 4km,\ \tau = 16s$; 2), "_ _ _ _" - $h_x = h_y = 4km,\ \tau = 200s$; 3), "_____" - $h_x = h_y = 2km,\ \tau = 16s$; 4), "_ _ _ _" - $h_x = h_y = 2km,\ \tau = 100s$.

Figure 4. Calculated and measured time histories of the temperature for three meteorological stations. Symbols– measurements; 1), "_____" - $h_x = h_y = 4km,\ \tau = 16s$; 2), "_ _ _ _" - $h_x = h_y = 4km,\ \tau = 200s$; 3), "_____" - $h_x = h_y = 2km,\ \tau = 16s$; 4), "_ _ _ _" - $h_x = h_y = 2km,\ \tau = 100s$.

Figure 5. Near-ground wind field (overlaid on topography contours) at 13:00h local time calculated with: $h_x = h_y = 4km,\ \tau = 16s$ (left) and $h_x = h_y = 4km,\ \tau = 200s$ (right).

Case No.	Case definition	Horizontal mesh size (m)	Time step τ (s) for MILU(1) and RB+ILUT	Computational time (minutes) of different methods		
				MILU(1)	RB+ILUT	LR (τ =16s)
1	20 June 2005, Attiki	4000	200	38	30	190
2	20 June 2005, Attiki	2000	100	285	220	990
3	Same conditions as case 1 but with increased inlet wind velocities	4000	160	45	35	191
4	Same conditions as case 1 but with reduced inlet wind velocities	4000	250	30	28	188

Table 1. Total computational times of 24 h. forecast with different numerical methods in different cases. Calculations were performed with Pentium-IV, 3GHz.

Case No.	rmsu,(m/s)		rmst, $^{\circ}C$	
	with $\tau = 16s$	with large τ	with $\tau = 16s$	with large τ
1	1.69	1.72	2.95	3.1
2	1.26	1.22	2.89	2.94

Table 2. Statistical comparisons between calculations and measurements; rmsu – root mean square deviation of wind velocity; rmst - root mean square deviation of temperature; Case No. corresponds to the Table1; large values of τ correspond to column 4 of Table 1.

Case No.	σ_U (m/s)	σ_T , $^{\circ}C$
1	0.15	0.24
2	0.09	0.11
3	0.18	0.25
4	0.2	0.28

Table 3. Root mean square differences in prediction of the wind velocity (σ_U) and temperature (σ_T) with the "large" and "small" time steps. Case descriptions and corresponding time steps are given in Table 1.

Figure 1

Click here to download high resolution image

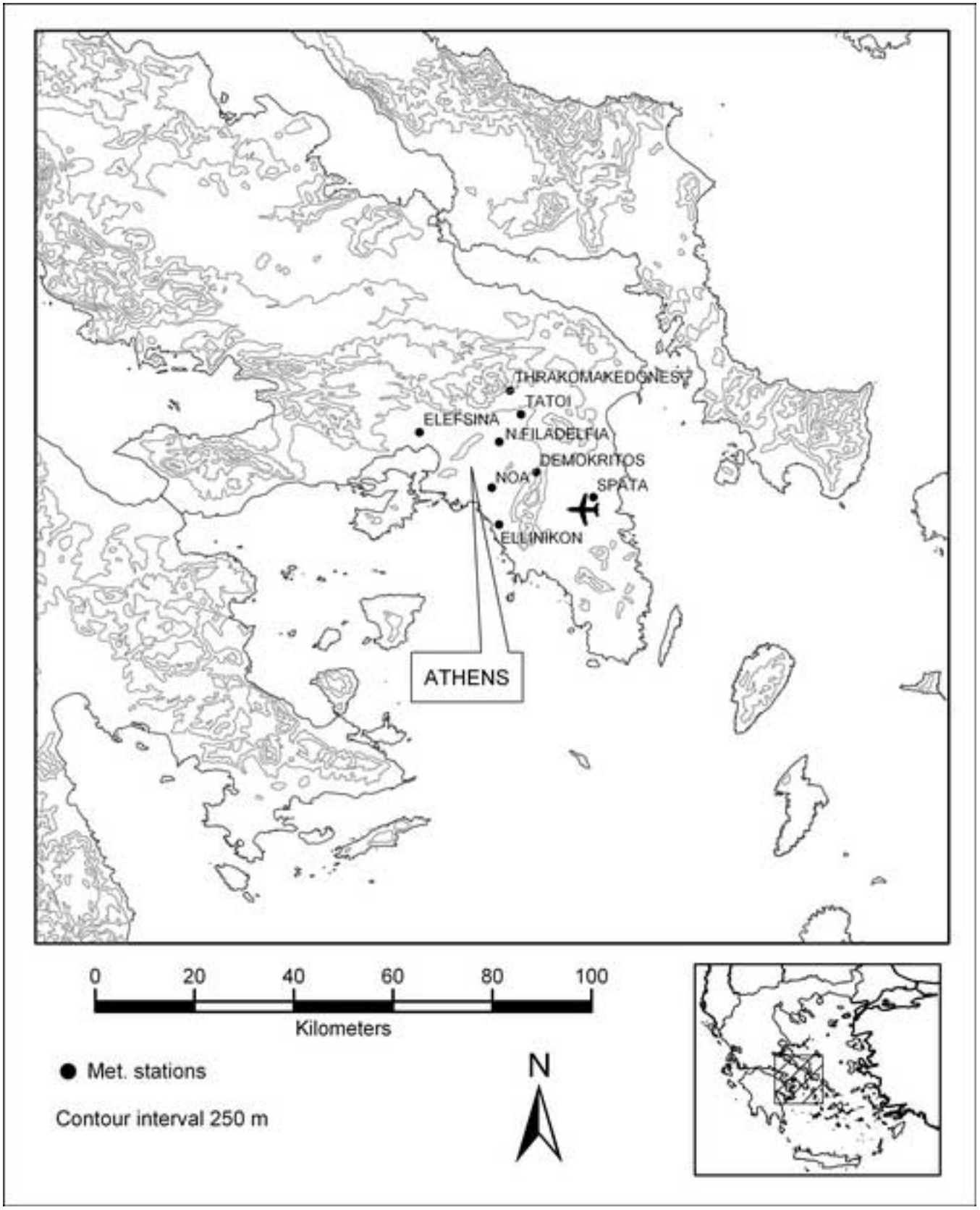


Figure 2

Click here to download high resolution image

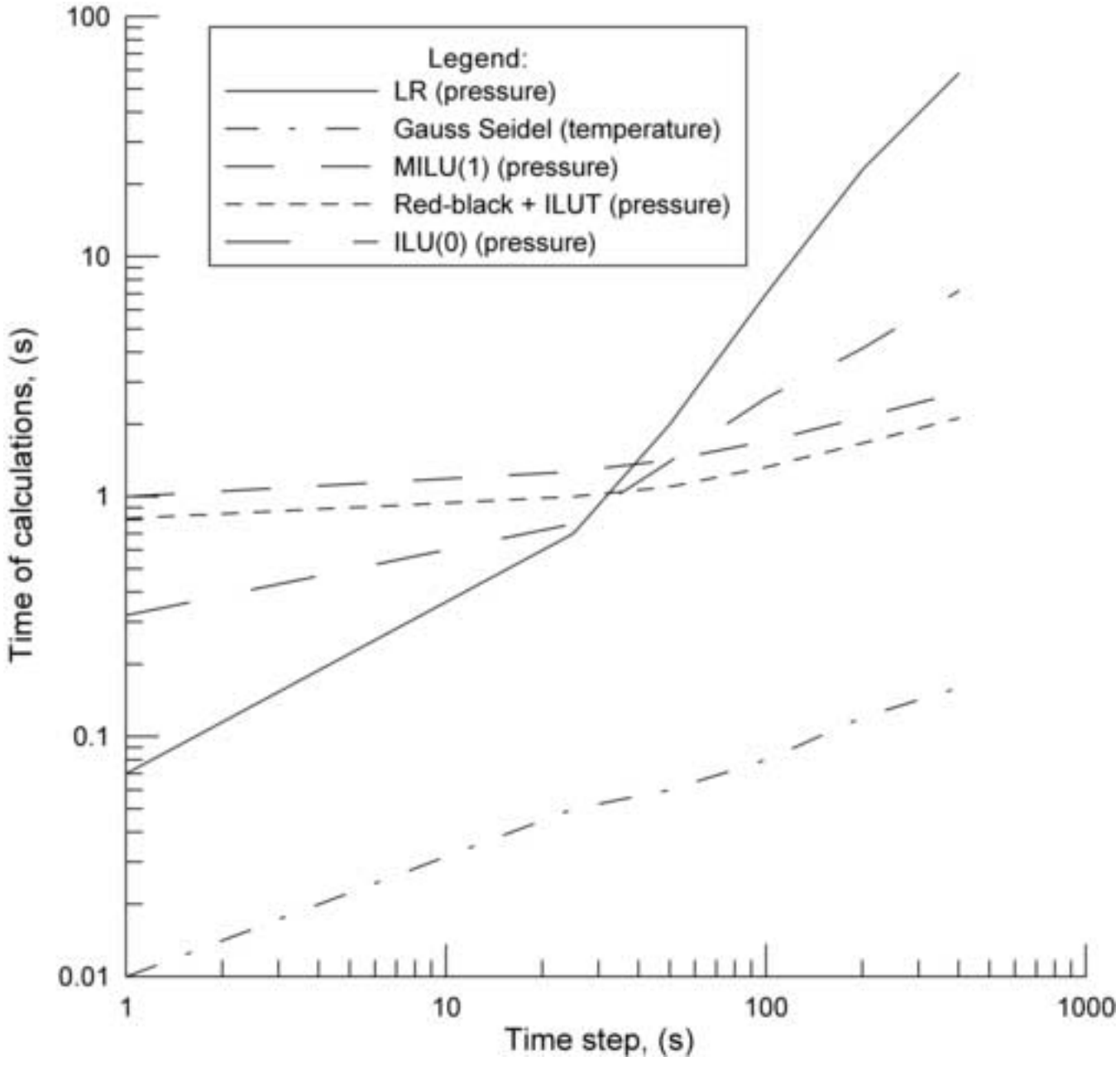


Figure3

Click here to download high resolution image

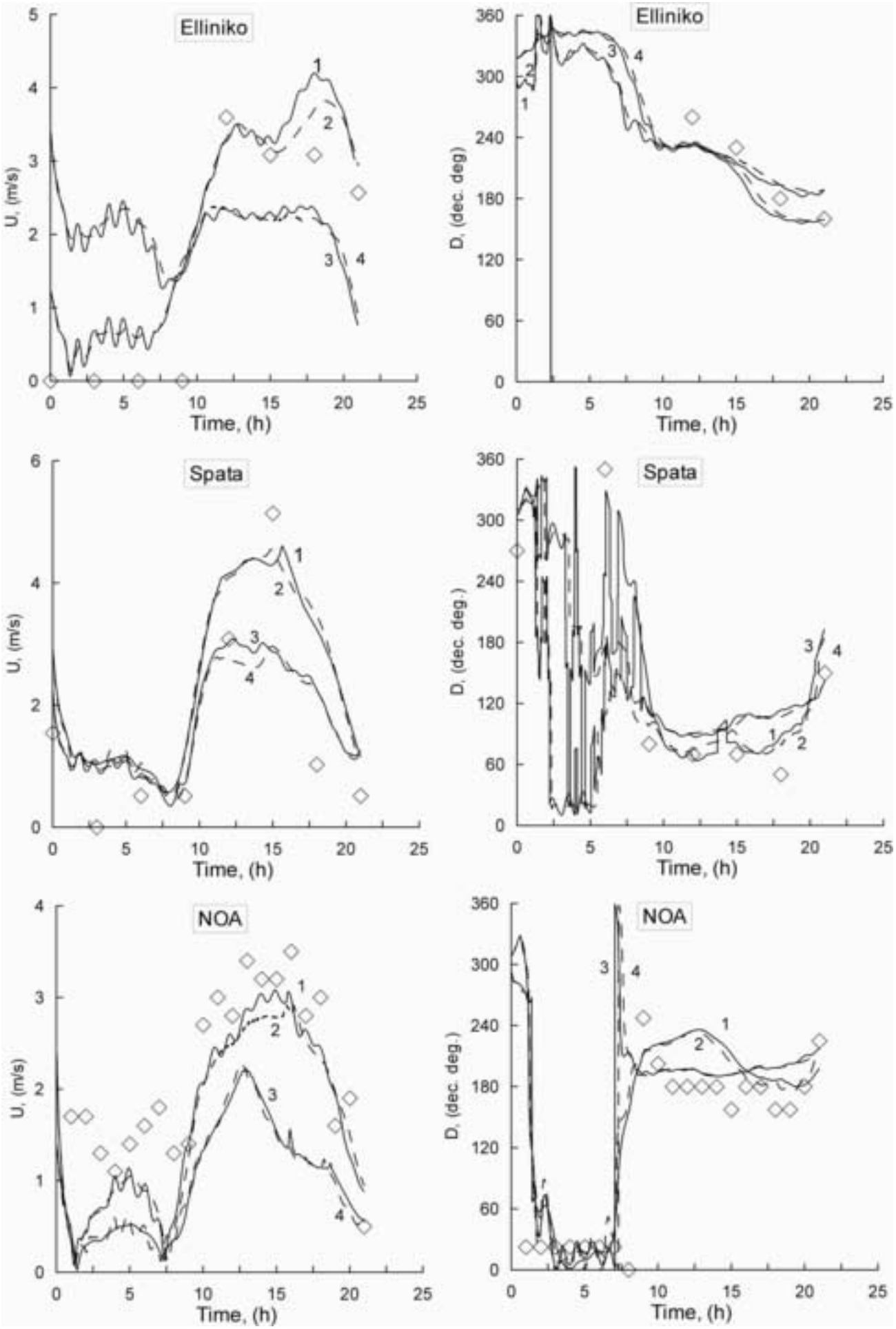


Figure4

Click here to download high resolution image

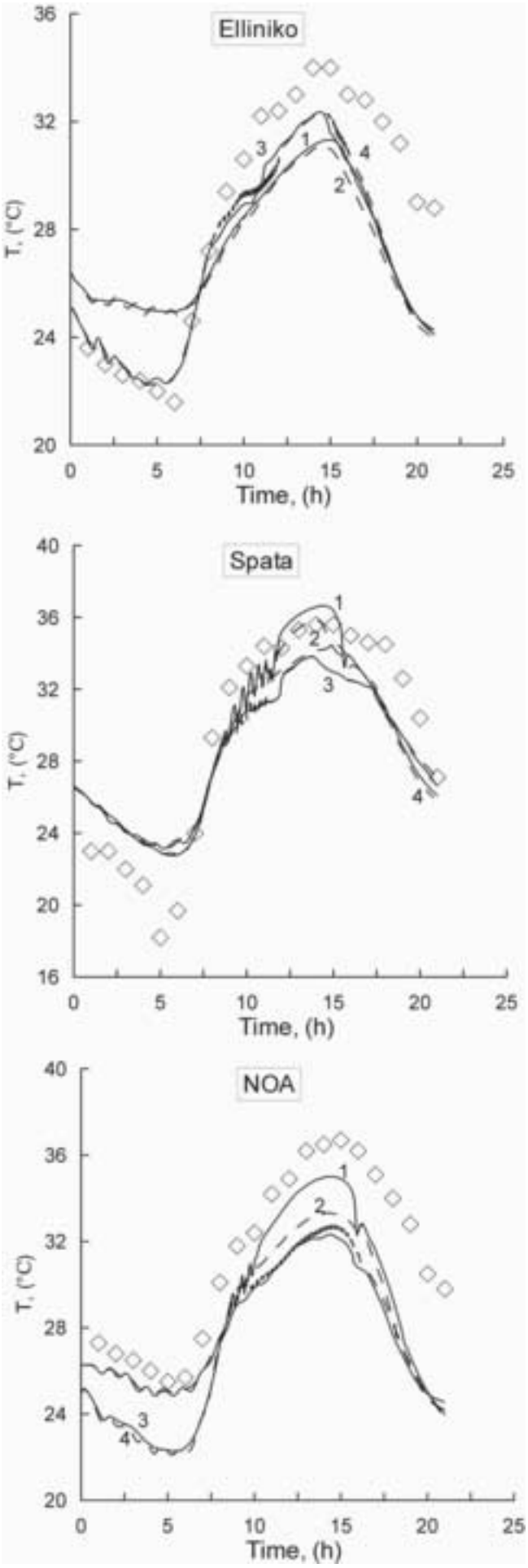
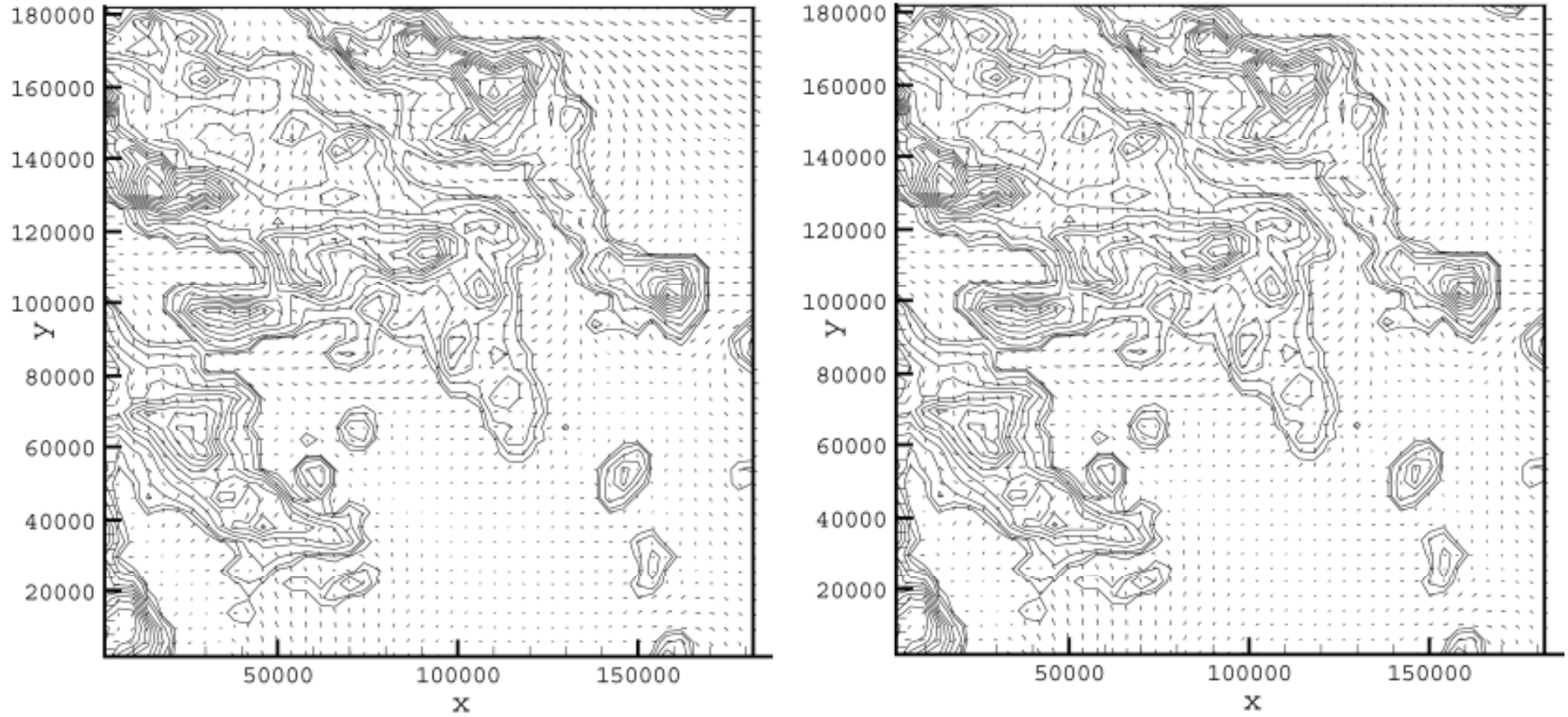


Figure 5


180000
160000
140000
120000
100000
80000
60000
40000
20000
y
50000
100000
150000
x
180000
160000
140000
120000
100000
80000
60000
40000
20000
y
50000
100000
150000
x