Journal of Computational Physics **165,** 496–510 (2000)
doi:10.1006/jcph.2000.6622, available online at http://www.idealibrary.com on IDEAL®

Adjoint Implementation of Rosenbrock Methods Applied to Variational Data Assimilation Problems

Dacian Daescu,[*] Gregory R. Carmichael,[†] and Adrian Sandu[‡]

[*]*Program in Applied Mathematical and Computational Sciences, The University of Iowa;* [†]*Center for Global and Regional Environmental Research and The Department of Chemical and Biochemical Engineering, The University of Iowa; and* [‡]*Department of Computer Science, Michigan Technological University*
E-mail: ddaescu@math.uiowa.edu; gcarmich@icaen.uiowa.edu; asandu@mtu.edu

Received September 24, 1999; revised August 24, 2000

In the past decade the variational method has been successfully applied in data assimilation problems for atmospheric chemistry models. In 4D-var data assimilation, a minimization algorithm is used to find the set of control variables which minimizes the weighted least squares distance between model predictions and observations over the assimilation window. Using the adjoint method, the gradient of the cost function can be computed fast, at the expense of few function evaluations, making the optimization process very efficient. For large-scale models, the high storage requirements and the difficulty of implementing the adjoint code when sophisticated integrators are used to solve the stiff chemistry make the assimilation a very intensive computational process. If the sparse structure of the chemical models is carefully exploited, Rosenbrock methods have been proved to be reliable chemistry solvers because of their outstanding stability properties and conservation of the linear invariants of the system. In this paper we present an efficient implementation of the adjoint code for the Rosenbrock type methods, which can reduce the storage requirements of the forward model and is suitable for automatization. The adjoint code is completely generated using symbolic preprocessing and automatic differentiation tools which allow flexibility and require minimal user intervention. © 2000 Academic Press

Key Words: adjoint model; stiff equations; automatic differentiation; optimization.

1. INTRODUCTION

Consider a 3D atmospheric transport–chemistry model given by the system of differential equations

$$\frac{\partial}{\partial t}\mathrm{c}_i = -\nabla\cdot(\mathbf{u}\mathrm{c}_i) + \nabla\cdot(\mathbf{K}\cdot\nabla\mathrm{c}_i) + f_i(\mathrm{c}) + E_i, \quad i = \overline{1, S}. \tag{1.1}$$

The initial condition is $\mathrm{c}(t_0) = \mathrm{c}_0$ and appropriate boundary values are prescribed. The solution $\mathrm{c}(t, x, y, z) \in R^S$ of problem (1.1) represents the concentration vector of the chemical species in the model, $\mathbf{u}$ is the wind field, and $\mathbf{K}$ is the eddy diffusivity tensor; the chemical reactions are modeled by the nonlinear stiff terms $f_i(\mathrm{c}) = P_i(\mathrm{c}) - D_i(\mathrm{c})\mathrm{c}_i$, with $P_i(\mathrm{c}), D_i(\mathrm{c})$ the chemical production and destruction terms; E_i represents the source term, and depositions are modeled as a boundary condition at earth's surface,

$$-\mathbf{n}_h \cdot (\mathbf{K} \cdot \nabla \mathrm{c}_i) = Q_i - v_i \mathrm{c}_i,$$

with $\mathbf{n}_h$ the inward vector normal to the earth's surface, Q_i and v_i the surface emission rate and deposition velocity of species i, respectively. Space and time dependence is assumed for all terms, but for simplicity the explicit notation is omitted. A complete model description is given in [6]. We will refer to problem (1.1) as the forward model and $\mathrm{c}(t), t \in [t_0, T]$ will represent the "forward trajectory." The forward trajectory is determined by the values of the parameters in (1.1), and typical choices for the set of control variables in data assimilation are the boundary values, initial concentrations, emissions, and deposition rates. When the identifying parameters are distributed in the space–time domain, instabilities and nonuniqueness problems may appear for the inverse modeling techniques (in addition to those given by the nonlinear structure of the model). A general framework for adjoint parameter estimation and a discussion of the parameter identifiability problem is presented in [23]. For the purpose of this paper, we consider the 4D variational data assimilation problem associated to (1.1) with the set of control variables given by the initial state of the model, c_0. Under suitable assumptions, problem (1.1) has a unique solution, and we can view this solution as a function of the initial conditions, $\mathrm{c} = \mathrm{c}(t, x, y, z, \mathrm{c}_0)$.

If space discretization is applied to problem (1.1) on a grid (N_x, N_y, N_z), the resulting ODE system of dimension $N = S \times N_x \times N_y \times N_z$ is

$$\begin{aligned} \frac{d\mathrm{c}}{dt} &= F_A(c) + F_D(c) + F_R(c) \\ \mathrm{c}(t_0) &= \mathrm{c}_0, \end{aligned} \tag{1.2}$$

where F_A represents the advection and horizontal diffusion, F_D is the vertical diffusion, which may introduce additional stiffness, and the reaction terms are given by F_R. The source and sink terms may be included in F_D [6] or in the chemistry operator F_R [3, 33]. We assume that a previous analysis provides a "background estimate" c^b of c_0 with the error covariance matrix $\mathbf{B}$, and measurements c_k^o, $k = 1, m$ of the concentrations at moments t_k are scattered over the interval $[t_0, T]$. The errors in measurements and model representativeness are given by the covariance matrices $\mathbf{R}_k$, $k = 1, m$. The covariance matrices $\mathbf{B}$ and $\mathbf{R}_k$ are symmetric and positive definite (if they do not contain null or infinite variances, or perfect correlations, [32]) such that $\mathbf{B}^{-1}, \mathbf{R}_k^{-1}$ are well defined. In practice, $\mathbf{B}, \mathbf{R}_k$ are often taken diagonal, which corresponds to the assumption that there is no spatial and chemical correlation in the background errors, and measurement and model errors are uncorrelated in space and time. The 4D-var data assimilation finds an initial state c_0 that minimizes the distance between the model predictions and observations expressed by the cost function:

$$\mathcal{F}(\mathrm{c}_0) = \frac{1}{2}(\mathrm{c}_0 - \mathrm{c}^b)^T \mathbf{B}^{-1} (\mathrm{c}_0 - \mathrm{c}^b) + \frac{1}{2} \sum_{k=1}^{m} \left(\mathrm{c}_k - \mathrm{c}_k^o\right)^T \mathbf{R}_k^{-1} \left(\mathrm{c}_k - \mathrm{c}_k^o\right). \tag{1.3}$$

The quality of the assimilation results depends on several factors such as availability and spatial–temporal distribution of measurements, accuracy of the background estimate, length of the assimilation window, errors in measurements, and model representativeness. The existence of multiple local minima of the functional $\mathcal{F}$ may lead to ambiguous assimilation results. This analysis is beyond the goal of this paper, and details can be found in [10, 12, 18] and the references therein.

Most of the powerful optimization techniques require the evaluation of the gradient $\nabla_{c_0}\mathcal{F}$ of the cost function. In a comprehensive atmospheric chemistry model, the dimension of the vector c_0 can easily be of order 10^6, which makes the optimization a very expensive computational process.

In the variational approach, one computes the gradient of the functional $\mathcal{F}$ using the "adjoint method." Since for nonlinear problems (such as 1.1, 1.2) the corresponding adjoint equations depend on the forward trajectory, the computational cost and the complexity of the adjoint implementation are significantly increased compared with those of linear problems. The general theory of adjoint equations is described in [20, 21] and the derivation of the adjoint model for the continuous and discrete case is given in [12, 31]. Below we outline the basic ideas. The gradient of the cost function is

$$\nabla_{c_0}\mathcal{F}(\mathrm{c}_0) = \mathbf{B}^{-1}(\mathrm{c}_0 - \mathrm{c}^b) + \sum_{k=1}^{m} \left(\frac{\partial \mathrm{c}_k}{\partial \mathrm{c}_0}\right)^T \mathbf{R}_k^{-1}\left(\mathrm{c}_k - \mathrm{c}_k^o\right). \tag{1.4}$$

Using the chain rule in its transpose form $(\frac{\partial \mathrm{c}_k}{\partial \mathrm{c}_0})^T = (\frac{\partial \mathrm{c}_{k-1}}{\partial \mathrm{c}_0})^T (\frac{\partial \mathrm{c}_k}{\partial \mathrm{c}_{k-1}})^T$, we can deduce the algorithm to compute the necessary gradient:

STEP 1. Initialize *gradient* $= 0$

STEP 2. *for* $k = m, 1, -1$ *do*

$$\textit{gradient} = \left(\frac{\partial \mathrm{c}_k}{\partial \mathrm{c}_{k-1}}\right)^T \left[\mathbf{R}_k^{-1}\left(\mathrm{c}_k - \mathrm{c}_k^o\right) + \textit{gradient}\right]$$

STEP 3. *gradient* $= \mathbf{B}^{-1}(\mathrm{c}_0 - \mathrm{c}^b) +$ *gradient*

The main advantage of the adjoint method is that explicit computation of the Jacobian matrices $\frac{\partial \mathrm{c}_k}{\partial \mathrm{c}_{k-1}}$ is avoided and the matrix–vector products can be computed directly at Step 2. For the theory and actual implementation of the adjoint computations the reader should consult [14, 15, 22]. Because the algorithm described above requires the values of c_k in reverse order, these values need to be stored from a previous run or recomputed. Moreover, in practice the measurements are usually sparse and the value of c_k is obtained from c_{k-1} with a sequence of steps $\mathrm{c}_{k-1} \rightarrow \mathrm{c}_k^1 \rightarrow \cdots \rightarrow \mathrm{c}_k^s \rightarrow \mathrm{c}_k$. The computational trade-off is then between allocating a huge amount of memory to store the states of the system during the forward run, or frequent recomputations which increase the running time of the code. If an explicit numerical method is used to solve the stiff chemistry part of problem (1.2), then the "trajectory" from c_{k-1} to c_k may become very long, increasing the cost of the adjoint code. On the other hand, if an implicit method is used, then the adjoint computations may become complicated. Ideally one would like a method capable of taking large stepsize and an efficient adjoint implementation.

2. OPERATOR SPLITTING

A popular way to solve problem (1.2) is to use operator splitting, which has the advantage that processes such as advection, vertical diffusion, and chemical reactions can be treated with different numerical methods. In a second order accurate Strang splitting [30] approach with the time step $h = t_{n+1} - t_n$, the solution c_{n+1} is obtained from c_n as

$$c_{n+1} = \bar{F}_A\left(t_{n+1/2}, \frac{h}{2}\right)\bar{F}_D\left(t_{n+1/2}, \frac{h}{2}\right)\bar{F}_R(t_n, h)\bar{F}_D\left(t_n, \frac{h}{2}\right)\bar{F}_A\left(t_n, \frac{h}{2}\right)c_n, \qquad (2)$$

where the operators $\bar{F}$ are defined by the numerical method used to solve the corresponding processes. If $\bar{J}$ denotes the $N \times N$ Jacobian matrix associated to $\bar{F}$, the adjoint algorithm to compute the gradient (1.4) of the cost function requires products of the form $\bar{J}^T u$, with u an arbitrary seed vector. Because constructing the adjoint code for large systems by hand can be a frustrating process, automatic tools have been developed [14, 24]. Automatic implementation allows also for flexibility, such that if the model is modified, minimal user intervention is required.

Usually $\bar{F}_A$ is defined by an explicit method and may be nonlinear (e.g., if a flux-limiter is applied for positivity [17, 35]); $\bar{F}_D$ is linear, defined by a (semi-) implicit method. The products $\bar{J}_A^T u$ and $\bar{J}_D^T u$ can be then efficiently computed using an automatic adjoint compiler. Coleman *et al.* [8] present an "extended Jacobian" framework to exploit the sparsity of a finite difference scheme, which leads to efficient computations of the adjoint products when automatic differentiation is applied on the finite difference stencils.

The operator $\bar{F}_R$ is highly nonlinear, given by a stiff numerical method, and the computation of $\bar{J}_R^T u$ needs special consideration. A key element for the efficient implementation of the forward code is to exploit the sparse structure of the chemical model. Sparse computations must be then performed as well during the backward integration. Because the adjoint method requires several integrations of the direct model, the storage of (part of) the forward trajectory and the computation of the $(jacobian)^T \cdot vector$ products, the performance of the adjoint model is dominated by the implementation of the direct and adjoint method used in the chemistry integration, which takes in practice as much as 90% of the CPU time. Fisher and Lary [12] show the adjoint computations for the adaptive-timestep Bulirsch-Stoer method, and Elbern and Schmidt [10] use the adjoint model for a quasi-steady-state approximation (QSSA) scheme. In the next section, we present the adjoint formulas for a general 2-stage Rosenbrock method and an efficient implementation which is suitable for automatization. The L-stable method *ROS*2 we obtain as a particular case was applied for the chemistry integration in the forward 3D model LOTOS [3] in the context of various types of operator splitting and using approximate Jacobians (as a W-method). Extension to a general s-stage method [16] is straightforward.

3. ADJOINT COMPUTATIONS AND IMPLEMENTATION FOR A 2-STAGE ROSENBROCK METHOD

3.1. *Derivation of the Adjoint Formulas*

We consider now the problem

$$\frac{d\mathrm{c}}{d\mathrm{t}} = f(\mathrm{c})$$

$$\mathrm{c}(t_0) = \mathrm{c}_0 \tag{3.1}$$

with $\mathrm{c}(t), \mathrm{c}_0 \in R^n$ and $f : R^n \to R^n,\ f = (f_1, f_2, \dots f_n)^T$.

One step from t_0 to t_1 with $h = t_1 - t_0$ of a 2-stage Rosenbrock method as presented in [16] reads

$$\left(\frac{1}{\gamma_{11} h} I - J_0\right)\mathrm{k}_1 = f(\mathrm{c}_0) \tag{3.2}$$

$$\left(\frac{1}{\gamma_{22} h} I - J_0\right)\mathrm{k}_2 = f(\mathrm{c}_0 + \alpha \mathrm{k}_1) + \frac{\beta}{h}\mathrm{k}_1 \tag{3.3}$$

$$\mathrm{c}_1 = \mathrm{c}_0 + m_1 \mathrm{k}_1 + m_2 \mathrm{k}_2, \tag{3.4}$$

where J_0 is the Jacobian matrix of f evaluated at c_0, $J_0 = (\frac{\partial f_i}{\partial \mathrm{c}_j})_{ij}|_{\mathrm{c}=\mathrm{c}_0}$, and the coefficients $\gamma_{11}, \gamma_{22}, \alpha, \beta, m_1, m_2$ are chosen to obtain a desired order of consistency and numerical stability. Because the methods that require only one LU decomposition of $\frac{1}{\gamma_{ii} h} I - J_0$ per step are of special interest, we consider the case when $\gamma_{11} = \gamma_{22} = \gamma$.

For the adjoint computations from (3.2), (3.3) we have

$$\begin{aligned}\left(\frac{\partial \mathrm{k}_1}{\partial \mathrm{c}_0}\right)^T &= \left(J_0^T + \left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_1\right)^T\right)\left(\left(\frac{1}{\gamma h} I - J_0\right)^T\right)^{-1} \\ \left(\frac{\partial \mathrm{k}_2}{\partial \mathrm{c}_0}\right)^T &= \left(\left(I + \alpha \frac{\partial \mathrm{k}_1}{\partial \mathrm{c}_0}\right)^T J_1^T + \frac{\beta}{h}\left(\frac{\partial \mathrm{k}_1}{\partial \mathrm{c}_0}\right)^T + \left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_2\right)^T\right)\left(\left(\frac{1}{\gamma h} I - J_0\right)^T\right)^{-1},\end{aligned} \tag{3.5}$$

where J_1 is the Jacobian evaluated at $\mathrm{c}_0 + \alpha \mathrm{k}_1$, and the terms $(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_i), i = 1, 2,$ are $n \times n$ matrices whose j column is $(\frac{\partial J_0}{\partial \mathrm{c}_0^j})\mathrm{k}_i, i = 1, 2$. We want to stress here the fact that these matrices are not symmetric; we will return to the computation of these terms later. Using (3.4, 3.5), for an arbitrary seed vector $\mathrm{u} \in R^n$ we have

$$\begin{aligned}\left(\frac{\partial \mathrm{c}_1}{\partial \mathrm{c}_0}\right)^T \mathrm{u} = \mathrm{u} &+ m_1\left(J_0^T + \left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_1\right)^T\right)\left(\left(\frac{1}{\gamma h} I - J_0\right)^T\right)^{-1}\mathrm{u} \\ &+ m_2\left(\left(I + \alpha\left(\frac{\partial \mathrm{k}_1}{\partial \mathrm{c}_0}\right)^T\right)J_1^T + \frac{\beta}{h}\left(\frac{\partial \mathrm{k}_1}{\partial \mathrm{c}_0}\right)^T \right. \\ &\left. + \left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_2\right)^T\right)\left(\left(\frac{1}{\gamma h} I - J_0\right)^T\right)^{-1}\mathrm{u}.\end{aligned}$$

To avoid frequent recomputations and to exploit the particular properties of the method, the order of the operations in the formula above become important. Below we present an efficient algorithm.

STEP 1. Solve for v the linear system $(\frac{1}{\gamma h} I - J_0)^T \mathrm{v} = \mathrm{u}$. Then,

$$\begin{aligned}\left(\frac{\partial \mathrm{c}_1}{\partial \mathrm{c}_0}\right)^T \mathrm{u} = \mathrm{u} &+ m_1\left(J_0^T + \left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_1\right)^T\right)\mathrm{v} + m_2 J_1^T \mathrm{v} \\ &+ m_2\left(\frac{\partial \mathrm{k}_1}{\partial \mathrm{c}_0}\right)^T\left(\alpha J_1^T + \frac{\beta}{h} I\right)\mathrm{v} + m_2\left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_2\right)^T \mathrm{v}.\end{aligned} \tag{3.6}$$

STEP 2. Compute $\omega = J_1^T(m_2\mathrm{v})$; $\omega_1 = \alpha\omega + \frac{m_2\beta}{h}\mathrm{v}$. Using (3.5) we get next:

STEP 3. Solve for θ the linear system $(\frac{1}{\gamma h}I - J_0)^T\theta = \omega_1$.
After replacing in (3.6), there results

$$\left(\frac{\partial \mathrm{c}_1}{\partial \mathrm{c}_0}\right)^T \mathrm{u} = \mathrm{u} + m_1\left(J_0^T + \left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_1\right)^T\right)\mathrm{v} + \omega$$
$$+ \left(J_0^T + \left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_1\right)^T\right)\theta + m_2\left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_2\right)^T \mathrm{v} \tag{3.7}$$

and after arranging the terms we obtain:

STEP 4. Compute

$$\left(\frac{\partial \mathrm{c}_1}{\partial \mathrm{c}_0}\right)^T \mathrm{u} = \mathrm{u} + \omega + J_0^T(m_1\mathrm{v} + \theta) + \left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_1\right)^T (m_1\mathrm{v} + \theta) + m_2\left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_2\right)^T \mathrm{v} \tag{3.8}$$

In formula (3.7) it appears that a routine to compute the product $J_0^T s$ (s a seed vector) must be called twice: first with the seed vector $m_1\mathrm{v}$, second with the seed vector θ. From (3.8) it is enough to call the routine once, with the seed vector $m_1\mathrm{v} + \theta$. The same observation is made for the products $(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_1)^T m_1\mathrm{v}$, $(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_1)^T\theta$. We now focus on the terms of the form $(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k})^T\mathrm{v}$ whose evaluation dominate the computational cost of the algorithm given by Steps 1–4. Here $\mathrm{k}, \mathrm{v} \in R^n$ are arbitrary constant vectors. For the i component we have

$$\left(\left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}\right)^T \mathrm{v}\right)_i = \left(\frac{\partial J_0}{\partial \mathrm{c}_0^i}\mathrm{k}\right)^T \mathrm{v} = \mathrm{k}^T\left(\frac{\partial\left(J_0^T\mathrm{v}\right)}{\partial \mathrm{c}_0^i}\right) = \left(\frac{\partial\left(J_0^T\mathrm{v}\right)}{\partial \mathrm{c}_0^i}\right)^T \mathrm{k}. \tag{3.9}$$

Consider now the function $g: R^n \to R^n$, $g(\mathrm{c}_0) = J_0^T\mathrm{v}$. Observe that the Jacobian matrix of g is symmetric. We have

$$g(\mathrm{c}_0) = \left(\sum_{l=1}^{n} J_{0,(l,1)}\mathrm{v}_l, \ldots, \sum_{l=1}^{n} J_{0,(l,n)}\mathrm{v}_l\right)^T,$$

which gives for the (i, j) entry in the Jacobian matrix

$$\frac{\partial g_i(\mathrm{c}_0)}{\partial \mathrm{c}_0^j} = \sum_{l=1}^{n}\left(\frac{\partial^2 f_l}{\partial \mathrm{c}_0^i \partial \mathrm{c}_0^j}\right)\mathrm{v}_l = \sum_{l=1}^{n} H_{f_l}(i, j)\mathrm{v}_l,$$

where H_{f_l} is the Hessian matrix of the function $f_l: R^n \to R$.
Thus $\frac{\partial g(\mathrm{c}_0)}{\partial \mathrm{c}_0} = \sum_{l=1}^{n} H_{f_l}\mathrm{v}_l$, so $\frac{\partial g(\mathrm{c}_0)}{\partial \mathrm{c}_0}$ is symmetric. Using (3.9) results in

$$\left(\left(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}\right)^T \mathrm{v}\right) = \left(\frac{\partial g(\mathrm{c}_0)}{\partial \mathrm{c}_0}\right)\mathrm{k}. \tag{3.10}$$

The symmetry of the Jacobian matrix of the function g used in relation (3.10) plays a significant role in the implementation of the adjoint code which we present next.

3.2. *Implementation of the Adjoint Code*

The forward integration of problem (3.1) using implicit methods together with the performance analysis is given in [27, 28, 34], proving that when the sparsity of the system is efficiently exploited Rosenbrock methods outperform traditional explicit methods like QSSA and CHEMEQ. Implementation is done in the symbolic kinetic preprocessor KPP environment [9], which generates the sparse matrix factorization LU required in (3.2, 3.3) with a minimal fill-in [26] and the routine to forward–backward solve the linear systems without indirect adressing. One step of the adjoint code (from t_1 to t_0) requires a forward run from t_0 to t_1 given by the formulas (3.2)–(3.4) followed by the pure adjoint computations given by Steps 1–4. It is important to notice that the LU decomposition accounts for most of the CPU time of the code, and there is no need to repeat it during the pure backward integration.

With the LU decomposition of $(\frac{1}{\gamma h}I - J_0)$ available from (3.2) Step 1 reads $U^T L^T \mathrm{v} = \mathrm{u}$. A new loop-free routine is generated by KPP for forward–backward solution of this system in sparse format, avoiding indirect addressing. The computational cost of Steps 1 and 3 can be then compared with the corresponding part from (3.2) and (3.3).

Step 2 requires evaluation of the product $J_1^T \mathrm{v}$, which is automatically computed by KPP using sparse multiplications. This introduces some extra work (J_1 is evaluated at $\mathrm{c}_0 + \alpha \mathrm{k}_1$), but its cost is relatively cheap. The efficiency of the adjoint code is then dominated by the implementation of Step 4, given by the formula (3.8). The computation of the terms of the form $(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k})^T \mathrm{v}$ in formula (3.8) appears to require the following order: forward automatic mode to compute $J_0\mathrm{k}$, and reverse mode to compute $(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k})^T \mathrm{v}$. Relation (3.10) can be used to rewrite (3.8) as

$$\left(\frac{\partial \mathrm{c}_1}{\partial \mathrm{c}_0}\right)^T \mathrm{u} = \mathrm{u} + \omega + g_1(\mathrm{c}_0) + \left(\frac{\partial g_1(\mathrm{c}_0)}{\partial \mathrm{c}_0}\right)\mathrm{k}_1 + m_2\left(\frac{\partial g_2(\mathrm{c}_0)}{\partial c_0}\right)\mathrm{k}_2, \tag{3.11}$$

with $g_1, g_2 : R^n \rightarrow R^n$, $g_1(\mathrm{c}_0) = J_0^T(m_1\mathrm{v} + \theta)$, $g_2(\mathrm{c}_0) = J_0^T \mathrm{v}$.

The functions g_1, g_2 are generated via KPP, taking full advantage of the sparsity of the Jacobian matrix J_0. In (3.11) we have then to compute the *Jacobian* · *vector* products for the functions g_1, g_2, which can be done by *forward* automatic differentiation [2, 14]. The cost is 2–3 times the cost of evaluating $g_1(\mathrm{c}_0)$, $g_2(\mathrm{c}_0)$ and remains low because of the sparse structure of J_0. This leads to a considerable saving in CPU time. By default, during the computation of *Jacobian* · *vector* products, forward automatic differentiation computes the value of the function. Automatic differentiation for g_1 then provides the value $g_1(\mathrm{c}_0)$, and there is no need to compute it separately. Last but not least, the computations related to g_1 and g_2 are independent, allowing parallel implementation.

4. PERFORMANCE AND VALIDATION OF THE ADJOINT ALGORITHM

The algorithm and implementation presented in Section 3 have the benefit that the adjoint part of the chemistry integration is generated completely automatically, taking full advantage of the sparsity of the system. This allows the user to move easily from one model to another and makes it very attractive compared with the handwritten codes whose construction for large models can be a difficult process. Moreover, because symbolic computations are used, rounding errors are avoided and the accuracy of the results goes up to the machine precision.

Implementation in the KPP context also has the advantage of avoiding the introduction of auxiliary adjoint variables, which has direct impact on the performance of the code both in terms of memory usage and CPU time.

In particular, we consider the 2nd order 2-stage Rosenbrock method *ROS*2 which is obtained from (3.2)–(3.4) by taking $\alpha = \frac{1}{\gamma}$, $\beta = -\frac{2}{\gamma}$, $m_1 = \frac{3}{2\gamma}$, $m_2 = \frac{1}{2\gamma}$. Choosing $\gamma = 1 \pm 1/\sqrt{2}$, the method is L-stable and the numerical experiments presented in this section were performed with $\gamma = 1 + 1/\sqrt{2}$. The superior stability, positivity, and conservation properties of this scheme are analyzed by Verwer *et al.* [34], who report good results in the context of various types of operator splitting, even when large fixed step sizes (10 to 20 min.) are used.

4.1. *The Box Model*

To test the performance of the implementation we consider first a box model for the problem (1.1). The chemistry part is based on the Carbon Bond Mechanism IV (CBM-IV [13]) with 32 chemical species involved in 70 thermal and 11 photolytic reactions. The data assimilation problem is set using the “twin experiments method,” with the background term dropped and the logarithmic form of (1.3). Taking the logarithm of the concentrations has the advantage that the positivity constraint is eliminated and scales the system. The minimization routine used is the Quasi-Newton limited memory L-BFGS algorithm [4, 5], anticipating extension to large-scale models. The initial concentrations follow the urban scenario as described in [27], with an initial concentration of 70 ppb for O3. Assimilation starts at the beginning of the third day (6:00 LT) over a period of 6 h, with measurements provided every 15 min. As the initial guess for the concentrations we chose the values at the beginning of the second day. The one-day period is introduced to allow the system to equilibrate. The integration is restarted every 15 min with a minimum stepsize of 1 s, simulating an operator-splitting environment. With the absolute and relative tolerances Atol $=$ 1 molecules, Rtol $= 0.01$, the number of intermediate steps within a 15-min interval ranges from 5 to 12, providing a relatively short forward trajectory. Two experiments were performed: in Run 1 measurements were provided for ozone only, and in Run 2 for ozone and NO_2. The results of the assimilation for O_3, NO_2, and NO are shown in Fig. 1. It can be seen that model predictions are highly improved even after the end of the assimilation window (12:00 LT), and introducing NO_2 measurements is of benefit not only for the NO_2 and NO analysis, but also for the O_3 analysis. However, because additional constraints are introduced, the number of iterations in the optimization is increased (Table I).

Alternatively, to compute the gradients we use the second-order central difference formula [1] with $\epsilon = (2.22 \times 10^{-16})^{1/3}$. Figure 2 (left Run 1, right Run 2) shows the relative and absolute differences between the computed gradients with respect to some important species in the model. Because in the context of stiff computations and a long trajectory, the roundoff errors can highly affect the accuracy of the difference schemes, we also show in Fig. 3 the corresponding results for assimilation with only one measurement, at T $=$ 6:15 LT, together with the computed gradients. On average, 8 to 10 significant digits of the gradients are matched (when $\nabla\mathcal{F}$ is nearly zero, the relative error size in the gradient approximation for difference schemes may become very large [1]).

For consistency with the implementation for large-scale models, where the storage of the entire trajectory is not a realistic option, a checkpointing scheme is applied for the gradient computations. First, a full forward run is used to store the states of the system

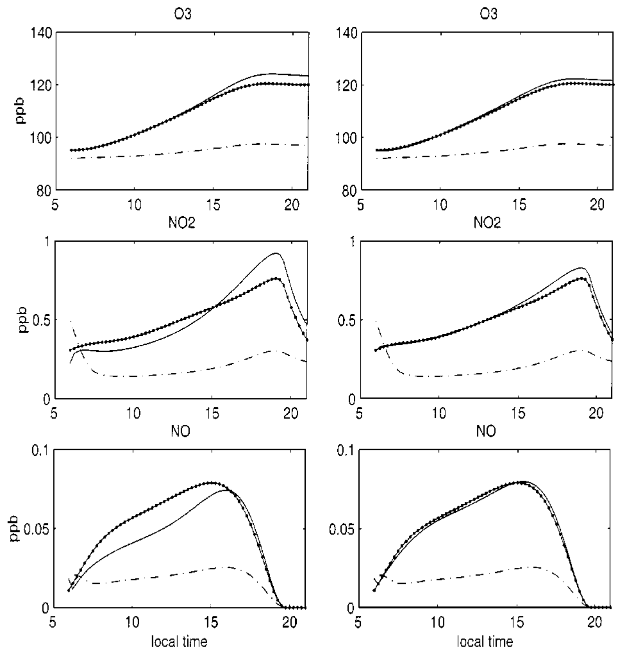

FIG. 1. Assimilation takes place from 6 to 12 LT. Measurements are provided every 15 min for O_3 only in Run 1 (left), and for O_3 and NO_2 in Run 2 (right). Solid line with dots = reference run; solid line = assimilation result; dotted line = first guess.

at the measurement moments. Second, during the backward integration, a full forward run stores the trajectory between measurements (see Section 1) and information about the stepsize used. Third, the computations given by Steps 1–4 in Section 3 are performed (pure backward integration). Observe that the numerical values of k_1, k_2 are still required. These recomputations may be avoided by storing all k_1 during the second forward run, but this will double its storage requirements. We prefer to repeat (3.2) to compute k_1, then avoid (3.3) by setting $m_2k_2 = c_1 - c_0 - m_1k_1$ in (3.11). The technical report of the optimization process is outlined in Table I (Run 1, 2). We denote by KPP-AD the implementation of the adjoint code using the KPP generated adjoint routines and forward automatic differentiation. It can

TABLE I

Performance of the Optimization Process and the Adjoint Code[a]

				$\sim$cpu/iter		$\sim$cpu$(\mathcal{F}+\nabla\mathcal{F})$/ cpu$(\mathcal{F})$		
Run	Iter.	$(\mathcal{F}, \nabla\mathcal{F})$-eval.	$\sim$cpu$(\mathcal{F})$	KPP–AD	TAMC	KPP–AD[b]	TAMC[b]	$\mathcal{F}_{\text{init}}/\mathcal{F}_{\text{end}}$
1.	27	32	0.017	0.35	0.42	3.66	7.71	1.e5
2.	32	38	0.017	0.36	0.44	3.71	7.75	2.8e3
3.[c]	34	34	2.05	8.32	16.6	3.65	7.69	1.e2

[a] All the computations were done on a HP-UX B.10.20 A 9000/778 machine with level 2 optimization. The CPU time is in seconds.

[b] The time to read–write data to files is not considered for these ratios.

[c] Simplified recomputations for the advection–diffusion part are taken during the backward integration (see Fig. 5).

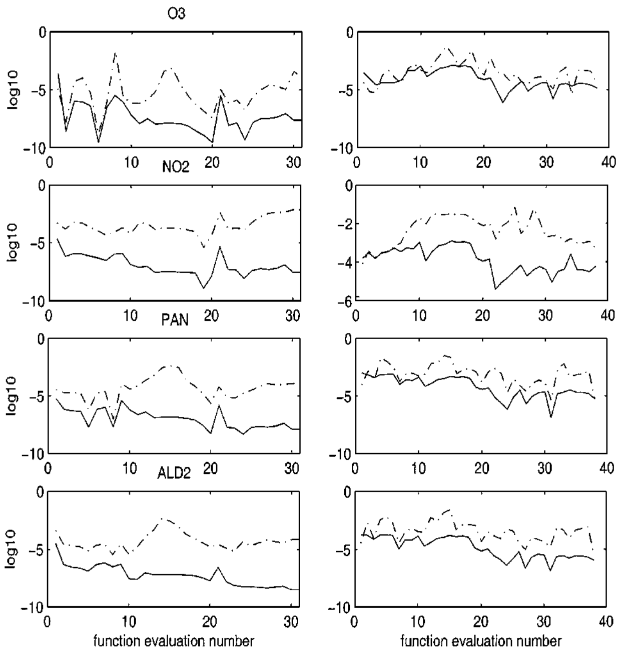

FIG. 2. Absolute (solid line) and relative (dotted line) differences between the computed gradients using the central difference formula and automatic adjoint computations during the optimization process. Left—Run 1, right—Run 2.

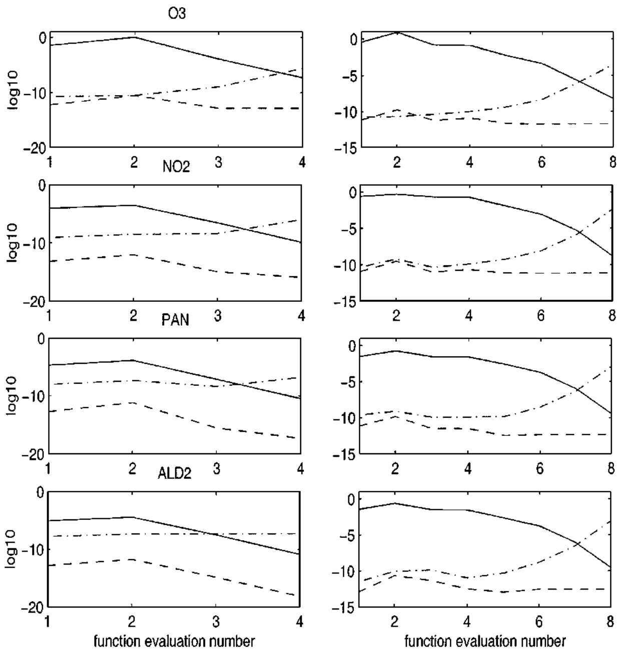

FIG. 3. Absolute (dashed line) and relative (dotted line) differences between the computed gradients (solid line) using the central difference formula and automatic adjoint computations. Left—Run 1, right—Run 2. To reduce the roundoff errors a short assimilation interval (15 min) is considered.

be seen that the average ratio between the CPU time required to compute the gradient (and cost function value) and the CPU time of a forward run is about 3.7, which gives an average ratio *cpu(pure backward integration* $+$ *(3.2))*$/$*cpu(forward integration)* ≈ 1.7. This makes our implementation very efficient.

As an alternative way for automatic adjoint code generation, we applied the adjoint model compiler TAMC [14, 15] to (3.2)–(3.4) using the same checkpointing scheme. In particular, we noticed that the adjoint compiler fails to generate an efficient adjoint code for the sparse chemistry computations. The timing results included in Table I (TAMC) were obtained with significant user intervention to reduce the frequent recomputations generated by the adjoint compiler.

4.2. *Application to a 1-D Problem*

We consider now a one-dimensional horizontal test problem corresponding to (1.1). The wind field and the diffusion coefficient are taken constant, $\mathbf{u} = 10$ km/h (left-to-right), $K = 10^{-3}$ km^2/sec. Second-order Strang splitting is applied,

$$c(t_{n+1}) = \bar{F}_A\left(t_{n+1/2}, \frac{h}{2}\right)\bar{F}_R(t_n, h)\bar{F}_A\left(t_n, \frac{h}{2}\right)c(t_n),$$

with a splitting interval $t_{n+1} - t_n = 15$ min. The advection operator is discretized using a limited $k = 1/3$ upwind flux interpolation as presented in [17], and the diffusion operator using central differences formula. Together they define F_A. The numerical method defining $\bar{F}_A$ is the explicit trapezoidal rule. Concentrations are kept constant at the left boundary ($x = 0$), and at the right boundary, we consider $\frac{\partial c}{\partial n} = 0$. For a full description of the space discretization the reader should consult [25]. With the spatial domain [0, 500] km and a uniform grid $\Delta x = 5$ km, the dimension of the corresponding (1.2) problem is 3200. A highly polluted region is considered between 200 and 300 km, with initial concentrations and emissions as in the urban scenario, and for the rest of the domain rural concentrations and emissions are provided [27]. Emissions take place at constant rate, and are included in F_R. Interpolation is done between the center (250 km) and the urban limits. To allow the system to equilibrate, box models (chemistry only) are integrated for one day over the whole grid. The results are the "true" initial conditions, c_0^{ref}. "Measurements" are then generated every 15 min by a 6-h transport–chemistry run. Figure 4 shows the spatial distribution of the reference concentrations at the beginning (6:00 LT) and at the end (12:00 LT) of the assimilation interval for O_3 and NO_2. First guess initial concentrations are generated similar to c_0^{ref}, but with a uniform injection of 0.1 ppb/hour over the rural area and 0.5 ppb/hour over the urban area of NO_x during the box models integration, which accounts for an error in emission estimates. Assimilation starts at 6:00 LT over a 6-hour interval, with measurements for ozone every 15 min and for NO_2 each hour, at all grid points. A checkpointing strategy is applied for the gradient computations. First, a full forward run is used to store the states of the model after each operator splitting interval. Second, for the backward integration, a forward run stores the states within a splitting interval (1 + all chemistry steps). Third, the pure backward integration is performed, where the intermediate states within transport and within the chemistry steps need to be provided by a simplified forward run. The adjoint part of the advection–diffusion equations is automatically generated using TAMC. The computational scheme for one split interval is described in Fig. 5. The performance of the optimization process is given in Table I (Run 3, KPP-AD) and the assimilation results

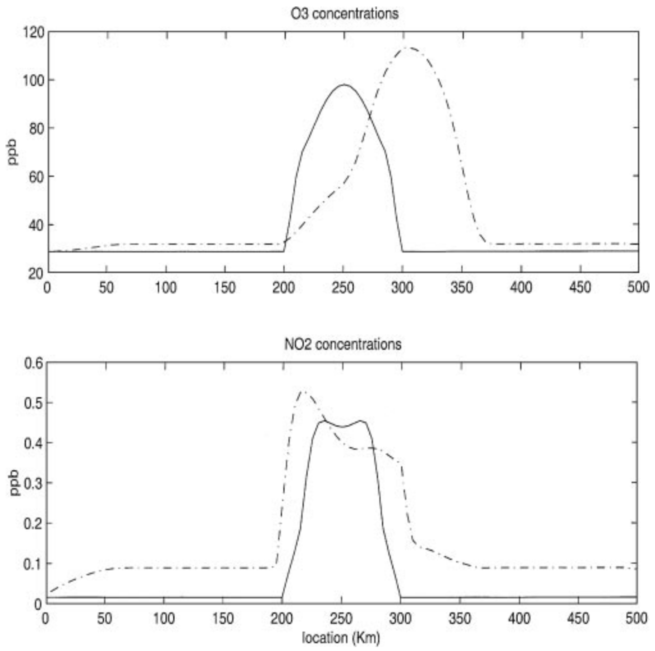

FIG. 4. Spatial distribution of the reference concentrations for O_3 and NO_2. Solid line = initial (LT = 6:00), dotted line = final (LT = 12:00). High NO_x emissions take place in the urban area (200–300 km).

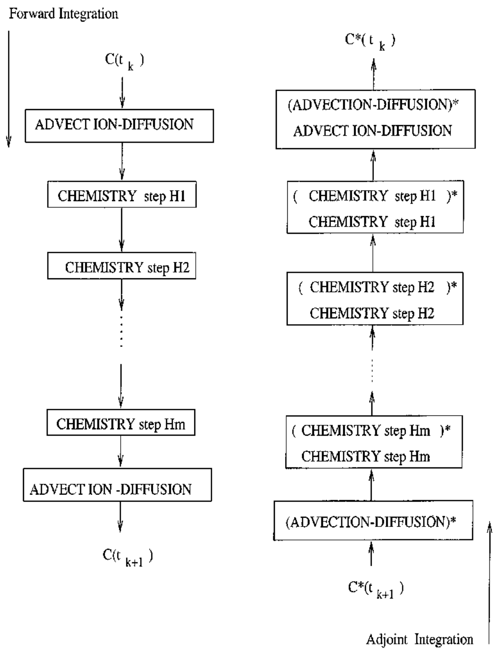

FIG. 5. Computational scheme of the adjoint code for one time split interval. Concentrations are stored after each step during the forward integration, then loaded for the adjoint integration. Additional partial forward computations are required during the adjoint integration.

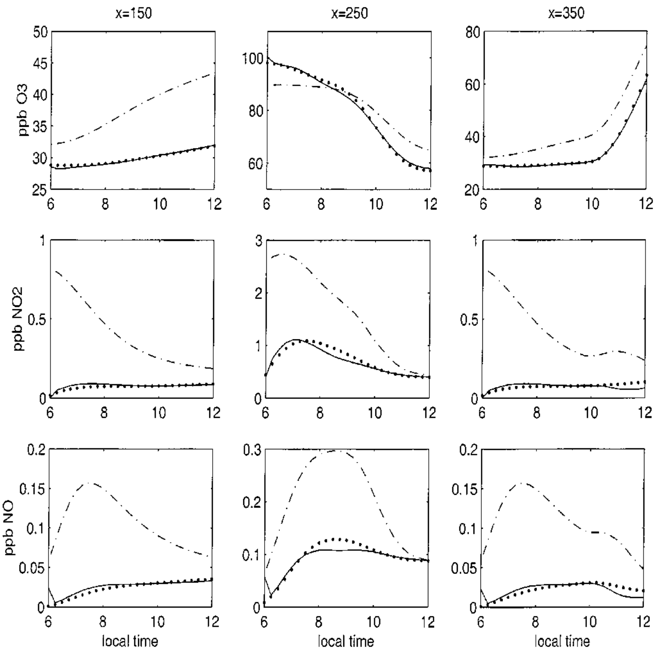

FIG. 6. Assimilation results at some representative points for each area. First line O_3, second NO_2, third NO. Solid dots = true solution, solid line = assimilation result, dotted line = first guess solution.

are presented in Fig. 6. We note here that the previous timing results for the adjoint code are recovered, confirming the success of the implementation. The results obtained with TAMC applied for full transport–chemistry adjoint computations are also included in Table I (Run 3, TAMC).

5. CONCLUSIONS AND FURTHER WORK

The development of powerful computing machines in the past decade made the variational data assimilation technique for large-scale models an intensively explored area. With a dimension of the systems of order 10^6, any attempt to provide the gradient of the cost function using a direct method (finite differences, solving the sensitivity systems) is not feasible, and the adjoint approach is an attractive alternative. In the context of stiff chemical equations, explicit integrators may take prohibitive small stepsize (or just fail), which highly affects the performance of the adjoint code.

While several adjoint models for explicit or semi-implicit numerical methods have been constructed, implementation of implicit methods remains a delicate problem. In this paper we introduced the adjoint computations and an efficient implementation of the two-stage Rosenbrock methods which is suitable for automatization and parallel coding. The algorithm and the properties we described can easily be generalized to *s*—stage methods and it is of interest to analyze how this implementation can be applied to implicit Runge-Kutta methods [16]. Further work includes testing on comprehensive models, extending the set of control variables to include the emission field and depositions, implementing in the context of W-transformation and different types of operator splitting, and the possibility of using approximate gradients.

REFERENCES

1. D. P. Bertsekas, *Nonlinear Programming* (Athena Scientific, Belmont, Massachusetts, 1995).
2. C. Bischof, A. Carle, P. Khademi, and A. Mauer, *The Adifor 2.0 System for the Automatic Differentiation of FORTRAN 77 Programs*, Preprint ANL-MCS-P481-1194 (Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1994).
3. J. G. Blom and J. G. Verwer, *A Comparison of Integration Methods for Atmospheric Transport–Chemistry Problems*, Report MAS-R9910 (CWI, Amsterdam, 1999).
4. R. Byrd, P. Lu, J. Nocedal, and C. Zhu, *"A Limited Memory Algorithm for Bound Constrained Optimization,"* Technical Report NAM-08 (Northwestern University, Evanston, IL, 1994).
5. R. Byrd, J. Nocedal, and R. Schnabel, *Representations of Quasi-Newton Matrices and Their Use in Limited Memory Methods*, Technical Report NAM-03 (Northwestern University, Evanston, IL, 1996).
6. G. R. Carmichael, L. K. Peters, and T. Kitada, A second generation model for regional-scale transport/chemistry/deposition, *Atmos. Environ.* **20**, 173 (1986).
7. W. C. Chao and L.-P. Chang, Development of a four-dimensional variational analysis system using the adjoint method at GLA. 1. Dynamics, *Mon.Weather Rev.* **120**, 1661 (1992).
8. T. F. Coleman, F. Santosa, and A. Verma, Efficient calculation of Jacobian and adjoint vector products in wave propagational inverse problems using automatic differentiation, *J. Comput. Phys.* **157**, 234 (2000).
9. V. Damian-Iordache and A. Sandu, *KPP—A Symbolic Preprocessor for Chemistry Kinetics—User's Guide*, Technical Report (Department of Mathematics, Univ. of Iowa, 1995).
10. H. Elbern and H. Schmidt, A four-dimensional variational chemistry data assimilation scheme for Eulerian chemistry transport modeling, *J. Geophys. Res.* **104-D15**, 18,583 (1999).
11. H. Elbern, H. Schmidt, and A. Ebel, Variational data assimilation for tropospheric chemistry modeling, *J. Geophys. Res.* **102-D13**, 15,967 (1997).
12. M. Fisher and D. J. Lary, Lagrangian four-dimensional variational data assimilation of chemical species, *Q. J. R. Meteorol. Soc.* **121**, 1681 (1995).
13. M. W. Gery, G. Z. Whitten, J. P. Killus, and M. C. Dodge, A photochemical kinetics mechanism for urban and regional scale computer modeling, *J. Geophys. Res.* **94**, 12,925 (1989).
14. R. Giering, *Tangent Linear and Adjoint Model Compiler, Users Manual 1.2*, available at http://puddle.mit.edu/∼ ralf/tamc (1997).
15. R. Giering and T. Kaminski, Recipes for adjoint code construction, *ACM Trans. Math. Software* **24**(4), 437 (1998).
16. E. Hairer and G. Wanner, *Solving Ordinary Differential Equations. II. Stiff and Differential-Algebraic Problems* (Springer-Verlag, Berlin, 1991).
17. D. Hundsdorfer, B. Koren, M. Loon, and J. G. Verwer, A positive finite-difference advection scheme, *J. Comput. Phys.* **117**, 35 (1995).
18. B. V. Khattatov, *et al.*, Assimilation of photochemically active species and a case analysis of UARS data, *J. Geophys. Res.* **104-D15**, 18715 (1999).
19. D. C. Liu and J. Nocedal, On the limited memory BFGS method for large scale minimization, *Math. Prog.* **45**, 503 (1989).
20. G. I. Marchuk, *Adjoint Equations and Analysis of Complex Systems* (Kluwer Academic, Dordrecht/Norwell, MA, 1995).
21. G. I. Marchuk, I. V. Agoshkov, and P. V. Shutyaev, *Adjoint Equations and Perturbation Algorithms in Nonlinear Problems* (CRC Press, Boca Raton, FL, 1996).
22. I. M. Navon, X. Zou, J. Derber, and J. Sela, Variational data assimilation with the N.M.C. spectral model. 1. Adiabatic model tests, *Mon. Weather Rev.* **120**, 1433 (1992).
23. I. M. Navon, Practical and theoretical aspects of adjoint parameter estimation and identifiability in meteorology and oceanography, in *Special Issue in Honor of Richard Pfeffer, Dynam. Atmos. Oceans* **27**(1–4), 55 (1998).
24. N. Rostaing, S. Dalmas, and A. Galligo, Automatic differentiation in Odyssée, *Tellus* **45**, 558 (1993).
25. A. Sandu, G. R. Carmichael, and F. A. Potra, Coupled chemistry and transport computations in air quality modeling, presented at the International Conference on Air Pollution Models APMS'98, Paris, 1998.

26. A. Sandu, F. A. Potra, G. R. Carmichael, and V. Damian, Efficient implementation of fully implicit methods for atmospheric chemical kinetics, *J. Comput. Phys.* **129**, 101 (1996).
27. A. Sandu, J. G. Verwer, M. Loon, G. R. Carmichael, A. F. Potra, D. Dabdub, and J. H. Seinfeld, Benchmarking stiff ODE solvers for atmospheric chemistry problems. I. Implicit versus explicit, *Atmos. Environ.* **31**, 3151 (1997).
28. A. Sandu, J. G. Verwer, J. G. Blom, E. J. Spee, and G. R. Carmichael, Benchmarking stiff ODE solvers for atmospheric chemistry problems. II. Rosenbrock solvers, *Atmos. Environ.* **31**, 3459 (1997).
29. E. J. Spee, *Numerical Methods in Global Transport–Chemistry Models*, Ph.D. thesis (Center for Mathematics and Computer Science (CWI), Amsterdam, 1998).
30. G. Strang, On the construction and comparison of difference schemes, *SIAM J. Numer. Anal.* **5**, 506 (1968).
31. O. Talagrand and P. Courtier, Variational assimilation of meteorological observations with the adjoint of the vorticity equations. I. Theory, *Q. J. R. Meteorol. Soc.* **113**, 1311 (1987).
32. A. Tarantola, *Inverse Problem Theory* (Elsevier, Amsterdam/New York, 1987).
33. J. G. Verwer, J. G. Blom, and W. H. Hundsdorfer, An implicit–explicit approach for atmospheric transport–chemistry problems, *Appl. Numer. Math.* **20**, 191 (1996).
34. J. G. Verwer, E. J. Spee, J. G. Blom, and W. H. Hundsdorfer, *A Second Order Rosenbrock Method Applied to Photochemical Dispersion Problems* (CWI Report, MAS-R9717, Amsterdam, 1997).
35. C. Vreugdenhil and B. Koren (Eds.), *Numerical Methods for Advection–Diffusion Problems*, Vieweg, Wiesbaden, 1994.