Application of sparse matrix techniques in the chemical part of a large air pollution model *

Tz. Ostromsky† Z. Zlatev ‡

Abstract

Large-scale air pollution models are represented by large systems of partial differential equations. After some discretization and splitting procedures these are transformed into several huge systems of ODE's, which are to be handled numerically during many time-steps. The use of both fast numerical methods and high speed computers is crucial for such models.

The chemical part of the model is normally the most time-consuming module. It consists of a huge number of **stiff** ODE systems, which can be treated as parallel tasks by using different partitioning procedures. One can also try to exploit the sparsity of the Jacobian matrix in order to improve further the efficiency of the computations. A special sparsity algorithm for the chemical part of a large air pollution model is described and tested in this paper.

AMS Subject Classifications: 65L05

Keywords: air pollution models, systems of PDE's, systems of ODE's, stiffness, partitioning, accuracy, sparsity

---

*This research was partially supported by grants from NATO (ENVIR.CGR.930449 and OUTR. CGR.960312), the Bulgarian Ministry of Education, Science and Technology (I-505/95 and MU-MM-02/96), and the European Union (WEPTEL Esprit 22727 and EUROAIR Esprit 24618). A grant from the Danish Natural Sciences Research Council gave us access to all Danish supercomputers.

†Central Laboratory for Parallel Information Processing, Bulgarian Academy of Sciences, Acad. G.Bonchev str., bl. 25-A, 1113 Sofia, Bulgaria; e-mail: ceco@iscbg.acad.bg

‡National Environmental Research Institute, Department of Atmospheric Environment Frederiksborgvej 399 P. O. Box 358, DK-4000 Roskilde, Denmark; e-mail: luzz@sun2.dmu.dk, Web site: http://www.dmu.dk/atmosphericenvironment/staff/zlatev.htm

1 Statement of the problem

An air pollution model is normally expressed mathematically by a system of partial differential equations:

$$\begin{aligned}\frac{\partial c_s}{\partial t} &= -\frac{\partial(uc_s)}{\partial x} - \frac{\partial(vc_s)}{\partial y} - \frac{\partial(wc_s)}{\partial z} \\ &+ \frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) \\ &+ E_s + Q_s(c_1, c_2, \dots c_q) - (k_{1s} + k_{2s})c_s\ , \quad s = 1, 2, \dots q\ .\end{aligned} \tag{1}$$

The number q of equations in this system is equal to the number of chemical species that are studied by the model. This number varies from 10 to 168 in the experiments described in [9]. The other quantities involved in the model can be described as follows (see again [9] for more details):

- the unknowns c_s are concentrations of the chemical species,
- u, v and w are the components of the wind along the coordinate axes,
- K_x, K_y and K_z are diffusion coefficients,
- the emissions in the space domain are represented by the functions E_s,
- k_{1s} and k_{2s} are coefficients of dry and wet deposition respectively ($s = 1, \dots q$),
- the chemical reactions between species are described by the non-linear functions $Q_s(c_1, c_2, \dots c_q)$, where $s = 1, 2, \dots, q$ (the condensed CBM IV scheme that was proposed by [4], see also [9], is the particular chemical scheme which will be used in this paper).

It is very difficult to treat directly the system (1). Therefore, some kind of splitting is to be used. Splitting according to the major physical processes is very popular; see, for example, [5], [6] and [9]. Such splitting procedures lead often to five sub-models which are to be treated cyclicly at every time-step ([9]). These sub-models are describing the horizontal advection (2), the horizontal diffusion (3), the chemical reactions (4) including the emissions, the deposition (5) and the vertical exchange (6).

$$\frac{\partial c_s^{(1)}}{\partial t} = -\frac{\partial(uc_s^{(1)})}{\partial x} - \frac{\partial(vc_s^{(1)})}{\partial y} \tag{2}$$

$$\frac{\partial c_s^{(2)}}{\partial t} = \frac{\partial}{\partial x}\left(K_x \frac{\partial c_s^{(2)}}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s^{(2)}}{\partial y}\right) \tag{3}$$

$$\frac{\partial c_s^{(3)}}{\partial t} = E_s + Q_s(c_1^{(3)}, c_2^{(3)}, \dots c_q^{(3)}) \tag{4}$$

$$\frac{\partial c_s^{(4)}}{\partial t} = -(k_{1s}+k_{2s})c_s^{(4)} \tag{5}$$

$$\frac{\partial c_s^{(5)}}{\partial t} = -\frac{\partial (w c_s^{(5)})}{\partial z} + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s^{(5)}}{\partial z}\right) \tag{6}$$

Five large systems of ordinary differential equations can be obtained from the sub-models (2)-(5) by applying any space discretization method:

$$dg^{(i)}/dt = f^{(i)}(t, g^{(i)}) \ , \quad g^{(i)} \in \mathcal{R}^{N_x \times N_y \times N_z \times N_s} \ , \quad f^{(i)} \in \mathcal{R}^{N_x \times N_y \times N_z \times N_s} \ , \tag{7}$$

where N_x, N_y and N_z are the numbers of grid-points on the grid-lines parallel to the coordinate axes and $N_s = q$ is the number of chemical species involved in the model. The functions $g^{(i)}$, $i = 1, 2, 3, 4, 5$, are vector-functions whose components are approximations to the concentrations at the grid-points, while the right-hand-side vectors $f^{(i)}$ depend on the particular discretization methods applied to the corresponding sub-models (2)-(6) (see [9]).

The number of equations in any of the five ODE systems in (7) is equal to the product of the number of grid-points and the number of chemical species studied by the model. Thus, if $N_x = 96$, $N_y = 96$, $N_z = 10$ and $N_s = 35$, then every ODE system contains 3 225 600 equations. Furthermore, the five ODE systems are to be treated numerically during many time-steps (typically several thousand time-steps are needed). This shows that the numerical treatment of large-scale air pollution models leads to very big computational problems. This is why both fast numerical algorithms and high speed computers are to be used.

The chemical sub-model, which is usually very stiff, is the most time-consuming part of the model. Therefore, one should be very careful when numerical algorithms for this sub-model are selected. The algorithms chosen must be fast. Furthermore, these must be efficient when high speed computers are used. The difficulties that have to be overcome when the chemical sub-model is treated are discussed in [1] and [7], and we shall not repeat this discussion here. It is important, however, to emphasize the fact that the chemical sub-model consists of $N_x \times N_y \times N_z$ *independent* ODE systems. Each of these systems contains $N_s = q$ equations and can be rewritten as

$$dg/dt = f(t, g) \ , \quad g \in \mathcal{R}^{N_s} \ , \quad f \in \mathcal{R}^{N_s} \ , \tag{8}$$

where g is a vector whose components are approximations to the concentrations at a given grid-point, while the right-hand-side vector f depends on the chemical mechanism which is used in the model (1). It is clear now that (8) can be considered (instead of the whole chemical sub-model) in the attempts to facilitate the work that is to be carried out in order to find a good numerical method for the chemical system. The Backward Euler Formula (together with some modification of the well-known Newton iterative procedure) is often used to solve numerically (8); see [1]. The straight-forward application of this numerical method leads to the solution of many systems of linear algebraic equations of the form:

$$A_{n+1}^{[0]} \triangle g_{n+1}^{[i]} = -g_{n+1}^{[i-1]} + g_n + \triangle t f(t_{n+1}, g_{n+1}^{[i-1]}), \tag{9}$$

where $g_{n+1}^{[i]}$ is an approximation to the unknown vector g at time-step $n+1$ and at iteration i, g_n the approximation calculated at the previous time-step n, $\triangle t$ is the time-stepsize and $A_{n+1}^{[0]}$ is equal to $I - \triangle t J_{n+1}^{[0]}$; $J_{n+1}^{[0]}$ being the Jacobian matrix of function f calculated at the beginning of the Newton iterative procedure at time-step $n+1$.

A partitioning procedure can be applied in an attempt to reduce the computational work needed to solve (9). The same partitioning as in [1] and [7] will be used in this paper. The application of this procedure is equivalent to the replacement of (9) with

$$S_{n+1}^{[0]} \triangle z_{n+1}^{[i]} = -z_{n+1}^{[i-1]} + z_n + \triangle t f(t_{n+1}, z_{n+1}^{[i-1]}), \tag{10}$$

where $S_{n+1}^{[0]}$ is a block-diagonal matrix. In the particular partitioning used, matrix $S_n^{[0]}$, $n = 1, 2, \ldots, N$, contains only one large (13×13) block. The following species are gathered in this block: NO, NO_2, O_3, $O^{(1)}D$, OH, HO_2, NO_3, N_2O_5, HNO_2, PAN, C_2O_3, $PHEN$ and PHO. All other species are forming small (1×1) blocks.

The use of a partitioned version of the Backward Euler Formula has several advantages: (i) it is not necessary to calculate all non-zero elements of the matrices $A_n^{[0]}$ (it is sufficient to calculate the non-zero elements of the matrices $S_n^{[0]}$), (ii) one (13×13) matrix only has to be factorized at time-step $n = 1, 2, \ldots, N$ instead of factorizing a (35×35) matrix, and (iii) (13×13) systems of linear algebraic equations are to be solved during the Newton iterations instead of solving (35×35) systems. It must be mentioned, however, that these advantages lead to a reduction of the computing time per iteration. The total computing time will also be reduced only if the convergence rate of the Newton iterative process applied to the partitioned version of the Backward Euler Formula is sufficiently fast. The numerical results presented in [1] and [7] show that this is true for the particular class of problems arising in the chemical sub-models of large air pollution modes. Several theorems proved in [10] explain why this is so.

Many people use partitioning to get parallel computations (see [**2**]). This will be efficient only if the blocks are well-balanced, which is not true for the chemical problems arising in air pollution modelling; see the paragraph after formula (10). However, parallelism on this level is not needed when large air pollution model are handled on parallel machines, because there are a lot of well-balanced and relatively big parallel tasks. Indeed, all chemical systems, found after the discretization and splitting procedures applied to (1), are independent tasks.

Some elements in the diagonal blocks of the matrices $S_n^{[0]}$, $n = 1, 2, \ldots, N$, are equal to zero; in the particular partitioning used here this is relevant only for the (13×13) block. One can try to exploit the fact that there are many zero elements by applying some sparse matrix technique. The development of a special sparse matrix technique, which can successfully be applied for the partitioned version of the Backward Euler Formula when the diagonal blocks are relatively small matrices, will be discussed in the following part of this paper.

2 Exploiting the sparsity

It may be worthwhile to apply some sparse matrix technique in the treatment of (10). However, if (10) arises from the chemical part of large air pollution models, then the ODE systems corresponding to the diagonal blocks of $S_n^{[0]}$, $n = 1, 2, \ldots, N$, are normally not large (the difficulties are caused both by the fact that the number of ODE systems, which are to be handled numerically at every chemical time-step, is very large and also by the fact that the number of chemical time-steps is very large). In the application discussed in the previous section, the largest block of $S_n^{[0]}$ contains 13 equations. Therefore, it is difficult to apply general sparse matrix techniques, as those discussed in [3] and [8]. However, special sparse matrix techniques may sometimes lead to an improvement of the performance. It will be explained in this section (i) why special sparse matrix techniques are to be developed and (ii) the main principles used in the development of a special sparse matrix technique.

2.1 Difficulties when general sparse techniques are used

The traditional sparse matrix tools, which are used in the treatment of large and sparse matrices (see [3] and [8]), are based on two important principles:

- computations are carried out only with the non-zero elements

and

- only the non-zero elements are stored.

Both the number of floating arithmetic operations and the storage needed are reduced in this way. However, one has to pay something for this reduction. The price that has to be paid is:

1. Indirect addressing has normally to be used.
2. Pivoting has to be applied in order both to preserve the stability of the computations and to obtain a factorized matrix which is as sparse as possible.
3. Many short loops are to be performed (both during the factorization and during the back substitution).
4. Many integer and logical operations are to be carried out when the sparse storage scheme is updated during the calculations.

If the matrix is sufficiently large (say, of order several thousands or greater), then the amount of extra operations caused by the four disadvantages listed above is rather small compared with the savings due to the fact that one works only with non-zero elements. The situation changes dramatically when the matrix is small and the general sparse matrix techniques, as those described in [3] and [8], become inefficient. Therefore, one must try to remove the drawbacks listed above.

2.2 Development of a special sparse matrix technique

Pivotal strategies are used both in order to preserve the stability of the computations and to keep the matrix which is transformed as sparse as possible during the factorization process. The pivotal strategies used in sparse matrix factorizations for general matrices, Markowitz pivotal interchanges and their modifications (see [8]), are very efficient for large matrices, but they become rather expensive when the factorized matrix is small. Therefore, the pivotal strategy must be improved. This has been achieved by removing the requirement to carry out pivoting for achieving numerical stability. The matrices that appear in our problems are general. Therefore, removing the pivotal interchanges for keeping the numerical stability may sometimes cause problems. However, the problems will normally be detected (because the Newton iterative procedure will not converge). The code will then automatically reduce the stepsize, which will lead to an improvement of the numerical stability of the computations during the *LU* factorization. On the other hand, removing pivoting for numerical stability has some advantages, which allow us to improve the performance of the code very considerably.

1. Pivoting for preserving the sparsity during the factorization can still be carried out. Moreover, all pivotal interchanges can be performed before the actual computations (and one can reserve locations for all non-zero elements, fill-ins, in the proper places of the sparse arrays). This means that a static storage scheme can be applied (instead of the dynamic storage schemes which are commonly used in connection with general sparse matrices).

2. No indirect addressing is needed, because all locations in the sparse arrays are fixed in the static storage scheme, and one can refer in the code directly to the actual locations where the non-zero elements are stored.

3. No extra integer arrays are needed. This together with the fact that a static storage scheme is used implies that no extra integer and logical operations are needed.

Moreover, the fact that the matrices that are factorized are small can be exploited to unroll all small loops. In this way the overhead for starting many small loops is removed. In fact, both the subroutine that calculates the factorization and the subroutine that performs the back substitution are completely loop-free codes.

The discussion given above shows that all drawbacks listed in the previous paragraph are removed. Of course, something has to be paid for these improvements. The fact that no pivoting for numerical stability is carried out has already been mentioned. The fact that the matrices are small has also been exploited (in order to remove the small loops and to remove all integer arrays). This means that the code is written for the particular chemical scheme used in the code. If the chemical scheme is changed, then the code has to be rewritten. Normally, it is not desirable to prepare such special codes. However, the reduction of the computing time spent in the chemical sub-model, which is the most time-consuming part of a large air pollution model, is a very important requirement when large air pollution models are handled numerically.

3 Numerical results

The experiments presented in this section cover a time period of 42 hours (starting at 6 o'clock in the morning and finishing at 24 o'clock in the next day). This period includes two sun-rise and two sun-set periods. The presence of such periods is very important in the attempt to check if the photochemical reactions are properly treated by the model, because quick changes of the concentrations of some species take place during sun-rises and sun-sets. It is also important to vary the temperature; temperature variations are simulated by a cosine function.

3.1 Selecting typical scenarios

The performance of the new algorithms must be tested for some typical situations; situations which appear often in practice. Six scenarios have been used to create such typical situations. These six scenarios have also been used in [1]. Different starting concentrations have been used with and without emissions in the six scenarios. A qualitative characterization of the scenarios, which have been selected by us, is given in Table 1.

Scenario	Short description
1	Small starting concentrations (remote rural areas)
2	Medium starting concentrations (areas close to big cities)
3	Large starting concentrations (big cities)
4	As Scenario 1 but also emissions are added
5	As Scenario 2 but also emissions are added
6	As Scenario 3 but also emissions are added

Table 1
Qualitative description of the six scenarios used in the experiments

3.2 The algorithms compared

Three algorithms have been tested during the experiments. The special sparse matrix technique described in the previous section has been compared with the corresponding partitioned algorithm in which a dense matrix technique is applied (i.e. no attempt to exploit the sparsity is made) and with the classical (non-partitioned) Backward Euler Formula. Both the computing times and the accuracy achieved have been compared.

3.3 Computing times

The six scenarios were first run with a starting time-stepsize $\triangle t = 30$ sec. The computing times (in seconds) obtained for the six scenarios by the different algorithms used in this study are given in Table 2. The scenarios were also run with a time stepsize $\triangle t = 1$ sec. (see Table 3), which resulted, roughly speaking, in an increase of all computing times by a factor of 30.

All experiments have been run on a SUN work-station with a 170 MHz processor. All subroutines have been compiled by using option *"-fast -O4"* of the SUN Fortran 77 compiler. No other jobs have been run on the work-station during the experiments.

Scenario	Non-partitioned Euler	Partitioned dense	Partitioned sparse
1	0.52	0.39	0.32
2	0.53	0.39	0.32
3	0.55	0.37	0.31
4	0.61	0.39	0.32
5	0.62	0.41	0.33
6	0.55	0.37	0.32

Table 2
Computing times obtained by running three numerical algorithms with $\triangle t = 30$ sec. on a SUN work-station.

Scenario	Non-partitioned Euler	Partitioned dense	Partitioned sparse
1	15.11	10.09	9.21
2	15.10	10.08	9.22
3	15.27	10.28	9.34
4	15.67	10.29	9.33
5	15.12	10.08	9.22
6	15.28	10.40	9.41

Table 3
Computing times obtained by running three numerical algorithms with $\triangle t = 1$ sec. on a SUN work-station.

3.4 The accuracy achieved with the different algorithms

One must also check the accuracy of the different algorithms. No pivoting for numerical stability is carried out in the special sparse matrix technique applied to the partitioned version of the Backward Euler Formula. It has been mentioned that this may, potentially, cause difficulties. However, the results given in the next two tables, Table 4 and Table 5, indicate that the accuracy achieved when the special sparse matrix technique is used is comparable with the accuracy achieved when the other two algorithms are used.

Scenario	Non-partitioned Euler	Partitioned dense	Partitioned sparse
1	1.78E-03	1.69E-3	1.69E-3
2	1.08E-02	9.76E-3	9.79E-3
3	1.03E-02	1.03E-2	1.09E-2
4	6.01E-03	6.03E-3	6.02E-3
5	1.85E-03	2.03E-3	2.01E-3
6	7.50E-03	8.03E-3	7.96E-3

Table 4
Accuracy of the nitrogen dioxide concentrations obtained by running three numerical algorithms with $\triangle t = 30$ sec. on a SUN work-station.

Scenario	Non-partitioned Euler	Partitioned dense	Partitioned sparse
1	3.30E-05	3.28E-5	3.28E-5
2	2.88E-04	2.88E-4	2.88E-4
3	2.56E-04	5.49E-4	5.49E-4
4	6.02E-05	3.70E-5	3.70E-5
5	6.22E-05	6.25E-5	6.25E-5
6	1.63E-04	5.00E-4	5.00E-4

Table 5
Accuracy of the nitrogen dioxide concentrations obtained by running three numerical algorithms with $\triangle t = 1$ sec. on a SUN work-station.

4 Conclusions and plans for the future work

Two main conclusions can be drawn from the numerical results, presented in in the previous section (see Table 2 - Table 5).

- The special sparse matrix technique described in Section 2 is faster than both the classical (non-partitioned) Backward Euler Formula and the dense version of the partitioned Backward Euler Formula. This is true both when the stepsize is rather large and when the stepsize is very small; see Table 2 and Table 3.

- The fact that no pivoting for numerical stability is carried out when the special sparse matrix technique is applied in the partitioned Backward Euler Formula has no influence on the accuracy achieved (it is of the same magnitude as for the other two methods used in our experiments). Again, this is true both when the stepsize is rather large and when the stepsize is very small; see Table 4 and Table 5.

The results can easily be extended to parallel computers. There are a lot of parallel tasks. The chemical species at every grid-point form an independent task. Thus the number of parallel tasks is equal to the number of grid-points. There are 92160 parallel tasks when a $(96 \times 96 \times 10)$ grid is specified. The situation is slightly more complicated when vector processors are used. However, also here can high efficiency be achieved by rearranging the computations across the grid-points in order to obtain long loops (see [9]). Tests on parallel and vector machines will be carried out in the near future.

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, to appear.

[2] K. Burrage, Parallel and sequential methods for ordinary differential equations, Oxford University Press, Oxford-New York, 1995.

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

[4] M. W. Gery, G. Z. Whitten, J. P. Killus and M. C. Dodge, A photochemical kinetics mechanism for urban and regional modeling, J. Geophys. Res., **94**, 1989, 12925-12956.

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

[6] G. J. McRae, W. R. Goodin and J. H. Seinfeld, Numerical solution of the atmospheric diffusion equations for chemically reacting flows, J. Comp. Phys., **45,** 1984, 1-42.

[7] S. Skelboe and Z. Zlatev, Exploiting the natural partitioning in the numerical solution of ODE systems arising in atmospheric chemistry, In: "Numerical Analysis and Its Applications" (L. Vulkov, J. Wasniewski and P. Yalamov, eds.), pp. 458-465, Springer, Berlin, 1997.

[8] Z. Zlatev, Computational methods for general sparse matrices, Kluwer Academic Publishers, Dordrecht-Toronto-London, 1991.

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

[10] Z. Zlatev, Partitioning ODE systems with an application to air pollution models, Internal report. National Environmental Research Institute, Department of Atmospheric Environment, Frederiksborgvej 399, P. O. Box 399, DK-4000 Roskilde, Denmark, 1997.