COMPUTATIONAL CHALLENGES IN THE NUMERICAL TREATMENT OF LARGE AIR POLLUTION MODELS

I. DIMOV*, K. GEORGIEV†, TZ. OSTROMSKY* , R. J. VAN DER PAS‡, AND Z. ZLATEV§

Abstract.

The air pollution, and especially the reduction of the air pollution to some acceptable levels, is an important environmental problem, which will become even more important in the next 10-20 years. This problem can successfully be studied only when high-resolution comprehensive models are developed and used on a routinely basis. However, such models are very time-consuming, also when modern high-speed computers are available. Indeed, if an air pollution model is to be applied on a large space domain by using fine grids, then its discretization will always lead to huge computational problems. Assume, for example, that the space domain is discretized by using a (480x480) grid and that the number of chemical species studied by the model is 35. Then several systems of ordinary differential equations containing 8064000 equations have to be treated at every time-step (the number of time-steps being typically several thousand). If a three-dimensional version of the same air pollution model is to be used, then the above figure must be multiplied by the number of layers. It is extremely difficult to treat such large computational problems; even when the fastest computers that are available at present are used.

There is an additional great difficulty which is very often underestimated (or even neglected) when large application packages are moved from sequential computers to modern parallel machines. The high-speed computers have normally a very complicated memory architecture and, therefore, the task of producing an efficient code for the particular high-speed computer that is available is both extremely hard and very laborious.

The use of standard parallelization tools in the solution of the problems sketched above is discussed in this paper. Results obtained on different types of parallel computers are given. It is demonstrated that the new efficient parallel algorithms allow us to solve more problems and bigger problems.

Key words. Air pollution modelling, partial differential equations, finite element method, ordinary differential equations, predictor-corrector methods, parallel computations, shared memory computers, distributed memory computers

AMS subject classifications. 65Y05, 65Y20, 65F50

1. Difficulties in the Treatment of Large Air Pollution Models.

High pollution levels (high concentrations and/or high depositions of certain harmful chemical species) may cause damages to plants, animals and humans. Moreover, some ecosystems can also be damaged (or even destroyed) when the pollution levels are very high. This is why the pollution levels must be carefully studied in the efforts to make it possible

- to predict the appearance of high pollution levels, which may cause different damages in our environment *and/or*
- to decide what can be done in order to prevent the exceedance of prescribed critical levels (or, in other words, to attempt to keep the harmful concentrations and/or depositions under the prescribed acceptable limits).

*Central Laboratory for Parallel Processing, Bulgarian Academy of Sciences, Acad. G. Bonchev Str. Bl. 25-A, 1113 Sofia, Bulgaria (ivdimov@bas.bg), (ceco@cantor.bas.bg).

†VITO - TAP Centre for Remote Sensing and Atmospheric Processes, Boeretang 200, B-2400 Mol, Belgium (kris.georgiev@vito.be).

‡HPC Application Performance Specialist, Sun Microsystems (ruud.vanderpas@sun.com).

§National Environmental Research Institute, Frederiksborgvej 399, P. O. Box 358, DK-4000 Roskilde, Denmark (zz@dmu.dk).

The control of the pollution levels in different highly developed and densely populated regions in the world is an important task that has to be handled in a systematic way. This is especially true for many regions in Europe and North America, but also other parts of the world are under quick economic development at present and urgent solutions of certain air pollution problems will soon be necessary also there. The importance of this task has been steadily increasing during the last two decades. The need to develop reliable and easily applicable control strategies for keeping harmful air pollution levels under certain limits will become even more important in the next two-three decades.

Large-scale air pollution models can successfully be used to design reliable control strategies when these models produce reliable results about the pollution levels in the studied region. The application of comprehensive models with different types of sensitivity tests is important in the efforts

- to understand better the physical and chemical processes involved in the air pollution models that are used either in different scientific studies or in the treatment of tasks whose solution is required by policy makers *and*
- to improve as much as possible the reliability of the control strategies that are to be used for keeping the air pollution levels under the prescribed acceptable limits.

The use of comprehensive models in many different environmental studies is a very challenging task. The major difficulties in the treatment of such models are:

- the need to carry out extensive computations,
- the need to store and handle very large input-output files,
- the need to visualize the output data in order to be able to see the trends and relationships hidden behind a great amount of digital data produced by the models
- the need to validate the model results (to show that these are reliable).

We shall concentrate our attention in this paper to discussions of efficient solutions related to the first difficulty. The paper is organized as follows. A short description of the air pollution model which is actually used in this study, the Danish Eulerian Model, is given in Section 2. It should be emphasized, however, that the results reported in this paper can be used also in connections with other large-scale air pollution models. The role of the order in which the computations are carried out when modern advanced computer architectures are used is discussed in Section 3. The parallel computations for different types of computers are described in Section 4. The performance of the code on different parallel computers are presented in Section 5. Some examples for different studies which we are able to carry out with the optimized code are given in Section 6. Some conclusions and plans for future work are discussed in Section 7.

2. Short Description of the Danish Eulerian Model. Large air pollution models are normally described by systems of partial differential equations (PDE's):

$$
\begin{aligned}
(2.1) \qquad \frac{\partial c_s}{\partial t} = & -\frac{\partial (u c_s)}{\partial x} - \frac{\partial (v c_s)}{\partial y} - \frac{\partial (w c_s)}{\partial z} \\
& + \frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x} \right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y} \right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z} \right) \\
& + E_s - (\kappa_{1s} + \kappa_{2s}) c_s + Q_s(c_1, c_2, \ldots, c_q), \quad s = 1, 2, \ldots, q,
\end{aligned}
$$

where (i) the concentrations of the chemical species are denoted by c_s, (ii) u, v and w are wind velocities, (iii) K_x, K_y and K_z are diffusion coefficients, (iv) the emission sources are described by E_s, (v) κ_{1s} and κ_{2s} are deposition coefficients and (vi) the chemical reactions are denoted by $Q_s(c_1, c_2, \ldots, c_q)$ (the CBM IV chemical scheme, which has been proposed in [11], is actually used in the Danish Eulerian Model [32] that is considered in this paper).

2.1. Application of splitting techniques. It is difficult to treat the system of PDE's (2.1) directly. This is the reason for using different kinds of splitting in all known large-scale air pollution models. A splitting procedure, based on ideas proposed in [18] and [19] is used in DEM. It leads, for $s = 1, 2, \ldots, q$, to five sub-models, representing respectively the horizontal advection, the horizontal diffusion, the chemistry (together with the emission terms), the deposition and the vertical exchange:

$$\frac{\partial c_s^{(1)}}{\partial t} = -\frac{\partial (u c_s^{(1)})}{\partial x} - \frac{\partial (v c_s^{(1)})}{\partial y} \tag{2.2}$$

$$\frac{\partial c_s^{(2)}}{\partial t} = \frac{\partial}{\partial x}\left(K_x \frac{\partial c_s^{(2)}}{\partial x} \right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s^{(2)}}{\partial y} \right) \tag{2.3}$$

$$\frac{dc_s^{(3)}}{dt} = E_s + Q_s(c_1^{(3)}, c_2^{(3)}, \ldots, c_q^{(3)}) \tag{2.4}$$

$$\frac{dc_s^{(4)}}{dt} = -(\kappa_{1s} + \kappa_{2s}) c_s^{(4)} \tag{2.5}$$

$$\frac{\partial c_s^{(5)}}{\partial t} = -\frac{\partial (w c_s^{(5)})}{\partial z} + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s^{(5)}}{\partial z} \right) \tag{2.6}$$

2.2. Space discretization of the sub-models. If the model is split into sub-models (2.2) - (2.6), then the discretization of the spatial derivatives in the right-hand-sides of the sub-models leads to the solution (successively at each time-step) of five systems ($i = 1, 2, 3, 4, 5$) of ordinary differential equations (ODE's):

$$\frac{dg^{(i)}}{dt} = f^{(i)}(t, g^{(i)}), \qquad g^{(i)} \in R^{N_x \times N_y \times N_z \times N_s}, \qquad f^{(i)} \in R^{N_x \times N_y \times N_z \times N_s}, \tag{2.7}$$

where N_x, N_y and N_z are the numbers of grid-points along the coordinate axes and $N_s = q$ is the number of chemical species. The functions $f^{(i)}$, $i = 1, 2, 3, 4, 5$, depend on the particular discretization methods used in the numerical treatment of the different sub-models, while the functions $g^{(i)}$, $i = 1, 2, 3, 4, 5$, contain approximations of the concentrations at the grid-points of the space domain (more details are given in [32]).

2.3. Need for fast numerical algorithms and parallel computations. The size of any of the five ODE systems (2.7) is equal to the product of the number of the grid-points and the number of chemical species. It grows very quickly when the number of grid-points and/or the number of chemical species are increased; see Table 2.1. It should be mentioned here that the models descretized at fine grids, (288×288)

TABLE 2.1

Numbers of equations per system of ODE's that are to be treated at every time-step. The typical number of time-steps is 3456 (when meteorological data covering a period of one month + five days to start up the model is to be handled). The number of time-steps for the chemical sub-model is even larger, because smaller step-sizes have to be used in this sub-model.

Number of species	$(32 \times 32 \times 10)$	$(96 \times 96 \times 10)$	$(288 \times 288 \times 10)$	$(480 \times 480 \times 10)$
1	10240	92160	829440	2304000
2	20480	184320	1658880	4608000
10	102400	921600	8394400	23040000
35	358400	3225600	29030400	80640000
56	573440	5160960	46448640	129024000
168	1720320	15482880	139345920	387072000

and (480×480), are at present used only as 2-D models. This means that the number of equations is reduced by a factor of ten. Even in this simplified situation it is very difficult to handle the arising huge computational tasks on the available computers.

Sometimes it is necessary to perform long simulation processes consisting of several hundreds of runs (see, for example, [4] or [34]). At present these problems are solved by the operational two-dimensional version of the Danish Eulerian Model (see [32]). In this version the following values of the parameters are used: $N_x = 96$, $N_y = 96$, $N_z = 1$, $N_s = 35$. This leads to the treatment of four ODE systems per time-step; each of them contains 322560 equations. It is desirable to solve these systems in a more efficient way. It is even more desirable to use the three-dimensional version of the model ([33]) in such runs and/or to implement chemical schemes containing more species (a chemical scheme with $N_s = 35$ is used in this paper). This explains why the search for more efficient numerical methods is continuing (the need of such methods is emphasized, for example, in [26] and [32]). It is also very important to exploit better the great potential power of the modern supercomputers.

3. Ordering the Computations. In the old days it was most important to reduce the number of simple arithmetic operations as much as possible, because the cost of performing an arithmetic operation was much greater than the cost of loading and storing the quantities that are involved in it. In the modern computers the situation is quite different. The cost of loading and storing the quantities needed to perform a given arithmetic operation depend essentially on the place in the memory where the data are located. Practically all modern computers have some cache memory (or even several levels of cache memory), and it is important to work as long as possible with data which are in cache (preferably in the fastest cache when several levels of cache memory are available). The solution of this task is by no means easy. However, if this task is successfully solved, then the computing time can be reduced very considerably. The key issue here is to order the arithmetic operations so that the same data are used as long as possible. It is described below how to order the arithmetic operations in the most time-consuming part of the Danish Eulerian Model, the chemical module, in an attempt to exploit the cache memory in a more efficient way. It should be stressed here that in fact a template which is relatively independent both of the particular numerical method used in the chemical module and of the particular computer is used.

First, it is necessary to show that the chemical module is indeed the most time-consuming part of the computations. This is clearly seen from the results shown in Table 3.1.

In order to reduce the computing time used in the chemical module it is worthwhile

TABLE 3.1

Computing times (measured in seconds) obtained when the first version of the code was run on one processor of the IBM SMP computer. The parts of the computing time spent in the modules (compared with the total time for the run) are given in percent.

Module	Comp. time	Percent
Chemistry	16147	83.09
Advection	3013	15.51
Initialization	2	0.00
Input operations	50	0.26
Output operations	220	1.13
Total time	19432	100.00

to divide the arrays, which are involved in this part of the computational process into chunks and to carry out the computations by chunks. Assume that M is the length of the leading dimension of the two-dimensional arrays used in the chemical module. We want to divide these arrays into $NCHUNKS$ chunks. If M is a multiple of $NCHUNKS$, then the size of every chunks, i.e. the leading dimension of the obtained smaller arrays, is $NSIZE = M/NCHUNKS$, and the following template can be used in the computations.

```
DO ICHUNK=1,NCHUNKS
   Copy chunk ICHUNK from some of the eight
   large arrays into small two-dimensional
   arrays with leading dimension NSIZE
   DO J=1,NSPECIES
      DO I=1,NSIZE
         Perform the chemical reactions involving
         involving species J for grid-point I
      END DO
   END DO
   Copy some of the small two-dimensional
   arrays with leading dimension NSIZE
   into chunk ICHUNK of the corresponding
   large arrays
END DO
```

Some results, which demonstrate the performance of the code when chunks of different sizes are used, are given in Table 3.2. These runs have been performed on four typical computers: a vector processor (Fujitsu), a parallel computer with shared memory (SGI ORIGIN 2000), a 8-processor Macintosh POWER PC cluster with G4 450 MHz processors and a parallel computer, which can be used in both shared memory mode and distributed memory mode (IBM SMP). All runs in Table 3.2 were carried out by using one processor only. It is seen that (i) the particular computer that is available has to be taken into account when the size of chunks is to be selected and (ii) the proper selection of the size of the chunks leads normally to considerable savings in computer time and storage (storage is saved because the leading dimensions of the working arrays in the chemical part (the body of the double loop in the above template) is reduced from M to $NSIZE$; $NSIZE$ is normally considerably smaller than M).

If the size of the chunks is maximal (9216 for the case where a 96x96 grid is used), then the inner loops are very long. It is rather easy to vectorize these loops. Therefore

TABLE 3.2
Computing times (measured in seconds) obtained with different chunks on three computers (using one processor only).

Size of the chunks	Fujitsu	SGI ORIGIN 2000	Power Mac G4	IBM SMP
1	76964	14847	6952	10313
48	2611	12114	5792	5225
9216	494	18549	12893	19432

it should be expected to obtain best results on a vector computer. The results in Table 3.2 show that the maximal length of the chunks is the best choice when the vector processor, Fujitsu, is used. The application of very small chunks, chunks of length 1, corresponds to the straight-forward and commonly used procedure of writing a box routine which performs all chemical reactions in a given grid-point and calling this routine in a loop over all grid-points. It is seen from Table 3.2 that this approach is a disaster when a vector processor is used (because the length of the inner loops is only one).

For the three parallel computers the results when chunks of length 1 are used are not very bad, but still it is more profitable to use chunks of medium size (especially on the IBM SMP computer).

It is clear that the optimal length of the chunks depends on the memory hierarchy the computer. In particular, the cache size at a certain level is an important parameter. Therefore, the length of the chunks, $NSIZE$, should be a parameter which can be selected in the main program. In such a case, it will be relatively easy to find a good value of this parameter by carrying out several tests. It should be mentioned here that our experiments indicate that good results can be achieved for many medium values of $NSIZE$. This is clearly seen from Fig. 1.

4. Preparation for Parallel Computations. It is very important to exploit in the best possible way the great potential power of the modern supercomputers. This is a very difficult task when large-scale air pollution models are to be run, because

- the codes are very big, containing up to several hundreds of subroutines,
- a very large amount of input data (meteorological data and emission data) have to be read and/or interpolated at every time-step and
- a very large amount of output data have to be prepared and stored for future use.

The preparation of efficient versions of the Danish Eulerian model for three types of parallel computer architectures:

- parallel systems with multiple processors, a shared memory architecture and cache coherent interconnect (we shall refer to computers of this type as *parallel computers with shared memory*),
- parallel systems with a single processor per node, a distributed memory and non-cache coherent interconnect (we shall refer to computers of this type as *parallel computers with distributed memory*) and
- a hybrid combination of these, in which multiple shared memory systems are coupled through a non-cache coherent interconnect (we shall refer to computers of this type as *more advanced parallel computers utilizing both shared memory and distributed memory*).

is sketched in this section.

4.1. Running the model on shared memory computers. OpenMP ([21]) directives are used when the code is run on parallel computers with shared memory. The OpenMP directives are becoming standard directives that are supported by many vendors. Therefore, it is easy (i) to get good results on different shared memory computers when such directives are used and (ii) to achieve a high degree of portability.

It is important to identify the parallel tasks and to group them in an appropriate way when necessary. For the different parts of the code this is done in the following way:

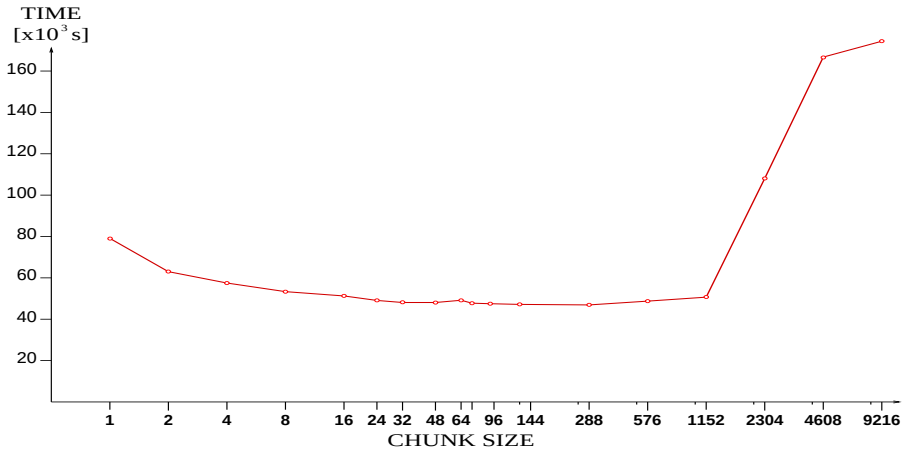

Figure 1

Computing times obtained in running the 3-D version of DEM (discretized on a 96×96 grid) on one processor of the SGI Origin 2000 computer.

- **The horizontal advection and diffusion.** It can easily be seen that, after the splitting procedure, the performance of the horizontal advection can be carried out independently for every chemical compound (and for the 3-D version for every layer). This means that the number of parallel tasks is equal to the number of chemical compounds (and to the product of the chemical compounds and the layers when the 3-D version is used). The same is true for the horizontal diffusion. Moreover, the advection and the diffusion parts can be treated, as already mentioned in the previous section, together. Thus, there are many parallel tasks in this part of the code and, moreover, the parallel tasks are very big.
- **The chemistry and deposition.** These two processes can be carried out in parallel for every grid-point. This means that there are many parallel tasks (the number of parallel tasks is equal to the number of grid-points), but each task is a small task. Therefore, the tasks should be grouped in an appropriate way. This can be done by using chunks. Both the procedure of splitting the data into chunks and the effect of using chunks are discussed in detail in Georgiev and Zlatev [10].
- **The vertical exchange.** The performance of the vertical exchange along each vertical grid-line is a parallel task. The number of these tasks is very large, $N_x \times N_y \times N_s$. If the grid is fine, then the number of these tasks is becoming enormous, see the example given in §4.3. However, the parallel

tasks are not very big and have to be grouped. This is done by trying to distribute equally the tasks among the assigned processors.

4.2. Running the model on distributed memory computers. Either the Message Passing Interface (MPI, [12]) or the Parallel Virtual Machine (PVM, [8]) can be used on parallel computers with distributed memory. We started by using PVM (see Bendtsen and Zlatev [3]), but only MPI has been used in the last four-five years (see Georgiev and Zlatev, [9] and [10]).

In the MPI implementation, the space domain of the model is divided into several sub-domains (the number of these sub-domains being equal to the number of the processors assigned to the job). Then each processor works on its own sub-domain.

Two procedures, a pre-processing procedure and a post-processing procedure, are performed in the beginning and in the end of the run.

- **The pre-processing procedure.** In the beginning of the job the input data (the meteorological data and the emission data) are distributed (consistently with the sub-domains) to the assigned processors. In this way, not only is each processor working on its own sub-domain, but it has also access to all meteorological and emission data which are needed in the run.
- **The post-processing procedure.** During the run, each processor prepares output data files. At the end of the job all these files have to be collected on one of the processors and prepared for using them in the future. This is done by the post-processing procedure.

The use of the pre-processing and post-processing procedures is done in order to reduce as much as possible the communications during the actual computations. However, some communications are to be carried out during the computations. The time needed for these communications is very small (normally, several percent).

Much more details about the runs of several versions of the Danish Eulerian Model on parallel computers with distributed memory by using MPI tools can be found in Georgiev and Zlatev, [10].

4.3. Running the code on some more complicated architectures. More complicated computer architectures are becoming available during the last decade. An example for such an architecture is the IBM SMP computer. In fact, some ideas, on which this architecture is built, have been used under the work on the CEDAR project; see [16]. The IBM SMP consists of several nodes. Every node contains several processors.

In the architecture available for us, there were two IBM SMP nodes, each of them containing eight processors. Each node could be considered as a shared memory computer, while message passing is needed across the nodes.

Some runs on this computer are described here. Again the space domain of the model is divided into sub-domains (one per each node). The pre-processing procedure is used to distribute the data among the nodes, while by the post-processing procedure the data is collected to one of the nodes and prepared for future use. OpenMP directives are to be used on each node in order to obtain parallel computations within the node (across the processors of the node).

Both 2-D versions and 3-D versions of the Danish Eulerian Model were run on this computer. Moreover, 2-D versions discretized on three grids, the (96×96) grid, the (288×288) grid and the (480×480) grid, were tested.

It is important to emphasize that we are using only standard parallelization tools in all these versions of the Danish Eulerian Model; both MPI tools and OpenMP

directives Therefore, it should be easy to port this code to other computers of this type.

More details about the organization of the parallel computations on computers of this type can be found in Owczarz and Zlatev, [22] and [23].

5. Performance of the Code on Different Types of Computers. Various versions of the Danish Eulerian Model have been run on three different computers available at the Danish Computing Centre. Some results obtained in these runs are given in the following tables:

- **Table 5.1** Results obtained with the 3-D version of the Danish Eulerian Model, which is discretized on a $(96 \times 96 \times 10)$ grid, when a shared memory computer is used. The computer actually used was an SGI Origin 2000.
- **Table 5.2** Results obtained with a 2-D version of the Danish Eulerian Model, which is discretized on a fine (480×480) grid, when a distributed memory computer is used. The computer actually used was an IBM SP. It should be emphasized here that the job is so big that it was not possible to run it on less than 8 processors. Therefore, the speed up and the efficiency were calculated by comparing the results obtained when 32 processors are used with the corresponding results obtained when 8 processors are used.
- **Table 5.3** Results obtained with a 2-D version of the Danish Eulerian Model, which is discretized on a (96×96) grid, when an IBM SMP computer (two nodes, eight processors per node) is used.

The results show that good speed-up can be achieved on different computers when standard parallelization tools are applied. Much more results can be found in [9], [10], [22] and [23].

TABLE 5.1
Computing times (measured in seconds) obtained by using OpenMP on the SGI Origin 2000 computer when the 3-D version of the Danish Eulerian Model is discretized on a $(96 \times 96 \times 10)$ *grid.*

Processors	Comp. time	Speed-up	Efficiency
1	42907	-	-
32	2215	19.37	61%

TABLE 5.2
Computing times (measured in seconds) obtained by using MPI on the IBM SP computer when the 2-D version of the Danish Eulerian Model is discretized on a (480×480) *grid.*

Processors	Comp. time	Speed-up	Efficiency
8	54978	-	-
32	15998	3.44	86%

TABLE 5.3
Computing times (measured in seconds) obtained by using the IBM SMP computer (applying OpenMP within each node and MPI across the nodes) when the 2-D version of the Danish Eulerian Model is discretized on a (96×96) *grid.*

Processors	Comp. time	Speed-up	Efficiency
1	5225	-	-
16	424	12.32	72%

5.1. Scalability of the code. It is important *to preserve* the efficiency of the code when the size of some of the involved arrays is increased (for example, as a result of refining the grid, increasing of the number of chemical compounds, the transition from the 2-D version to a 3-D version, etc.). This property is often referred to as a scalability of the code. While such a property is highly desirable (the requirements to the air pollution codes are permanently increasing), it is by no means clear in advance whether the code has such a property or not when the modern complicated computer architectures are used.

Some experiments were performed in an attempt to check the scalability of the parallel systems discussed in the previous sections. A (288×288) grid was considered instead of the (96×96) grid considered in the previous sections. Since the space domain remains unchanged (a $4800\,km \times 4800\,km$ area containing the whole of Europe) this corresponds to a transition from cells of size $(50\ km\ \times\ 50\ km)$ to cells of size $(16.67\ km\ \times\ 16.67\ km)$; see also Table 1.1. This means that in the refined on a (288×288) grid version of the code the number of grid-points was increased by a factor of 9. The number of chemical species was kept 35.

In the advection part, we had also to decrease the time-stepsize by a factor of 3. Thus, the number of arithmetic operations (or, in other words, the amount of computational work) is increased by a factor of 27 in the advection part.

There was no need to decrease the time-stepsize in the chemical part. This means that the number of arithmetic operations (or, in other words, the amount of computational work) is increased by a factor of 9 in the chemical part.

If a 3-D version of the Danish Eulerian Model is used on a $(288 \times 288 \times 10)$ grid, then the computational work in the horizontal advection diffusion part and in the chemical part is increased with the same factors, 27 and 9 respectively, compared with the 3-D version discretized on a $(96 \times 96 \times 10)$ grid; it should be emphasized, however, that the 3-D version discretized on a $(288 \times 288 \times 10)$ grid is not ready yet. Furthermore, there is no need to decrease the time-stepsize in the vertical exchange part either when a version discretized on a $(288 \times 288 \times 10)$ grid is prepared. This means that in this part of code the number of arithmetic operations (or, in other words, the amount of computational work) is increased as in the chemical part (i.e. by a factor of 9).

The short analysis presented above indicates that if the code of the 2-D Danish Eulerian Model is scalable, then the computing times should be increased by factors approximately equal to 27 and 9 in the advection part and the chemical part respectively in the transition from a (96×96) grid to the refined (288×288) grid. For the code of the 3-D Danish Eulerian Model the increasing factors for the advection and chemistry part are the same, 27 and 9 respectively, as for the code of the 2-D Danish Eulerian Model. Furthermore the computing time for the vertical exchange part should be increased by a factor of 9 in the transition from a (96×96) grid to the refined (288×288) grid if the code is scalable.

Some runs have been performed in an attempt to establish whether the code is scalable or not. Results obtained in the transition from a (96×96) grid to the refined (288×288) grid for the 2-D Danish Eulerian Model are given in Table 5.4 The results given in Table 5.4 indicate that the ratios of the computing times for the refined grid and the coarser grid are close to the expected ratios (see these ratios in Table 5.4). Our experiments suggest that the application is scalable, but of course this conclusion is restricted to the problem sizes considered here.

We have also studied what is happening in the transition from the 2-D Danish

TABLE 5.4

Computing times obtained by using a refined (288×288) grid and a coarse (96×96) grid with the 2-D Danish Eulerian Model. The ratios of the times for the refined and coarse grids are given in the last column. The two codes were run on 16 processors of the IBM SMP computer by using OpenMP within each node and MPI across the two nodes.

Process	(288×288)	(96×96)	Ratio
Advection	1523	63	24.6
Chemistry	2883	288	10.0
Total	6209	424	14.6

Eulerian Model discretized on a (96×96) grid to the the 3-D Danish Eulerian Model discretized on a $(96 \times 96 \times 10)$ grid. In this case the computing times for the advection part and for the chemical part is increased by a factor of 10 when the 3-D version is used, while the computing time for the vertical exchange part is relevant only for the 3-D version. Some processes (the processes that are relevant for the surface layer) are common for the 2-D version and the 3-D version. Therefore, it is not very clear whether the total computing time will be increased by a factor greater than 10 or not in the transition from the 2-D version to the 3-D version. Some results indicate that the increasing factor for the total computing time is less than 10. However, more experiments are needed in order to understand the situation better.

The conclusion (that the code is scalable, which was drawn by using the results shown in Table 5.4) was further supported:

- by using results obtained in some runs with the more precise version in which the space domain is discretized on a (480×480) grid and
- by performing, on the SGI Origin 2000 computer, many runs with the 3-D version obtained by using a $(96 \times 96 \times 10)$ grid and comparing the results with the corresponding results calculated by applying the 2-D version obtained by using a (96×96).

6. Portability of the code. Some versions of the Danish Eulerian Model have recently been run on several other parallel computers such as a SUN E10000 shared memory computer with up to 16 processors at SUN's Global Customer Benchmarking HPC Center in Beaverton (Oregon) and a CRAY T3E distributed memory computer with up to 64 processors at EPCC (Edinburgh Parallel Computing Centre). Rather good results have been achieved without any need to make changes in the code. Some of these results are given in Table 6.1 -Table 6.4.

The OpenMP version of the 3-D Danish Eulerian Model is used on the SUN computer (a SUN E10000 server using UltraSPARC-II processors running at 333MHz, each of them having 4MB of L2 caches) to produce the results given in Table 6.1 and Table 6.3. The comparison of the results in Table 6.1 with the results in Table 5.1 indicates that the efficiency of the parallel computation is fully preserved in the transition from one shared memory computer to another.

The MPI version of the 2-D Danish Eulerian Model applied on a refined (480×480) grid is used on the CRAY T3E computer to produce the results given in Table 6.2. As mentioned in Section 6, the code is so large that it cannot be run if only a few processors are used. The results in Table 6.2 show that the MPI version of the refined 2-D Danish Eulerian Model runs rather efficiently not only on the IBM SP computer, but also on the CRAY T3E computer.

Some runs with the 2-D version of the Danish Eulerian Model applied on a coarser (96×96) grid have also been carried out. Results obtained with the OpenMP code

on the SUN computer are given in Table 6.3. The corresponding results obtained by the MPI code on the CRAY T3E computer and on Macintosh Power PC Cluster are given in Table 6.4 and Table 6.5. These results confirm, once again, the statement that the codes (both the OpenMP codes and the MPI codes) can easily be ported from one computer to another.

The results could probably be improved by some tuning. Nevertheless, the results show clearly that one should use standard parallelization tools in the attempt to run efficiently the code on different modern supercomputers. This facilitates the transition from one supercomputer to another.

6.1. MPI versus OpenMP. OpenMP is normally the preferred option when shared memory machines are used. The obvious reason for this is the fact that it is much easier to implement an OpenMP version. In many cases the compiler itself will find out where to carry out parallel computations. However, some researchers have obtained better results by using MPI also on shared memory computers. Therefore it is necessary to perform some comparisons in order to decide what is the best possibility when the Danish Eulerian Model is to be run. Such comparisons have been carried out. Some results are given in Table 6.6.

TABLE 6.1
Computing times (measured in seconds) obtained by using OpenMP on the SUN computer at Global Benchmarking Center in Oregon when the 3-D version of the Danish Eulerian Model is discretized on a $(96 \times 96 \times 10)$ *grid.*

Processors	Comp. time	Speed-up	Efficiency
1	52615	-	-
8	6847	7.68	96%
16	3586	14.67	92%

TABLE 6.2
Computing times (measured in seconds) obtained by using MPI on the CRAY T3E computer at EPCC when the 2-D version of the Danish Eulerian Model is discretized on a (480×480) *grid.*

Processors	Comp. time	Speed-up	Efficiency
32	18306	-	-
64	9637	1.90	95%

TABLE 6.3
Computing times (measured in seconds) obtained by using OpenMP on the SUN computer at Global Benchmarking Center in Oregon when the 2-D version of the Danish Eulerian Model is discretized on a (96×96) *grid.*

Processors	Comp. time	Speed-up	Efficiency
1	5402	-	-
8	743	7.27	91%
16	429	12.59	79%

It is seen that although there are three extra processes which have to be carried out when the MPI version is run, the total computing time for this version is considerably smaller (and the speed-up is higher than the speed up obtained when the OpenMP version is run; the speed-ups being 14.5 and 10.3 respectively). The fact that one

TABLE 6.4
Computing times (measured in seconds) obtained by using MPI on the CRAY T3E computer at EPCC when the 2-D version of the Danish Eulerian Model is discretized on a (96×96) grid.

Processors	Comp. time	Speed-up	Efficiency
1	7503	-	-
16	506	14.83	93%

TABLE 6.5
Computing times (measured in seconds) obtained by using MPI on the Macintosh Power PC cluster at the Bulgarian Academy of Sciences when the 2-D version of the Danish Eulerian Model is discretized on a (96×96) grid.

Processors	Comp. time	Speed-up	Efficiency
1	5792	-	-
8	787	7.36	92%

works with shorter arrays and, thus, the cache memory is better exploited in the MPI version (because of the domain decomposition) is one of the reasons for the better performance. However, even in the chemical part, where the chunks used in both versions are the same, the MPI version performs better.

The above conclusion is only valid for the current implementations. It is clear that the MPI implementations also scales on shared memory systems. More experimentation and analysis is needed to study and possibly improve the efficiency of the OpenMP versions of the Danish Eulerian Model.

7. Applications of the Model. Two examples are given in this section in order to illustrate the real need for high-speed computations in the area of environmental modelling. In the first application long-term computations were performed in order to study trends of the air pollution levels over a period of 10 years. The fine resolution version of DEM is used in the second example.

7.1. Studying relationships between emissions and pollution levels in Denmark. A 10-year run of DEM was performed in order to investigate the relationship between the emissions in Europe and the pollution levels. The calculated data has been used to study the relationship in the Danish area, but a similar study can be performed for any other country in Europe.

The variation of the emissions in Denmark is shown in Fig. 2. It is seen that the ammonia emissions in Denmark were not reduced in this period. However, the variation of the ammonia-ammonium concentrations in Denmark, shown in Fig.3, indicates a clear trend of reductions. Moreover, the same trend is seen both when the variation of the measurements taken in three Danish sites are taken into account and when model results are studied. The fact that the pollution levels are reduced even when the emissions in Denmark remain the same deserve some explanation. The results in Table 7.1 show that while, Denmark has not reduced its emissions, considerable reductions were achieved in two of the neighboring countries. Thus, the reduction of the pollution levels in Denmark is caused by the reduction of the transport of ammonia-ammonium to Denmark.

TABLE 6.6
Computing times (measured in seconds) obtained by applying the OpenMP version and the MPI version when 16 processors of the SGI Origin computer are used. The 2-D version of the Danish Eulerian Model discretized on a (96×96) grid is run in this experiment.

Process	OpenMP version	MPI version
Start	0.1	12.4
Wind + Sinks	5.8	2.2
Advection	101.2	30.1
Chemistry	232.6	161.9
Input-output	54.2	4.1
Communications	0.0	46.9
Preprocessing	0.0	11.1
Post-processing	0.0	12.0
Total time	394.1	270.5

TABLE 7.1
Ammonia emissions in three European countries.

Country	1989	1998	Reduction
Germany	661	502	24%
The Netherlands	232	171	24%
Denmark	104	104	0%

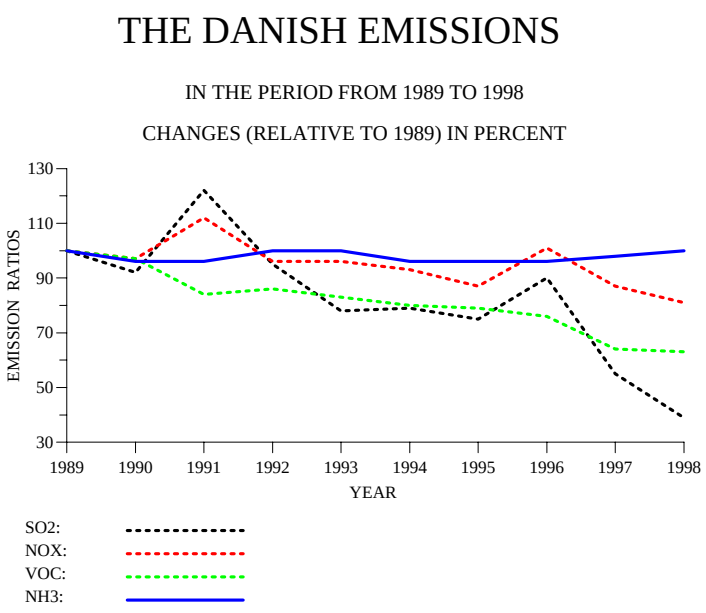

Figure 2

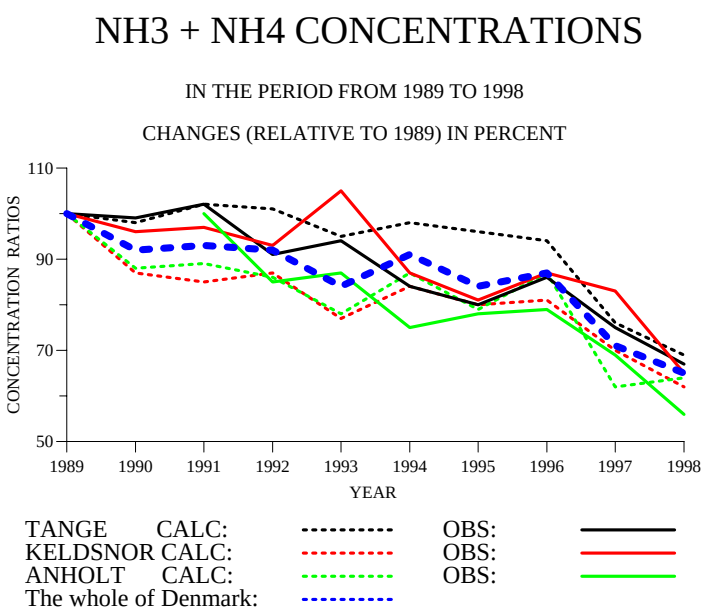

Figure 3

7.2. Using finer resolution. For small countries, such as Denmark, the use of finer resolution models is important, because this allows us to zoom to prescribed areas and to see more details in the different parts of the country. Some results are shown in Fig. 4 and Fig. 5.

The distribution of the nitrogen dioxide pollution in Europe is shown in Fig. 4. It is seen that the most polluted areas in Europe are parts of England, the Netherlands, Belgium, Germany and parts of France, the Check Republic and Poland as well as the Northern part of Italy. Moreover, in the parts of Europe which are not very polluted one can locate large cities, such as Madrid, Rome, Oslo, Stockholm, Helsinki, Sct. Petersburg and Moscow, which are large sources of nitrogen pollution (the most important reason being the fact that pollution from the traffic is one of the major sources for nitrogen emissions).

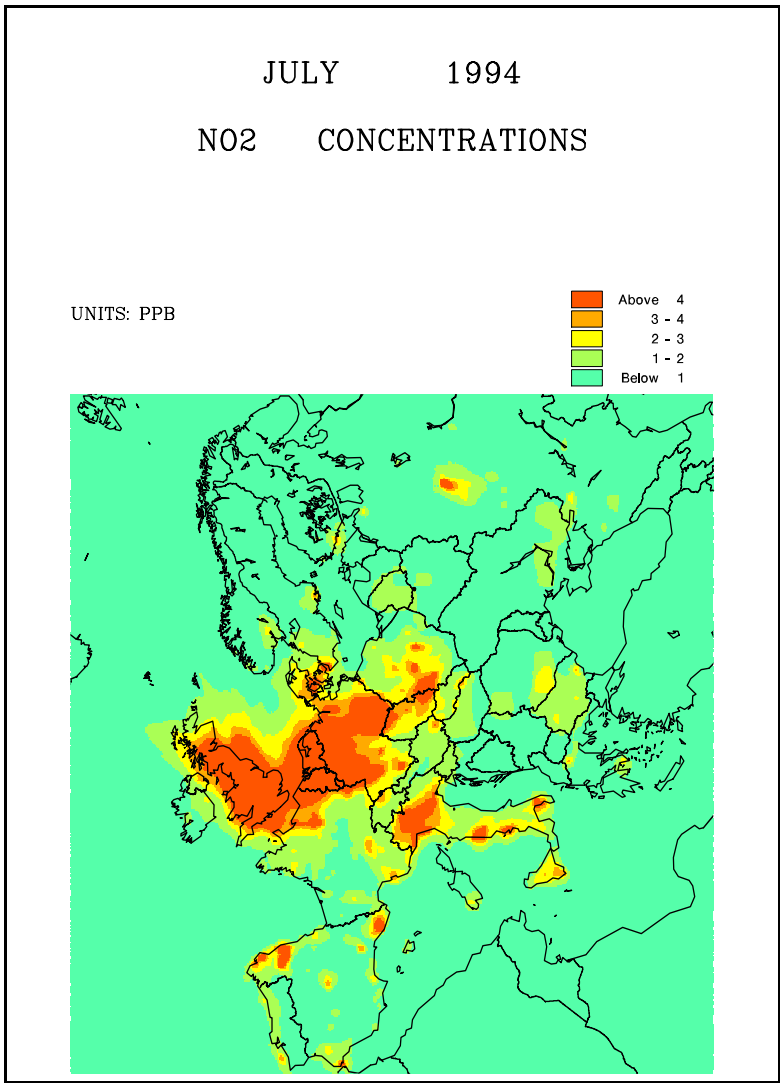

Figure 4

The distribution of the nitrogen dioxide concentrations in different parts of Europe and its surroundings.

The same output data as those used to draw Fig. 4 are also used to draw Fig. 5. Indeed, Fig. 5 could be viewed as a result from zooming in Fig. 4 onto the area around Copenhagen, the capital of Denmark, and the Swedish city of Malmö.

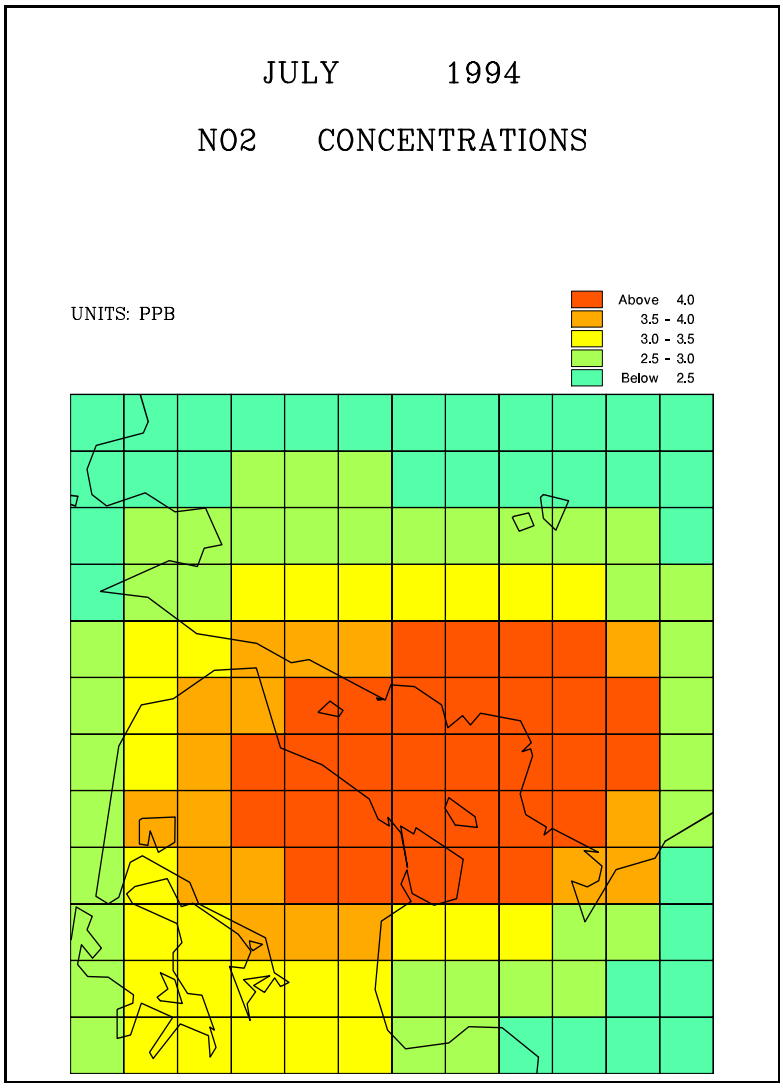

Figure 5

The nitrogen dioxide concentrations in area around the Danish capital Copenhagen and the Swedish city Malmö (the Øresund region).

The grid lines are drawn in Fig. 5. This is impossible when the whole space domain is used. Indeed, Fig. 4 will become complete black if the option for drawing the grid-lines is not switched off when results on the whole space domain, containing 230 400 grid-squares, are plotted. The number of grid-squares used in Fig. 5 is 144, obtained by using only a tiny part, a (12×12) sub-grid, of the whole (480×480) grid. At the same time, this area is nearly twice smaller than one grid-square of the (32×32) grids which were commonly used only a few years ago, and are still used in some models (see, for example, Amann et al. [2] and the EMEP Report 1/98 [7]). While the size of only one grid-square is 22500 km^2 when a (32×32) grid is applied, the size of the area of 144 grid-squares used in Fig. 5 is 14400 km^2. This means that, while it will not be possible to see any difference in the area shown in Fig. 5 when a

coarse (32×32) grid is used, the results from the high resolution model shown in Fig. 5 show clearly that there are several different levels of nitrogen pollution in this area. This illustrates the great potential power of the high resolution models. However, there is a price that is to be paid: very large sets of digital data are to be handled when such models are run.

8. Concluding Remarks and Plans for Future Work. Several conclusions can be drawn by using the results presented in the previous sections.

The most important conclusion is the use of standard parallelization tools, such as OpenMP or MPI, simplifies the transition from one computer to another. This has been demonstrated in this paper by running a very big application code (i.e. a code with large memory and execution time requirements on several different computers.

The efficient use of parallel computers allows the scientists to enlarge the class of problems that can be successfully handled, i.e. it becomes possible to solve more problems and bigger problems.

There are still many unresolved problems. Many of the techniques that are currently used in the Danish Eulerian Model will not be very efficient or will not work if the number of available processors is very large (say several hundreds processors or even several thousands processors). It is nor very clear how to run the model on heterogeneous computers and/or on a grid of computers. The solution of these problems is a very challenging task. We are planning to start soon some work in this direction.

REFERENCES

[1] V. Alexandrov, A. Sameh, Y. Siddique and Z. Zlatev, *Numerical integration of chemical ODE problems arising in air pollution models*, Environmental Modelling and Assessment, Vol. 2 (1997), pp. 365–377.

[2] Amann, M., Bertok, I., Cofala, J., Gyarfas, F., Heyes, C., Klimont, Z., Makowski, M., Schöp, W. and Syri, S. (1999). *Cost-effective control of acidification and ground-level ozone.* Seventh Interim Report, IIASA (International Institute for Applied System Analysis), Laxenburg, Austria.

[3] C. Bendtsen and Z. Zlatev, *Running air pollution models on message passing machines*, in: Parallel Virtual Machine and Message Passing Interface (M. Bubak, J. Dongarra and J. Wasniewski, eds.), pp. 417–426. Springer-Verlag, Berlin, (1997).

[4] A. Bastrup-Birk, J. Brandt, I. Uria and Z. Zlatev, *Studying cumulative ozone exposures in Europe during a seven-year period*, Journal of Geophysical Research, Vol. 102 (1997), pp. 23917–23935.

[5] W. P. Crowley, *Numerical advection experiments*, Monthly Weather Review, Vol. 96 (1968), pp. 1–11.

[6] I. S. Duff, A. M. Erisman and J. K. Reid, *Direct methods for sparse matrices,* Oxford University Press, Oxford-London (1986).

[7] EMEP (1998). *Transboundary acidifying air pollution in Europe: Part 1, Estimated dispersion of acidifying and eutrofying compounds and comparison with observations.* Status Report 1/98. EMEP MSC-W (Meteorological Synthesizing Centre - West), Norwegian Meteorological Institute, P. O. Box 43 - Blindern, N-0313 Oslo, Norway.

[8] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, *PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Networking Parallel Computing.* MIT Press, Cambridge, Massachusetts, 1994.

[9] K. Georgiev and Z. Zlatev, Running an advection-chemistry code on message passing computers, In *Recent Advances in Parallel Virtual Machine and Message Passing Interface* (V. Alexandrov and J. Dongarra, eds.), pages 354-363. Springer, Berlin, 1998.

[10] K. Georgiev and Z. Zlatev, Parallel Sparse Matrix Algorithms for Air Pollution Models, *Parallel and Distributed Computing Practices*, Vol. 2 (1999), pp. 429-442.

[11] M. W. Gery, G. Z. Whitten, J. P. Killus and M. C. Dodge, *A photochemical kinetics mechanism for urban and regional computer modeling,* Journal of Geophysical Research, Vol. 94 (1989), pp. 12925–12956.

[12] W. Gropp, E. Lusk and A. Skjellum, *Using MPI: Portable programming with the message passing interface*, MIT Press, Cambridge, Massachusetts (1994).

[13] E. Hairer and G. Wanner, *Solving ordinary differential equations, II: Stiff and differential-algebraic problems.* Berlin, Springer Verlag (1991).

[14] E. Hesstvedt, Ø. Hov and I. A. Isaksen, *Quasi-steady-state approximations in air pollution modelling: comparison of two numerical schemes for oxidant prediction*, International Journal of Chemical Kinetics, Vol. 10 (1978), pp. 971–994.

[15] Ø. Hov, Z. Zlatev, R. Berkowicz, A. Eliassen and L. P. Prahm, *Comparison of numerical techniques for use in air pollution models with non-linear chemical reactions*, Atmospheric Environment, Vol. 23 (1988), pp. 967–983.

[16] D. J. Kuck, E. S. Davidson, D. H. Lawrie and A. H. Sameh, Parallel supercomputing today and the CEDAR approach, *Science*, 231:967-974, 1986.

[17] J. D. Lambert, *Numerical methods for ordinary differential equations.* New York, Wiley (1991).

[18] G. I. Marchuk, *Mathematical modeling for the problem of the environment*, Studies in Mathematics and Applications, No. 16, North-Holland, Amsterdam (1985).

[19] G. J. McRae, W. R. Goodin and J. H. Seinfeld, *Numerical solution of the atmospheric diffusion equations for chemically reacting flows*, Journal of Computational Physics, Vol. 45 (1984), pp. 1–42.

[20] C. R. Molenkampf, *Accuracy of finite-difference methods applied to the advection equation,* Journal of Applied Meteorology, Vol. 7 (1968), pp. 160–167.

[21] WEB-site for OpenMP tools, *http://www.openmp.org*, 1999.

[22] W. Owczarz and Z. Zlatev, Running a large air pollution model on an IBM SMP computer, *International Journal on Computer Research*, to appear.

[23] W. Owczarz and Z. Zlatev, Parallel matrix computations in air pollution modelling, Internal report, National Environmental Research Institute, DK-4000 Roskilde, Denmark, 2000.

[24] D. W. Pepper and A. J. Baker, *A simple one-dimensional finite element algorithm with multidimensional capabilities*, Numerical Heath Transfer, Vol. 3 (1979), 81-95.

[25] D. W. Pepper, C. D. Kern and P. E. Long, Jr., *Modelling the dispersion of atmospheric pollution using cubic splines and chapeau functions*, Atmospheric Environment, Vol. 13 (1979), pp. 223–237.

[26] L. K. Peters, C. M. Berkowitz, G. R. Carmichael, R. C. Easter, Fairweather, G. Ghan, J. Hales, L. Leung, W. Pennell, F. Potra, R. D. Saylor and T. Tsang, *The current state and future direction of Eulerian models in simulating the tropospherical chemistry and transport of trace species: A review.* Atmospheric Environment, Vol. 29 (1995), pp. 189–221.

[27] A. Sandu, J. Verwer, J. Bloom, E. Spee and G. Carmichael, *Benchmarking stiff ODE systems for atmospheric chemistry problems: II. Rosenbrock solvers*, Atmospheric Environment, Vol. 31 (1997), pp. 3459–3472.

[28] J. Swart and J. Blom, *Experience with sparse matrix solvers in parallel ODE software*, Comp. Math. Appl., Vol. 31 (1996), pp. 43–55.

[29] R. A. Willoughby, *Sparse matrix algorithms and their relation to problem classes and computer architecture*, In: Large Sparse Sets of Linear Equations (J. K. Reid, ed.), pp. 255-277. Academic Press, London-New York, (1970).

[30] Z. Zlatev, *Application of predictor-corrector schemes with several correctors in solving air pollution problems*, BIT, Vol. 24 (1984), pp. 700–715.

[31] Z. Zlatev, *Computational methods for general sparse matrices*, Kluwer Academic Publishers, Dordrecht-Boston-London (1991).

[32] Z. Zlatev, *Computer treatment of large air pollution models*, Kluwer Academic Publishers, Dordrecht-Boston-London (1995).

[33] Z. Zlatev, I. Dimov and K. Georgiev, *Three-dimensional version of the Danish Eulerian Model*, Zeitschrift für Angewandte Mathematik und Mechanik, Vol. 76 (1996), pp. 473–476.

[34] Z. Zlatev, J. Fenger and L. Mortensen, *Relationships between emission sources and excess ozone concentrations*, Computers and Mathematics with Applications, Vol. 32, No. 11 (1996), pp. 101–123.