Efficient treatment of large-scale air pollution models on supecomputers

Zahari Zlatev

National Environmental Research Institute
Frederiksborgvej 399, P. O. Box 358, DK-4000 Roskilde, Denmark

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 two-three decades. 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 (288x288) grid and that the number of chemical species studied by the model is 35. Then ODE systems containing 2903040 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 air pollution model is to be used, then the number of equations must be multiplied by the number of layers. The treatment of large air pollution models on modern parallel computers by using efficient numerical algorithms will be discussed in this paper.

1 Large-scale air pollution models

Mathematical models are indispensable tools when different air pollution phenomena are to be studied on a large space domain (say, the whole of Europe). Such models can be used to find out when and where certain critical levels are exceeded or will soon be exceeded. All important physical and chemical processes must be taken into account in order to obtain reliable results. This leads to huge computational tasks, which have to be handled on big high-speed computers.

A short description of the particular air pollution model used, the Danish Eulerian Model, [11], will be given in this section in order to illustrate the numerical difficulties that are to be overcome when large air pollution models are treated on modern computers. We shall concentrate our attention on the mathematical backgrounds of this model. However, many details about the physical and chemical processes involved in the models as well as many of the validation tests and different air pollution studies that were carried out with the Danish Eulerian Model are documented in [2], [4], [5], [9], [11]-[16]. Many comparisons of model results with measurements can be seen in web-site of this model, [8].

The Danish Eulerian Model is described by a system of partial differential equations (PDE's):

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2074, pp. 82–91, 2001.

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (uc_s)}{\partial x} - \frac{\partial (vc_s)}{\partial y} - \frac{\partial (wc_s)}{\partial z} \tag{1}$$

$$+\frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right)$$

$$+E_s - (\kappa_{1s} + \kappa_{2s})c_s + Q_s(c_1, c_2, \ldots, c_q), \quad s = 1, 2, \ldots, q.$$

The notation used in the system of PDEs (1) can be explained as follows: (i) the concentrations of the chemical species are denoted by c_s, (ii) the variables u, v and w are wind velocities, (iii) K_x, K_y and Kz are diffusion coefficients, (iv) the emission sources are described by E_s, (v) k_{1s} and k_{2s} are deposition coefficients (dry deposition and wet deposition, respectively) and (vi) the chemical reactions are represented by the non-linear functions $Q_s(c_1, c_2, \ldots, c_q)$.

2 Splitting procedures

It is difficult to treat the system of PDE's (1) directly. This is the reason for using different kinds of splitting. A splitting procedure, based on ideas proposed in [6] and [7], leads to five sub-models, representing respectively: (i) the horizontal transport (the advection) of the pollutants in the atmosphere, (ii) the horizontal diffusion, (iii) the chemistry (together with the emission terms), (iv) the deposition and (v) the vertical exchange. These sub-models are transformed to systems of ordinary differential equations, ODEs, by using different kinds of space discretization. The splitting procedure, actually used when the Danish Eulerian Model is handled on computers, is discussed in detail in [11] and [14]. The most time- consuming sub-models are the horizontal transport and the chemistry. The numerical methods used in these sub-models are shortly discussed in the next two sections.

3 Horizontal Transport

The horizontal transport (the advection) is described mathematically by the following system of q independent PDEs:

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (uc_s)}{\partial x} - \frac{\partial (vc_s)}{\partial y}, \quad s = 1, 2, \ldots, q. \tag{2}$$

Several numerical algorithms have been tried to handle the horizontal transport sub-model ([5], [11], [14], [15]). All numerical methods, which have been tried, have both advantages and drawbacks. Two of the methods gave good results: the pseudo-spectral algorithm and a finite element method.

3.1 The pseudo-spectral algorithm

If only the accuracy is considered, then the pseudo-spectral algorithm ([9], [11]) seems to be the best one among the tested methods. However, it requires periodical boundary conditions, which is a serious drawback especially when grid-refinement on some parts of the space domain is to be used.

The pseudo-spectral algorithm applied in a version of the Danish Eulerian Model is fully described in [11]. Approximations of the first order spatial derivatives are obtained by expanding the unknown functions c_s in Fourier series containing *sine* and *cosine* functions. The truncation of this series leads to a trigonometric polynomial. The derivative of this polynomial is the required approximation of the first order spatial derivative under consideration. The use of Fourier series implies a requirement for periodicity of the concentrations on the boundaries. If high accuracy is to be achieved by keeping not too many terms in the Fourier series, then an additional requirement for periodicity of the derivatives of the concentrations up to some order n has also to be satisfied. The last requirement is a serious drawback of this algorithm, but the results are rather good if the space domain is very large and if some care is taken in order to obtain periodicity on the boundaries. The method is not suitable for modern procedures based on local refinement in some parts of the space domain.

3.2 Application of a finite element approach

The finite element approach, [5], [11], is in general less accurate (unless very complicated finite elements are used), but it is much more flexible when grid-refinement is needed in some parts of the space domain. Moreover, it is easier to develop parallel codes for the advection sub-model when some kind of finite elements are used. The finite element approach can be considered, as many other numerical methods for solving systems of PDEs, as a method based on expanding the unknown functions, the concentrations, in Taylor series. Truncating these series one arrives at algebraic polynomials. The derivatives of these polynomials are then used to obtain approximations to the first order spatial derivatives of the concentrations. In the following part of this paper it will be assumed that the finite elements algorithm from [3] is used.

4 Treatment of the Chemical Part

The chemical part of an air pollution model (including the emissions) is described mathematically by a non-linear and stiff system of ODEs, which can be written in the following form:

$$dg/dt = f(g,t), \quad g \in \mathbf{R}^{N_x \times N_y \times N_z \times q}, \quad f \in \mathbf{R}^{N_x \times N_y \times N_z \times q}, \tag{3}$$

where N_x, N_x and N_x are the numbers of grid-points along the coordinate axes, g contains approcximations of the concentrations at the grid-points and f depends on the chemical reactions.

This is normally the most time-consuming part. Also here different numerical algorithms have been tried. A detailed discussion of the algorithms used until now is given in [1]. Three of the used until now numerical methods will be shortly discussed here: (i) the QSSA (Quasi-Steady-State-Approximation) algorithm, (ii) some classical methods for solving systems of ODEs and (iii) numerical methods based on partitioning the system of ODEs and using the Backward Euler Method to treat the partitioned system.

4.1 The QSSA algorithm

The QSSA algorithm is based on a division of the chemical reactions into three groups and trying to use different devices for every group. It is simple and fast. Moreover, it is easy to implement this algorithm on parallel computers. The low accuracy of the algorithm is its major drawback. There are several improvements of the algorithm. More details about the QSSA algorithm and its performance can be found in [1].

An improved version of the QSSA algorithm, which is also discussed in [1], will be used in this paper. An attempt to improve the performance of the algorithm is made in the improved version.

4.2 Using classical numerical methods

Classical numerical methods for solving systems of ODEs are also applicable. The methods used until now were: (i) the Backward Euler Method, (ii) the Trapezoidal Rule and (iii) a second order Runge-Kutta algorithm.

If these methods are carefully implemented they can become competitive with the QSSA algorithm when performance is considered and, furthermore, the classical numerical methods are more accurate than the QSSA algorithm. There are, however, still some open problems when the classical methods are to be used on parallel architecture. The major difficulties are connected with the quasi-Newton iterative methods, which have to be used, because all considered methods are implicit. The iterative procedure converges with different rates in different parts of the space domain. This leads to lack of loading balance and, thus, deteriorates the performance of the code on the parallel computers. It should be mentioned, however, that some progress has been achieved recently (see, for example, [3]).

Some more details about the classical methods for solving systems of ODEs, which have been used until now, can be found in [1].

4.3 Using partitioning

Some of the chemical reactions used in a large-scale air pollution model are slow, while the others are fast. This fact can be exploited in the solution of the system of ODEs arising in the chemical sub-model. A method based on this idea has been developed and tested in [1]. The system of ODEs is first partitioned and

then the Backward Euler Method is used to handle the partitioned system. The results were very encouraging. However, if the method has to be applied on a parallel computer, then difficulties, which are quite similar to those arising when classical numerical methods are used, have to be overcome. The conclusion is that the partitioning methods can successfully be used on a sequential computer, but some more work is needed in order to make them more efficient when a parallel computer is to be applied.

5 Major Modules of the Code

When a code for running a large-scale air pollution model on computers is developed, then it is important to have different easily exchangeable modules for the different parts of the computational work. Two two-dimensional versions of the Danish Eulerian model, which are discussed in this paper, consists of the following five major modules:

- **Advection module.** This module treats the horizontal transport. The diffusion subroutines have been added to this module.
- **Chemistry module.** This module handles the chemical reactions involved in the model. The deposition subroutines have been added to this module.
- **Initialization module.** All operations needed to initialize arrays and to perform the first readings of input data are united in this module.
- **Input operations module.** Input data are read in this module when this is necessary; otherwise simple interpolation rules are used, both in space and in time, to produce the needed data.
- **Output operations module.** This module prepares output data. The total number of output files prepared in the 2-D versions of the model is at present eight. The total amount of the output data is about 675 Mbytes when the 2-D model is discretized on a (288x288) grid and run over a time-period of one month (but it must be emphasized here that the model is normally run over a time-period of several, up to ten, years). For some of the concentrations (ozone and nitrogen dioxide) hourly mean values are calculated. For other concentrations and some depositions, diurnal mean values are prepared. For all concentrations and depositions studied by the model, monthly mean values are calculated in one of the files.

The initialization module is called only once (in the beginning of the computations), while all other modules are called at every time-step.

6 Use of Parallel Computers

The performance of two 2-D versions of the Danish Eulerian Model will be discussed in this section. The first version is discretized on a (96x96) grid, while the second one is discretized on a (288x288) grid. It is assumed that the model is run over a time-period of one month. In practice, many such runs, several

hundreds or even several thousands, are to be performed. The computational tasks that are to be handled in these two versions consist of four systems of ODEs that have to be treated during more than 10 thousand time-steps. The two versions have been run on several large supercomputers (such as Fujitsu, SGI Origin 2000, IBM SP, etc.). The performance of the code on IBM SMP nodes will be demonstrated in this section.

6.1 The IBM SMP Nodes

Every IBM SMP nodes of the available architecture at the Danish Computing Centre contains 8 processors. There are two nodes at present. Each node can be considered as a shared memory computer. A combination of several nodes can be considered as a distributed memory machine. Thus, there are two levels of parallelism: (i) a shared memory mode within each node of the IBM SMP computer and (ii) a distributed memory mode across the nodes of the IBM SMP computer.

6.2 Shared Memory Mode

Assume first that the computer is to be used as a shared memory machine. It is important to identify large parallel tasks in the two most time-consuming parts of the code: (i) the advection module and (ii) the chemistry module.

6.2.1. Parallel Tasks in the Advection Module. In the advection module, the horizontal transport of each pollutant can be considered as a parallel task. The number of pollutants in the Danish Eulerian Model is 35. This means that the loading balance is not perfect. This is partly compensated by the fact that the parallel tasks so selected are very large.

6.2.2. Parallel Tasks in the Chemical Module. In the chemical module, the chemical reactions at a given grid-point can be considered as parallel tasks. This means that the number of parallel tasks is very large (i.e. the loading balance is perfect). This number is (i) 82944 in the version of the model that is discretized on a (288x288) grid and (ii) 9216 in the version of the model that is discretized on a (96x96) grid. However, the tasks are not large. Therefore it is convenient to group several grid-points as a bigger task. It will be shown that the performance can be improved considerably by a suitable choice of the groups of grid-points.

6.3 Distributed Memory Mode

Consider now the case where parallelism across the nodes is to be utilized. Then we need two extra modules: (i) a pre-processing module and (ii) a post-processing module.

6.3.1. Pre-processing Module. The first of these modules performs a pre-processing procedure. In this procedure the data is divided into several parts (according to the nodes that are to be applied) and sent to the assigned for the

job nodes. In this way each node will work on its own data during the whole run.

6.3.2. Post-processing Module. The second module performs a post-processing procedure. This means that every node prepares during the run its own output files. At the end of the job, the post-processing module is activated (i) to collect the output data on one of the nodes and (ii) to prepare them for future applications (visualizations, animations, etc.).

6.3.3. Benefits of Using the Additional Modules. By the preparation of the two additional modules, one avoids some excessive communications during the run. However, it should also be stressed that not all communications are avoided. The use of the pre-processor and the post-processor is in fact equivalent to the application of a kind of domain decomposition. Therefore, it is clear that some communications are needed along the inner boundaries of the domains. Such communications are to be carried out only once per step and only a few data are to be communicated. Thus, the actual communications that are to be carried out during the computational process are very cheap.

6.4 Numerical results

Many hundreds of runs have been carried out in order to check the performance under different conditions. The most important results are summarized below.

6.4.1. Choosing the size of the parallel tasks. One of the crucial issues is the proper division of the grid-points into a suitable number of parallel tasks in the chemical module. Tests with more than 20 numbers of grid-points per parallel task (i.e. with different sizes of the parallel tasks) have been carried out. Some results are given in Table 1.

Size	Advection	Chemistry	Total time
1	62	586	730
24	63	276	432
48	63	272	424
96	64	288	447
576	61	389	612

Table 1

Computing times (given in seconds) obtained in runs of the version discretized on a (96x96) grid on the IBM SMP computer with different numbers of grid-points per parallel task in the chemical part. The numbers of grid-points per parallel task in the chemical part are given in the first column (under "Size"). The number of processors used in these runs is 16.

Three conclusions can be drawn from the tests with different sizes of the parallel tasks: (i) if the parallel tasks are too small, then the computing times are large, (ii) if the parallel tasks are too large, then the computing times are also large and (iii) for tasks with medium sizes the performance is best. It is difficult to explain this behaviour. The following explanation should be further tested and

justified. If the tasks are very small (1,2 and 4 grid- points per parallel task), then the overhead needed to start the large number of parallel tasks becomes considerable. On the other hand, if the parallel tasks are too large (576 grid-points per parallel task), then the data needed to carry out them are too big and cannot stay in cache. Parallel tasks of medium sizes seem to be a good compromise.

The maximum number of grid-points per parallel tasks when the version of the model discretized on a (96x96) grid is 9216/p, where p is the number of processors used. The maximum number is 576 when 16 processors are used

6.4.2. Runs on different numbers of processors. The version of the model, which is discretized on a (96x96) grid, has been run on 1, 2, 4, 8 and 16 processors. When the number of processors is less than 16, only one node of the IBM SMP computer has been used. This means that in this case the code works in a shared memory mode only. Two nodes of the IBM SMP computer are used when 16 processors are selected. In this case a shared memory mode is used within each node, while the code performs in a distributed memory mode across the nodes. Some results are given in Table 2 (computing times) and Table 3 (speed-ups and efficiency of the parallel computations).

The results show clearly that good performance can be achieved in both cases: (i) when the code is running in a shared memory mode during the whole run (this happens when the number of processors is less than 16) and (ii) when the code is running by utilizing both modes (this is the case when the number of processors is 16). OPEN MP instructions are used when the code distributed memory mode. runs in a shared memory mode, while MPI subroutines are called when the code is run in a distributed memory mode.

6.4.3. Scalability of the code. It is important to be able to run efficiently the code when the grid is refined. Some tests with a version of the code, which is discretized on a (288x288) grid, have been carried out. There are plans to carry out some runs with a version of the code discretized on a (480x480) grid in the near future.

The computational work in the advection part is increased by a factor of 27 when the (288x288) grid is used, because the number of grid-points is increased by a factor of 9, while the time-step is decreased 3 times.

The computational work in the chemical part is increased only by a factor of 9 when the (288x288) grid is used, because the number of grid points is increased by a factor of 9, while the time-step used in the chemical part remains unchanged.

A few runs with the version discretized on a (288x288) grid have been carried out and compared with the corresponding runs for he version discretized on a (96x96) grid. The purpose was to check if the times spent are changed as should be expected (about 27 times for the advection part and about 9 time for the chemical part) when the same number of processors are used. Some results are given in Table 4. It is seen that the results are rather closed to the

Processors	Advection	Chemistry	Total time
1	933	4185	5225
2	478	1878	2427
4	244	1099	1405
8	144	521	799
16	62	272	424

Table 2

Computing times (given in seconds) obtained in runs of the version discretized on a (96x96) grid on the IBM SMP computer with different number of processors. The number of grid-points per parallel task in the chemical part is 48.

Processors	Advection	Chemistry	Total time
2	1.95 (98%)	2.23 (112%)	2.15 (108%)
4	3.82 (96%)	3.81 (95%)	3.72 (93%)
8	6.48 (81%)	8.03 (100%)	6.54 (82%)
16	15.01 (94%)	15.39 (96%)	12.32 (77%)

Table 3

Speed-ups and efficiency (given in brackets) obtained in runs of the version discretized on a (96x96) grid on the IBM SMP computer with different number of processors. The number of grid-points per parallel task in the chemical part is 48. expected results. Some more experiments are needed.

Process	(288x288)	(96x96)	Ratio
Advection	1523	63	24.6
Chemistry	2883	288	10.0
Total	6209	432	14.4

Table 4

Computing times (given in seconds) obtained in runs of the two versions of the code, discretized on a (96x96) grid and on a (288x288) grid, on the IBM SMP computer by using 16 processors. The number of grid-points per parallel task in the chemical part is 96.

7 Conclusions

Many important environmental studies can successfully be performed only if (i) efficient numerical methods are implemented in the mathematical models used and (ii) the great potential power of the modern supercomputers is utilized in an optimal way.

Some results obtained in the effort to solve these two tasks have been presented. There are still a lot of unresolved problems. The efforts to find more efficient numerical methods for the large air pollution models and to optimize further the code for runs on parallel machines are continuing.

References

1. Alexandrov, V., Sameh, A., Siddique, Y. and Zlatev, Z., Numerical integration of chemical ODE problems arising in air pollution models, Environmental Modelling and Assessment, Vol. 2, 1997, 365–377.
2. Bastrup-Birk, A., Brandt, J., Uria, I. and Zlatev, Z., Studying cumulative ozone exposures in Europe during a seven-year period, Journal of Geophysical Research, Vol. 102, 1997, 23917-23935.
3. Georgiev, K. and Zlatev, Z., Parallel sparse matrix algorithms for air pollution models, Parallel and Distributed Computing Practices, to appear.
4. Harrison, R. M., Zlatev, Z. and Ottley, C. J., A comparison of the predictions of an Eulerian atmospheric transport chemistry model with measurements over the North Sea, Atmospheric Environment, Vol. 28, 1994, 497-516.
5. Hov, Ø., Zlatev, Z., Berkowicz, R., Eliassen, A. and Prahm, L. P., Comparison of numerical techniques for use in air pollution models with non-linear chemical reactions , Atmospheric Environment, Vol. 23 (1988), 967-983.
6. Marchuk, G. I., Mathematical modeling for the problem of the environment, Studies in Mathematics and Applications, No. 16, North-Holland, Amsterdam, 1985.
7. McRae G. J., Goodin, W. R. and Seinfeld, J. H., Numerical solution of the atmospheric diffusion equations for chemically reacting flows, Journal of Computational Physics, Vol. 45 (1984), 1-42.
8. WEB-site for the Danish Eurlerian Model, http://www.dmu.dk/AtmosphericEnvironment/DEM.
9. Zlatev, Z., Application of predictor-corrector schemes with several correctors in solving air pollution problems, BIT, Vol. 24, 1984, 700-715.
10. Zlatev, Z., Computational methods for general sparse matrices, Kluwer Academic Publishers, Dordrecht-Boston-London, 1991.
11. Zlatev, Z., Computer treatment of large air pollution models, Kluwer Academic Publishers, Dordrecht-Boston-London, 1995.
12. Zlatev, Z., Christensen, J. and Eliassen, A., Studying high ozone concentrations by using the Danish Eulerian Model, Atmospheric Environment, Vol. 27A, 1993, 845-865.
13. Zlatev, Z., Christensen, C. and Hov,Ø., A Eulerian air pollution model for Europe with non-linear chemistry, Journal of Atmospheric Chemistry, Vol. 15 (1992), 1-37.
14. Zlatev, Z., Dimov, I. and Georgiev K., Studying long-range transport of air pollutants, Computational Science and Engineering, Vol. 1, No. 3, (1994), 45-52.
15. Zlatev, Z., Dimov, I. and Georgiev K., Three-dimensional version of the Danish Eulerian Model, Zeitschrift für Angewandte Mathematik und Mechanik, Vol. 76, S4, (1996), 473-476.
16. Zlatev, Z., Fenger, J. and Mortensen, L., Relationships between emission sources and excess ozone concentrations, Computers and Mathematics with Applications, Vol. 32, No. 11 (1996), 101-123.

This article was processed using the LaTeX macro package with LLNCS style