Adjoint Implementation of Rosenbrock Methods Applied to Variational Data Assimilation Problems

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

Abstract

In the past decade the variational method has become an attractive alternative to the traditional Kalman filter in data assimilation problems for atmospheric chemistry models. From the computational point of view, its major advantage is that the gradient of the cost function is computed fast, at the expense of few function evaluations, making the optimization process very efficient. The drawbacks are the high storage requirements and the difficulty to implement the adjoint code when sophisticated integrators are used to solve the stiff chemistry. If the sparse structure of the models is carefully exploited Rosenbrock solvers have been proved to be a reliable replacement to the classical methods (QSSA, CHEMEQ,...) due to 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.

Key words: Adjoint model, stiff equations, automatic differentiation, optimization.

1. Introduction. Consider the system of differential equations

$$\begin{cases} \frac{d\mathrm{c}}{dt} = F_1(\mathrm{c}) + F_2(\mathrm{c}) \\ \mathrm{c}(t_0) = \mathrm{c}_0 \end{cases} \tag{1.1}$$

obtained after space discretization on a grid (N_x, N_y, N_z) of an atmospheric chemistry model in a bounded domain $\Omega \in R^3$, where $\mathrm{c}(t) \in R^N$ represents the vector of concentrations of the species involved in the model. For example F_1 may result from the discretization of the advection and diffusion operators, and F_2 represents the stiff chemistry part with emissions and depositions, but several other types of decompositions can be also considered [15, 22]. If the model contains S species,the dimension of problem (1.1) is $N = S \times N_x \times N_y \times N_z$.
Under suitable assumptions, problem (1.1) has an unique solution and we can view this solution

*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
‡Department of Computer Science, Michigan Technological University

as a function of the initial conditions, $\mathrm{c} = \mathrm{c}(t, x, y, z, \mathrm{c}_0)$.
In a classical 4-D variational data assimilation problem [8] one wants to find the values of parameters c_0 that minimize the cost functional

$$\mathcal{F}(\mathrm{c}_0) = \frac{1}{2}(\mathrm{c}_0 - \mathrm{c}_b)^T \mathrm{B}^{-1}(\mathrm{c}_0 - \mathrm{c}_b) + \frac{1}{2}\sum_{k=1}^{m}(\mathrm{c}_k - \mathrm{M}_k)^T \mathrm{R}_k^{-1}(\mathrm{c}_k - \mathrm{M}_k) \tag{1.2}$$

where the "background term" c_b represents an estimate of the initial concentrations which may results from a previous analysis,M_k represents a set of observations at moment t_k, and B and R_k are the covariance matrices given by the errors in data for c_b and M_k, respectively.
Most of the powerfull 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 make the optimization a very expensive computational process. In the variational approach one computes the gradient of the functional $\mathcal{F}$ by using the "adjoint method". The general theory of adjoint equations is described in detail in [14] and the derivation of the adjoint model for the continuous and discrete case are given in [8, 21]. Below we outline the basic ideas:
Taking the covariance matrices to be diagonal,which correspond to the assumption that the errors in measurements are uncorrelated in space and time the gradient of the cost function is:

$$\nabla_{c_0}\mathcal{F}(\mathrm{c}_0) = \mathrm{B}^{-1}(\mathrm{c}_0 - \mathrm{c}_b) + \sum_{k=1}^{m}\left(\frac{\partial \mathrm{c}_k}{\partial \mathrm{c}_0}\right)^T \mathrm{R}_k^{-1}(\mathrm{c}_k - \mathrm{M}_k) \tag{1.3}$$

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

Step1. Initialize $gradient = 0$

Step2. $for\ k = m, 1, -1\ do$

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

Step3. $gradient = \mathrm{B}^{-1}(\mathrm{c}_0 - \mathrm{c}_b) + gradient$

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. Since for large systems constructing the adjoint code by hand can be a frustrating process,automatic tools have been developed. For the theory and actual implementation of the adjoint computations the reader should consult [5, 10, 11]. The algorithm described above requires the values of c_k in reverse order so 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 $c_{k-1} \to c_k^1 \to \ldots \to c_k^s \to c_k$. 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 or semi-implicit numerical method is used to solve the stiff chemistry part of problem (1.1) then the "trajectory" from c_{k-1} to c_k may become very long, increasing the cost of the adjoint code. On another 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 (1.1) is to use operator splitting,which has the advantage that different processes can be treated with different numerical methods.In a second order accurate Strang splitting [20] approach the solution $c(t_{k+1})$ is obtained from $c(t_k)$ as follows:

$$\frac{\partial}{\partial t}c^1(t) = F_1(c^1(t)) \ , \ t_k \le t \le t_{k+\frac{1}{2}} \ , \ c^1(t_k) = c(t_k) \tag{2.1}$$

$$\frac{\partial}{\partial t}c^2(t) = F_2(c^2(t)) \ , \ t_k \le t \le t_{k+1} \ , \ c^2(t_k) = c^1(t_{k+\frac{1}{2}}) \tag{2.2}$$

$$\frac{\partial}{\partial t}c^3(t) = F_1(c^3(t)) \ , \ t_{k+\frac{1}{2}} \le t \le t_{k+1} \ , \ c^3(t_{k+\frac{1}{2}}) = c^2(t_{k+1}) \tag{2.3}$$

$$c(t_{k+1}) = c^3(t_{k+1}) \tag{2.4}$$

If the numerical solution of (2.1-2.4) is written as:

$$c^1(t_{k+\frac{1}{2}}) = F_1^{n_1}(c(t_k)) \ ; \ c^2(t_{k+1}) = F_2^{n_2}(c^1(t_{k+\frac{1}{2}})) \ ; \ c(t_{k+1}) = F_1^{n_3}(c^2(t_{k+1}))$$

one has for the adjoint computations:

$$\left[\frac{\partial c(t_{k+1})}{\partial c(t_k)}\right]^T = \left[\frac{\partial F_1^{n_1}(c(t_k))}{\partial c(t_k)}\right]^T \cdot \left[\frac{\partial F_2^{n_2}(c^1(t_{k+\frac{1}{2}}))}{\partial c^1(t_{k+\frac{1}{2}})}\right]^T \cdot \left[\frac{\partial F_1^{n_3}(c^2(t_{k+1}))}{\partial c^2(t_{k+1})}\right]^T$$

Usually the advection-diffusion equations (2.1,2.3) are solved using explicit methods,while the stiff chemistry requires an implicit integrator.The computational cost of solving problem (2.1-2.4) is then concentrated in the chemistry part which takes in practice as much as 90% of the CPU time. Since the adjoint method requires several integrations of the direct model, the storage of (part of) the forward trajectory and the $(jacobian)^T \cdot vector$ products, the performance of the adjoint model is dominated by the implementation of the direct and adjoint method used to solve equation (2.2). *Fisher and Lary*[8] show the adjoint computations for the semi-implicit midpoint rule, and *Elbern et al.*[7] use the adjoint model for the QSSA integration.In the next section we present the adjoint formulas for a general 2-stages Rosenbrock method and an efficient

implementation which is suitable for automatization. Extension to a s-stages method [12] is straightforward.

3. Adjoint computations and implementation for a 2-stages Rosenbrock method.
i) Derivation of the adjoint formulas. We consider now the problem

$$\left\{ \begin{array}{l} \frac{d\mathrm{c}}{dt} = f(\mathrm{c}) \\ \mathrm{c}(t_0) = \mathrm{c}_0 \end{array} \right. \tag{3.1}$$

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-stages Rosenbrock method as presented in [12] reads:

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

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

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

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. Since of special interest are the methods that require only one LU decomposition of $\frac{1}{\gamma_{ii} h} I - J_0$ per step, we consider the case when $\gamma_{11} = \gamma_{22} = \gamma$.
For the adjoint computations we have from (3.2), (3.3):

$$\left(\frac{\partial \mathrm{k}_1}{\partial \mathrm{c}_0}\right)^T = \left(J_0^T + (\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_1)^T\right) \left((\frac{1}{\gamma h} I - J_0)^T\right)^{-1} \tag{3.5}$$

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

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 and we will return to the computation of these terms later.
Using (3.4), for an arbitrary seed vector $\mathrm{u} \in R^n$ we have:

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

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

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

Step 2. Compute $\omega = J_1^T(m_2\mathrm{v})$; $\omega 1 = \alpha\omega + \frac{m_2\beta}{h}\mathrm{v}$.

$$\left(\frac{\partial \mathrm{c}_1}{\partial \mathrm{c}_0}\right)^T \mathrm{u} = \mathrm{u} + \omega + m_1\left(J_0^T + (\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_1)^T\right)\mathrm{v} + (\frac{\partial \mathrm{k}_1}{\partial \mathrm{c}_0})^T\omega_1 + m_2(\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k}_2)^T\mathrm{v} \tag{3.6}$$

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) and arranging the terms,

Step 4. Compute

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

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 *Step* 1-4. Here k , v $\in R^n$ are arbitrary constant vectors. For the i component we have:

$$\left((\frac{\partial J_0}{\partial \mathrm{c}_0} \times \mathrm{k})^T\mathrm{v}\right)_i = \left(\frac{\partial J_0}{\partial \mathrm{c}_0^i}\mathrm{k}\right)^T \mathrm{v} = \mathrm{k}^T\left(\frac{\partial J_0}{\partial \mathrm{c}_0^i}\right)^T \mathrm{v} = \mathrm{k}^T\left(\frac{\partial (J_0^T\mathrm{v})}{\partial \mathrm{c}_0^i}\right) = \left(\frac{\partial (J_0^T\mathrm{v})}{\partial \mathrm{c}_0^i}\right)^T \mathrm{k} \tag{3.8}$$

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

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

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

$$\partial H_{i,j} = \frac{\partial H_i(\mathrm{c}_0)}{\partial \mathrm{c}_0^j} = \left(\frac{\partial^2 f_1}{\partial \mathrm{c}_0^i \partial \mathrm{c}_0^j}\right)\mathrm{v}_1 + \cdots + \left(\frac{\partial^2 f_n}{\partial \mathrm{c}_0^i \partial \mathrm{c}_0^j}\right)\mathrm{v}_n = H_{f_1}(i,j)\mathrm{v}_1 + \cdots + H_{f_n}(i,j)\mathrm{v}_n$$

where H_{f_i} is the hessian matrix of the function $f_i : R^n \to R$.
Then $\frac{\partial H(\mathrm{c}_0)}{\partial \mathrm{c}_0} = \sum_{i=1}^n H_{f_i}\mathrm{v}_i$, so $\frac{\partial H(\mathrm{c}_0)}{\partial \mathrm{c}_0}$ is symmetric. Using now (3.8) it results :

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

ii) Implementation of the adjoint code. The forward integration of problem (3.1) using implicit methods together with the performance analysis is given in [17, 18, 22], 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 [6] which generates the sparse matrix factorization LU required in (3.2, 3.3) with a minimal fill-in [16] and the routine to backsolve the linear systems without indirect adressing. 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 backward adjoint integration.
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 *Step*1-4. With the LU decomposition of $(\frac{1}{\gamma h} I - J_0)$ available from (3.2), *Step 1* reads : $U^T L^T \mathrm{v} = \mathrm{u}$. We introduce then a loop free routine *tsolve* for forward-backward solving this system in sparse format avoiding indirect addressing. The computational cost of *Step* 1,3 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 generated using sparse multiplications by a new routine *jactrvect*. 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 (3.7).
Using (3.9), we can rewrite (3.7) as:

$$\left(\frac{\partial \mathrm{c}_1}{\partial \mathrm{c}_0}\right)^T \mathrm{u} = \mathrm{u} + \omega + H_1(\mathrm{c}_0) + \left(\frac{\partial H_1(\mathrm{c}_0)}{\partial \mathrm{c}_0}\right) \mathrm{k}_1 + m_2 \left(\frac{\partial H_2(\mathrm{c}_0)}{\partial c_0}\right) \mathrm{k}_2 \tag{3.10}$$

with $H_1, H_2 : R^n \to R^n$, $H_1(\mathrm{c}_0) = J_0^T(m_1 \mathrm{v} + \theta)$, $H_2(\mathrm{c}_0) = J_0^T \mathrm{v}$.
In (3.10) we have then to compute the $jacobian \cdot vector$ products for the functions H_1, H_2 which can be done by $forward$ automatic differentiation [4, 10] of the functions genereted via routine *jactrvect*. The cost is then 2-3 times the cost of evaluating $H_1(\mathrm{c}_0), H_2(\mathrm{c}_0)$ and remains low due to the sparse structure of J_0. Automatic differentiation for H_1 provides also the value $H_1(\mathrm{c}_0)$, so there is no need to compute it separately. Last but not least, these computations are independent allowing parallel implementation.

4. Performance and validation of the adjoint model. The algorithm presented in Section 3 has 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 easily move from one model to another and makes it very attractive compared with the hand written codes which construction for large models can be a difficult process. Moreover since symbolic computations are used, rounding errors are avoided and the accuracy of the results goes up to

the machine precision. In Figure 1 we present the scheme of the implementation of the adjoint code applied to the data assimilation problem for a general chemistry-transport model.
For the numerical experiments we consider the 2^{nd} order 2-stages 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[1]. The superior stability, positivity and conservation properties of this scheme are analized by *Verver et al.* [22] who reports good results in the context of various types of operator splitting even when large fixed step sizes (10 to 20 min.) are used.
i) The box model. In order 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, [9]) 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 (1.2). 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 [2, 3], anticipating extension to large scale models. The initial concentrations follow the urban scenario as described in [17], with an initial concentration of 70 ppb for O3. Assimilation starts at the beginning of the second day (6:00 LT) over a period of 6 hours, with measurements provided every 15 minutes. As the initial guess for the concentrations we choose the values at the beginning of the first day. The one day period is introduced in order to allow the system to equilibrate. The integration is restarted every 15 minutes with a minimum stepsize of 1 sec., simulating an operator splitting environment. In the first experiment measurements were provided for ozone only and in the second one for ozone and NO2. As an alternative way to compute the gradient we use the second order central difference formula [1]. Figure 2 (left 1^{st} case, right 2^{nd} case) shows the relative and absolute differences between the computed gradients for some of the most important species in the model, and Figure 3 (left,right) shows the results of the assimilation. It can be seen that introducing NO2 measurements is of benefit not only for the NO2 and NO analysis, but also for the O3 analysis. However, since additional constraints are introduced this increases the number of iterations in the optimization routine as we will see next.
Implementation of the adjoint code uses three forward integrations per backward integration: the first run is used to store the states of the system at the measurement moments, the second run to store the trajectory between measurements and information about the stepsize used, and the third run is for the forward-backward integration. The technical report of the optimization process is outlined in the table below. It can 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 less than four[2] , which makes our implementation very efficient.

[1] The numerical experiments presented in this section were performed with $\gamma = 1 + 1/\sqrt{2}$.

[2] All the computations were done on a HP-UX B.10.20 A 9000/778 machine. The time to read-write data to files is not considered

Run	No. iter.	∼time/iter (sec)	∼(grad.time)/(fwd.time)	$\mathcal{F}$(it.0)/$\mathcal{F}$(it.end)
1.($O3$)	27	0.36	3.82	1.e5
2.($O3$, $NO2$)	32	0.37	3.85	2.8e3

ii) Application to a 1-D problem. We consider now a one dimensional test problem corresponding to the advection-diffusion-reaction model:

$$\frac{d}{dt}\mathrm{c}_i = -\nabla(u\mathrm{c}_i) + \mathrm{div}(K\nabla \mathrm{c}_i) + f_i(t, x, \mathrm{c}_1, \ldots, \mathrm{c}_S) + E_i(t, x),\ i = 1 \ldots S. \tag{4.1}$$

The wind field and the diffusion coefficient are taken constant, $u = 10$ km/hour (left-to-right), $K = 10^{-3}\mathrm{Km}^2$/sec. The advection operator is discretized using a limited $k = 1/3$ upwind flux interpolation as presented in [23], and the diffusion operator using central differences formula. Concentrations are kept constant at the left boundary (x=0) and at the right boundary we consider $\frac{\partial \mathrm{c}}{\partial x} = 0$. For a full description of the space discretization the reader should consult [15]. Second order Strang splitting is applied according to (2.1-2.4), with a splitting interval $t_{k+1} - t_k = 15$ min. With the spatial domain [0,500] Km, and a uniform grid $\Delta x = 5$ Km, the dimension of the corresponding (1.1) problem is 3200. A highly polluted region is considered between 200 and 300 Km, with initial concentrations and emissions as for the urban scenario and for the rest of the domain rural concentrations and emissions are provided [17]. Interpolation is done beetween the center (250 Km) and the urban limits. "True" initial concentrations (then measurements) are obtained by integrating box models over the whole grid (with no transport) for one day. 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 O3 and NO2. The perturbations are generated in the same way, but with an uniform injection of 0.1 ppb/hour over the rural area and 0.5 ppb/hour over the urban area of NO_x, which accounts for an error in emission estimates. Assimilation starts at 6:00 LT over a six hour interval, with measurements for ozone every 15 min. and NO2 each hour, at all grid points. The adjoint code uses three[3] forward integrations per backward integration as described above with the adjoint part of the transport- difussion equations automatically generated by the adjoint model compiler TAMC [10, 11]. The computational scheme for one split interval is described in Figure 5. The performance of the optimization process is given below and the assimilation results are presented in Figure 6. We note here that the previous timing results for the adjoint code are recovered, confirming the succes of the implementation.

Run	No. iter.	∼time/iter (sec)	∼(grad.time)/(fwd.time)	$\mathcal{F}$(it.0)/$\mathcal{F}$(it.end)
3. ($O3, NO2$)	34	10.8	3.76	1.e2

5. Conclusions and further work. The development of powerfull computing machines in the past decade made the variational data assimilation technique for large scale models an intensive

[3]Only $2\frac{1}{2}$ integrations for the transport-diffusion part are taken since there are no intermediar steps

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 eliminated, which requires an adjoint approach. 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 2-stages Rosenbrock methods which is suitable for automatization and parallel coding. The algorithm and the properties we described can be easily generalized to a $s-$stages method and it appears of full interest to analyse how this implementation can be extended to SDIRK and IRK methods [12]. Further work includes testing on comprehensive models, implementation in the context of W-transformation and different types of operator splitting as well as the possibility to use approximate gradients.

References

[1] Bertsekas, D.P.: Nonlinear Programming. *Athena Scientific*, 1995

[2] Byrd, R., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. *Tech. Rep. NAM-08, Northw. Univ., Evanston, Ill.*, 1994

[3] Byrd, R., Nocedal, J., Schnabel, R.: Representations of Quasi-Newton matrices and their use in limited memory methods. *Tech. Rep. NAM-03, Northw. Univ., Evanston, Ill.*, 1996

[4] Bischof, C., Carle, A., Khademi, P., Mauer, A.: The Adifor 2.0 system for the automatic differentiation of FORTRAN 77 programs. *Tech. Rep., Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois*, 1994

[5] Chao, W.C., Chang, L.-P.: Development of a four-dimensional variational analysis system using the adjoint method at GLA. Part 1: Dynamics. *Mon. Weather Rev.,* **120**, *1661-1673*, 1992

[6] Damian-Iordache, V., Sandu, A.: KPP-A symbolic preprocessor for chemistry kinetics-User's guide. *Tech. Rep., Univ. of Iowa, Department of Mathematics*, 1995

[7] Elbern, H., Schmidt, H., Ebel, A.: Variational data assimilation for tropospheric chemistry modeling. *J. of Geographical Research,* **102-D13**, *15967-15985*, 1997

[8] Fisher, M., Lary, D.J.: Lagrangian four-dimensional variational data assimilation of chemical species. *Q.J.R. Meteorol. Soc.,* **121**, *1681-1704*, 1995

[9] Gery, M.W., Whitten, G.Z., Killus, J.P., Dodge, M.C.: A photochemical kinetics mechanism for urban and regional scale computer modeling. *J. of Geophysical Research,* **94**, *12925-12956*, 1989

[10] Giering, R.: Tangent linear and Adjoint Model Compiler, Users manual 1.2 . *"http : //puddle.mit.edu/ ∼ ralf/tamc"*, 1997

[11] Giering, R., Kaminski, T.: Recipes for Adjoint Code Construction. *ACM Trans. Math. Software*, 1998

[12] Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. *Springer-Verlag, Berlin*, 1991

[13] Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale minimization. *Math. Prog.*, **45**, *503-528*, 1989

[14] Marchuk, G.I.: Adjoint equations and analysis of complex systems. *Kluwer Academic Publishers*, 1995

[15] Sandu, A., Carmichael, G.R., Potra, F.A.: Coupled chemistry and transport computations in air quality modeling. *International conference on air pollution models APMS'98*, Paris 1998

[16] Sandu, A., Potra, F.A., Carmichael G.R., Damian, V.: Efficient implementation of fully implicit methods for atmospheric chemical kinetics. *J. of Comput. Physics,* **129**, *101-110*, 1996

[17] Sandu, A., Verwer, J.G., Loon, M., Carmichael, G.R., Potra, A.F., Dabdub, D., Seinfeld, J.H.: Benchmarking stiff ODE solvers for atmospheric chemistry problems I: Implicit versus explicit. *Atmos. Environ.,* **31**, *3151-3166*, 1997

[18] Sandu, A., Verwer, J.G., Blom, J.G., Spee, E.J., Carmichael, G.R.: Benchmarking stiff ODE solvers for atmospheric chemistry problems II: Rosenbrock solvers. *Atmos. Environ.,* **31**, *3459-3472*, 1997

[19] Spee, E.J.: Numerical methods in global transport-chemistry models. *Ph.D. Thesis, Center for Mathematics and Computer Science (CWI), Amsterdam*, 1998

[20] Strang, G.: On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis, 5:506-517*, 1968

[21] Talagrand, O., Courtier, P.: Variational assimilation of meteorological observations with the adjoint of the vorticity equations. Part I. Theory. *Q.J.R. Meteorol. Soc.,* **113**, *1311-1328*, 1987

[22] Verwer, J.G., Spee, E.J., Blom, J.G., Hundsdorfer, W.H.: A second order Rosenbrock method applied to photochemical dispersion problems. *CWI Report, MAS-R9717*, 1997

[23] Vreugdenhil, C., Koren, B., editors: Numerical Methods for Advection-Diffusion Problems. *Vieweg*, 1994

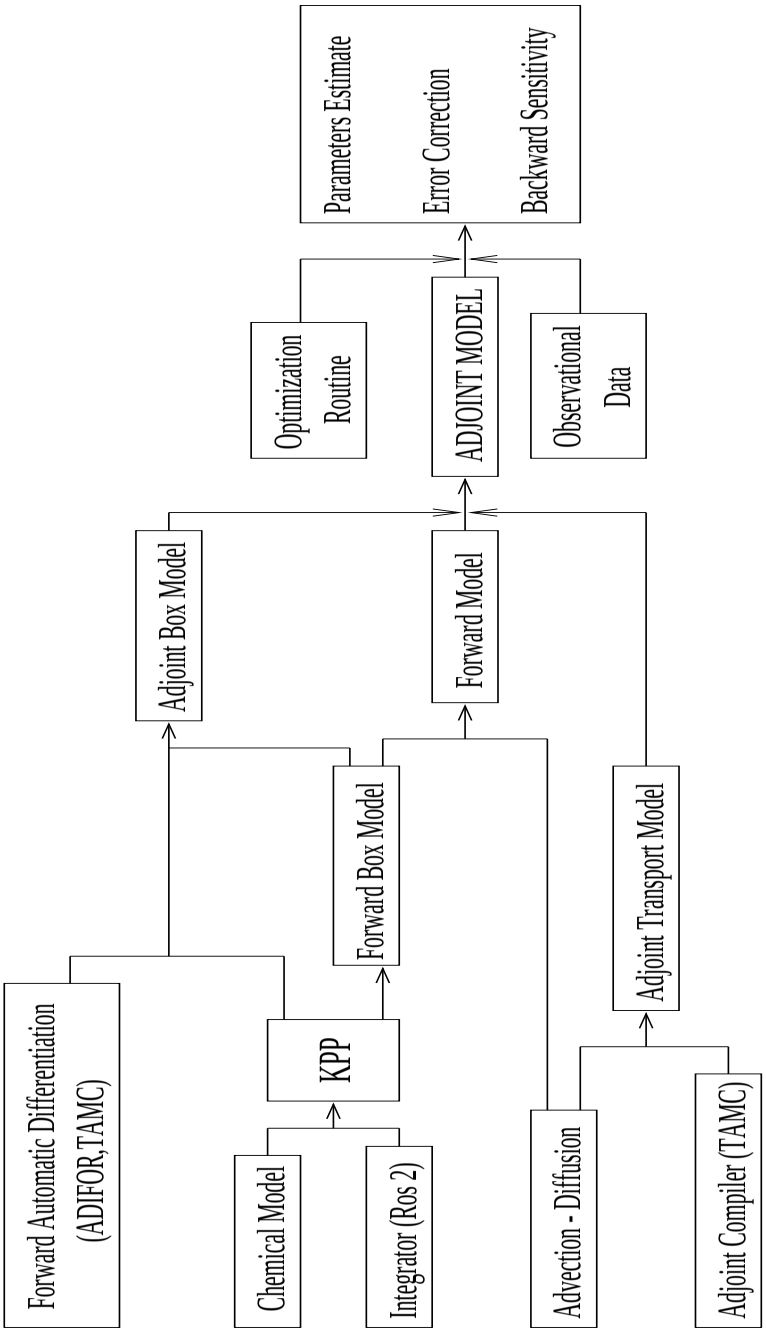

Figure 1: Implementation of the adjoint code applied to the data assimilation problem for a chemistry-transport model.

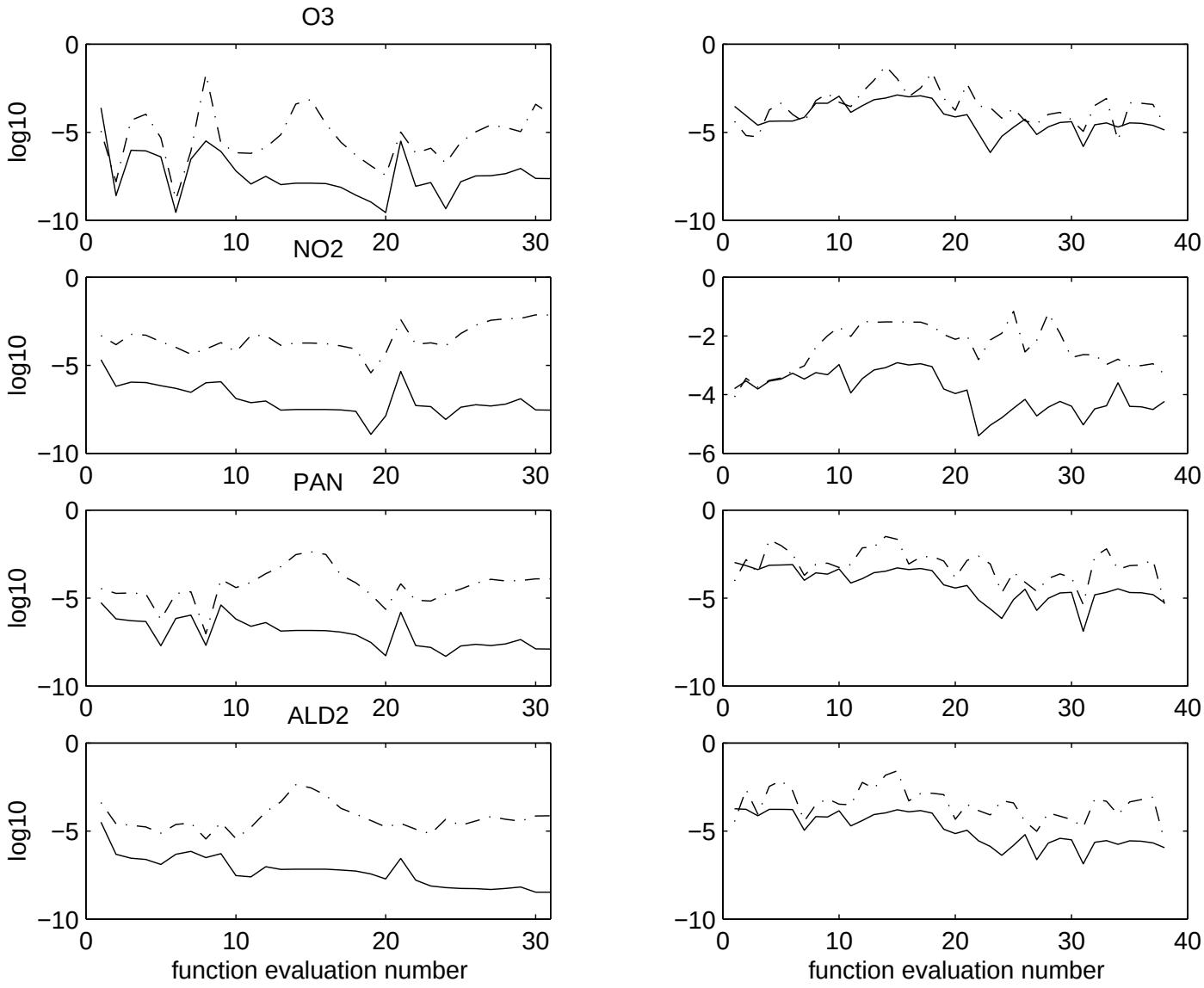

Figure 2: Absolute (solid line) and relative (dotted line) differences between the computed gradients. 1^{st} case -left, 2^{nd} case-right

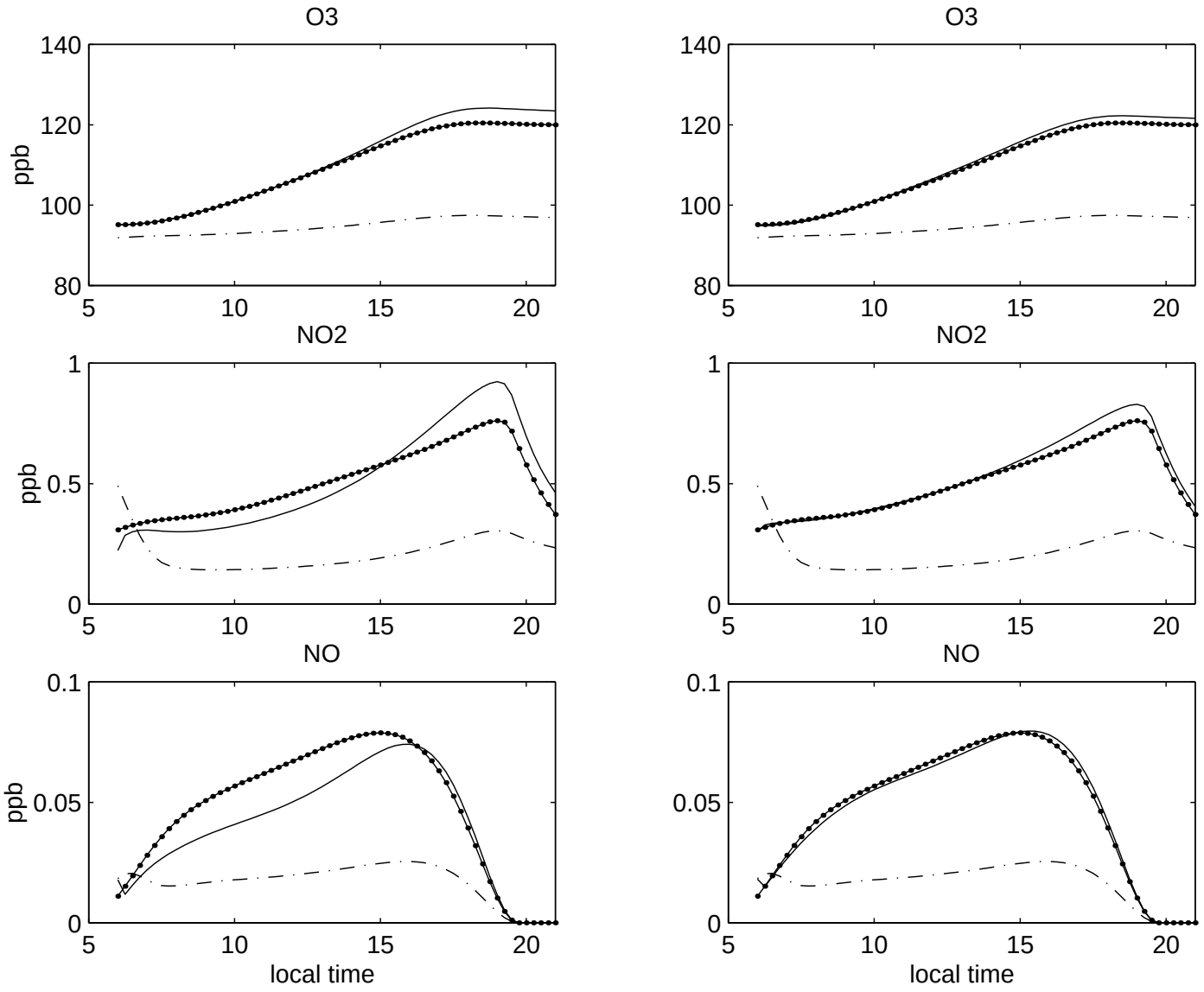

Figure 3: Assimilation takes place from 6 to 12 LT; left 1^{st} case, right 2^{nd}. Solid line with dots =true; solid line = assimilation result; dotted line= first guess

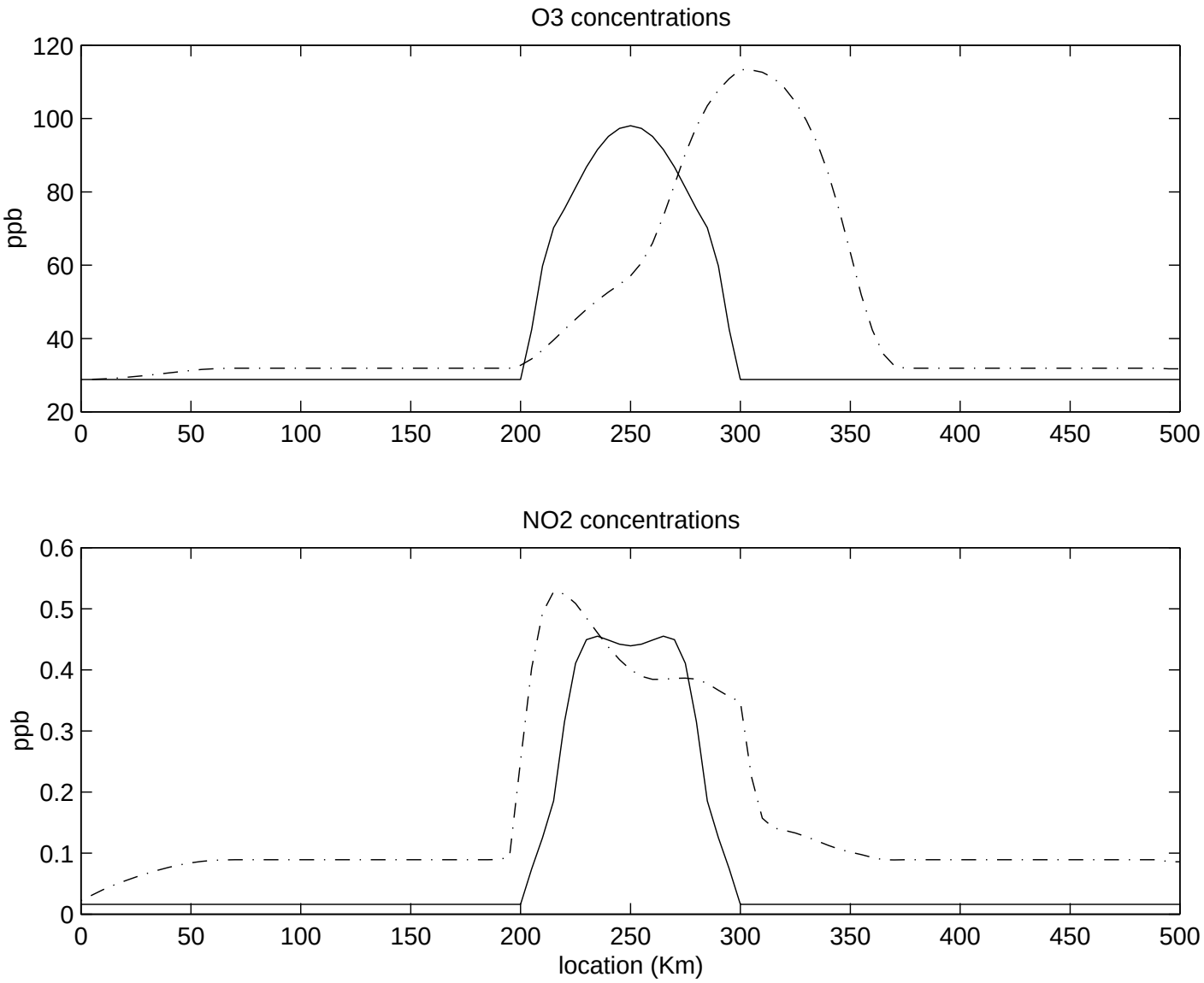

Figure 4: Spatial distribution of the reference concentrations for O3 and NO2. Solid line = initial(LT=6:00), dotted line = final (LT=12:00)

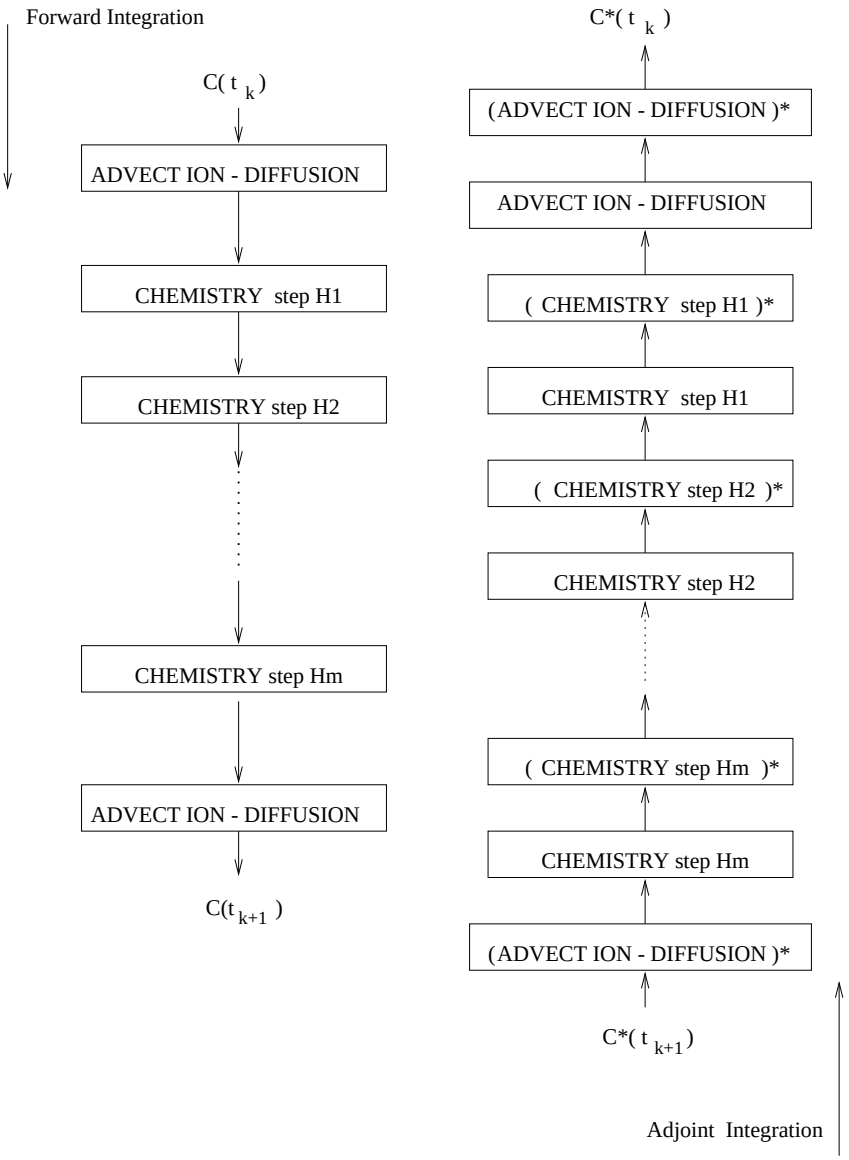

Figure 5: Computational scheme of the adjoint code for one time split interval. Concentrations are stored for each step during the forward integration, then loaded during the adjoint integration.

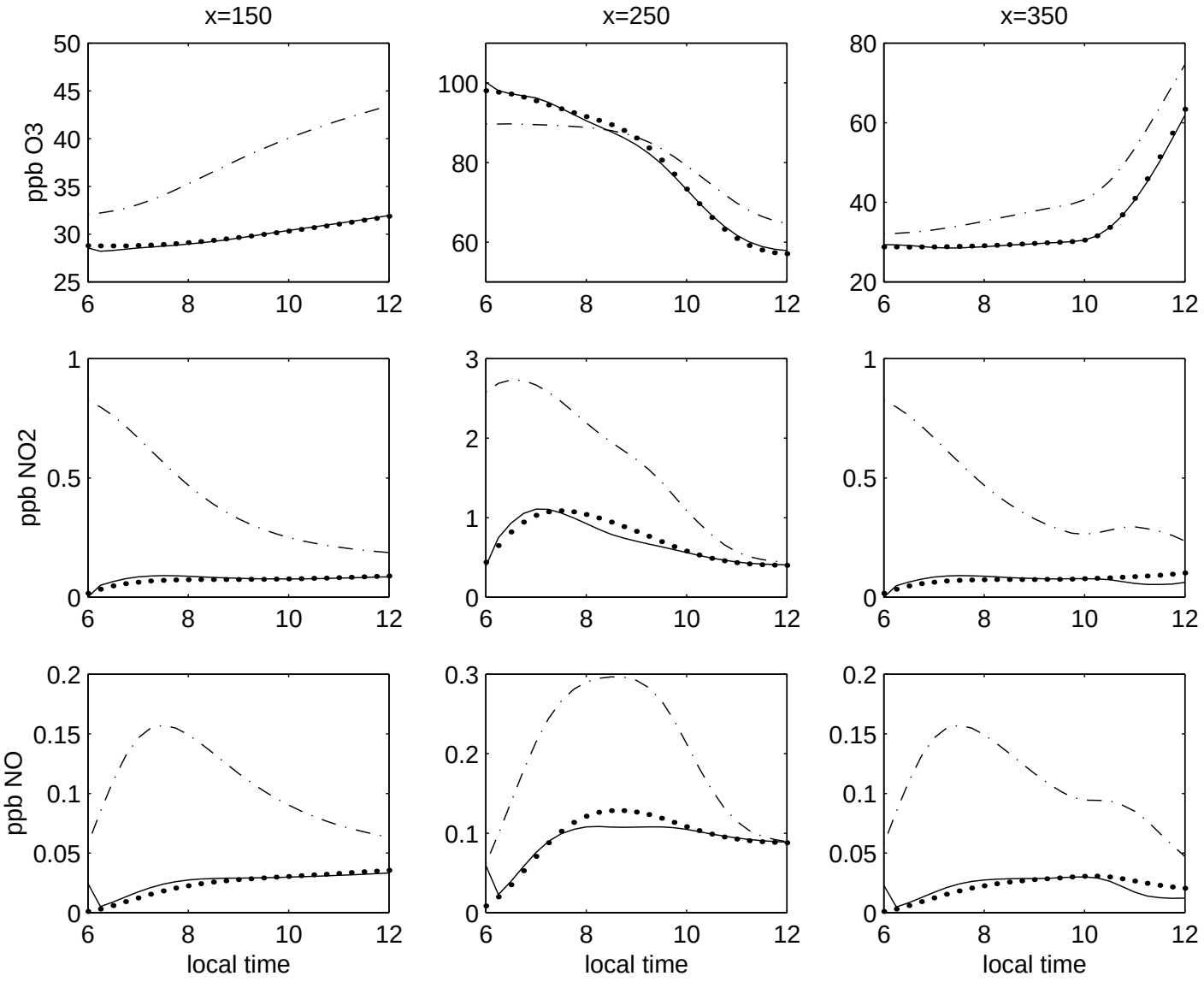

Figure 6: Assimilation results at some representative points for each area. First line O3, second NO2, third NO. Solid dots= true solution, solid line= assimilation result, dotted line= first guess solution.