Atmospheric Environment 37 (2003) 5083–5096

ATMOSPHERIC ENVIRONMENT

www.elsevier.com/locate/atmosenv

Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: Part I—theory and software tools

Adrian Sandu[a], Dacian N. Daescu[b], Gregory R. Carmichael[c,*]

[a] *Department of Computer Science, Virginia Polytechnic Institute and State University, 660 McBryde Hall, Blacksburg, VA 24061, USA*
[b] *Institute for Mathematics and its Applications, University of Minnesota, 400 Lind Hall, 207 Church Street S.E., Minneapolis, MN 55455, USA*
[c] *Center for Global and Regional Environmental Research, University of Iowa, 204 IATL, Iowa City, IA 52242-1297, USA*

Received 26 January 2003; accepted 7 August 2003

Abstract

The analysis of comprehensive chemical reactions mechanisms, parameter estimation techniques, and variational chemical data assimilation applications require the development of efficient sensitivity methods for chemical kinetics systems. The new release (KPP-1.2) of the kinetic preprocessor (KPP) contains software tools that facilitate direct and adjoint sensitivity analysis. The direct-decoupled method, built using BDF formulas, has been the method of choice for direct sensitivity studies. In this work, we extend the direct-decoupled approach to Rosenbrock stiff integration methods. The need for Jacobian derivatives prevented Rosenbrock methods to be used extensively in direct sensitivity calculations; however, the new automatic and symbolic differentiation technologies make the computation of these derivatives feasible. The direct-decoupled method is known to be efficient for computing the sensitivities of a large number of output parameters with respect to a small number of input parameters. The adjoint modeling is presented as an efficient tool to evaluate the sensitivity of a scalar response function with respect to the initial conditions and model parameters. In addition, sensitivity with respect to time-dependent model parameters may be obtained through a single backward integration of the adjoint model. KPP software may be used to completely generate the continuous and discrete adjoint models taking full advantage of the sparsity of the chemical mechanism. Flexible direct-decoupled and adjoint sensitivity code implementations are achieved with minimal user intervention. In a companion paper, we present an extensive set of numerical experiments that validate the KPP software tools for several direct/adjoint sensitivity applications, and demonstrate the efficiency of KPP-generated sensitivity code implementations.
© 2003 Elsevier Ltd. All rights reserved.

Keywords: Chemical kinetics; Sensitivity analysis; Direct-decoupled method; Adjoint model

1. Introduction

The mathematical formulation of chemical reaction mechanisms is given by a coupled system of stiff nonlinear differential equations

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y; p), \quad y(t^0) = y^0, \quad t^0 \leqslant t \leqslant t^{\mathrm{F}}. \tag{1}$$

The solution $y(t) \in \Re^n$ represents the time evolution of the concentrations of the species considered in the chemical mechanism starting from the initial configuration y^0. Throughout this work vectors will be represented in column format and an upper script $(\cdot)^{\mathrm{T}}$ will denote the transposition operator. The rate of change in the concentrations y is determined by the nonlinear

*Corresponding author. Tel.: +1-319-335-3333; fax: +1-319-335-3337.

E-mail addresses: asandu@cs.vt.edu (A. Sandu), daescu@ima.umn.edu (D.N. Daescu), gcarmich@cgrer.uiowa.edu (G.R. Carmichael).

1352-2310/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.atmosenv.2003.08.019

production/loss function $f = (f_1, \ldots, f_n)^{\mathrm{T}}$, which depends on a vector of parameters $p \in \Re^m$. In practice, the vector p may represent reaction rate coefficients, the initial state of the model ($p = y^0$), additional source/sink terms (e.g. emission rates), etc. We assume that problem (1) has a unique solution $y = y(t, p)$ once the model parameters are specified. Comprehensive atmospheric reaction mechanisms take into consideration as many as 100 chemical species involved in hundreds of chemical reactions (see e.g. Carter, 2000), such that to efficiently integrate system (1) fast and reliable numerical methods must be implemented (Byrne and Dean, 1993; Jacobson and Turco, 1994; Sandu et al., 1997a, b). In addition, the model parameters are often obtained from experimental data and their accuracy is hard to estimate. The development and validation of chemical reactions mechanisms require a systematic sensitivity analysis to evaluate the effects of parameter variations on the model solution.

The sensitivities $S_\ell(t) \in \Re^n$ are defined as the derivatives of the solution with respect to the parameters

$$S_\ell(t) = \frac{\partial y(t)}{\partial p_\ell}, \quad 1 \leqslant \ell \leqslant m. \tag{2}$$

A large sensitivity value $S_\ell(t)$ shows that the parameter p_ℓ plays an essential role in determining the model forecast $y(t)$, therefore one problem of interest is to evaluate the sensitivities $S(t) \in \Re^{n \times m}$ for $t^0 \leqslant t \leqslant t^{\mathrm{F}}$. Some practical applications (e.g. data assimilation) require the sensitivity of a scalar response function

$$g = \int_{t^0}^{t^{\mathrm{F}}} \hat{g}(t, y(t, p))\,\mathrm{d}t \tag{3}$$

with respect to the model parameters

$$\frac{\partial g}{\partial p} = \int_{t^0}^{t^{\mathrm{F}}} S^{\mathrm{T}}(t) \frac{\partial \hat{g}}{\partial y}(t, y(t, p))\,\mathrm{d}t. \tag{4}$$

Depending on the problem at hand, an appropriate method for sensitivity evaluation must be selected. The most popular and efficient techniques for sensitivity studies are the direct-decoupled and the adjoint sensitivity methods. These approaches are complementary; for example, a single direct sensitivity calculation could provide $\partial[y_1(t), \ldots, y_n(t)]/\partial y_j(t^0)$ for all $t^0 \leqslant t \leqslant t^{\mathrm{F}}$, while a single adjoint calculation provides $\partial y_j(t^{\mathrm{F}})/\partial[y_1(t), \ldots, y_n(t)]$ for all $t^0 \leqslant t \leqslant t^{\mathrm{F}}$. The sensitivity information provided by the direct method may be used to asses how parametric uncertainties propagate in the system; the adjoint method is more suitable for inverse modeling and may be used to identify the origin of uncertainty in a given model output. In this paper, special emphasis is given to the adjoint technique which has not been used as extensively as the direct method in the context of chemical systems.

Given the multitude of applications, the continuous development of new reaction mechanisms and the frequent modifications of the existing ones, there is a need for software tools that facilitate the sensitivity analysis of general chemical kinetic mechanisms. The kinetic preprocessor (KPP) (Damian-Iordache et al., 1995) has been successfully used in the forward integration of the chemical kinetics systems (Sandu et al., 1997a, b; Verwer et al., 1999). The new release (KPP-1.2) presented in this paper implements a comprehensive set of software tools for direct and adjoint sensitivity analysis. Given a chemical mechanism described by a list of chemical reactions, KPP generates a flexible code for the model, its forward integration and the direct-decoupled/adjoint sensitivity analysis.

The paper is organized as follows: a review of the direct-decoupled sensitivity analysis and extensions to Runge–Kutta and Rosenbrock stiff integration methods are presented in Section 2. In Section 3, we review the continuous and discrete adjoint sensitivity methods and discuss practical issues of the adjoint code implementation for chemical kinetics systems. The KPP tools that facilitate the implementation of the sensitivity methods are presented in Section 4. Tutorial examples for building the direct-decoupled code and the adjoint code with KPP are presented in Sections 5 and 6, respectively. In Section 7, we outline the new numerical methods for sensitivity calculations available in the KPP numerical library. A summary of the results and concluding remarks are presented in Section 8.

2. Direct sensitivity analysis

For the direct analysis we consider the parameters p to be constant, i.e. they do not change in time. By differentiating (1) with respect to the parameters one obtains the sensitivity equations (variational equations)

$$\frac{\mathrm{d}S_\ell}{\mathrm{d}t} = J(t, y; p) S_\ell + f_{p_\ell}(t, y; p),$$
$$S_\ell(t^0) = \frac{\partial y^0}{\partial p_\ell}, \quad 1 \leqslant \ell \leqslant m, \tag{5}$$

where J is the Jacobian matrix of the derivative function and f_{p_ℓ} are its partial derivatives with respect to the parameters

$$J(t, y; p) = \frac{\partial[f_1, \ldots, f_n]}{\partial[y_1, \ldots, y_n]},$$
$$f_{p_\ell}(t, y; p) = \frac{\partial[f_1, \ldots, f_n]}{\partial p_\ell}, \quad 1 \leqslant \ell \leqslant m. \tag{6}$$

If the parameter p_ℓ represents the initial concentration of the ℓth species, $p_\ell = y_\ell^0$, then $f_{p_\ell} = 0$ and $S_\ell(t^0) = (0, \ldots, 0, 1, 0, \ldots, 0)^{\mathrm{T}}$, the ℓth vector of the canonical basis in $\mathscr{R}^n$; otherwise, $S_\ell(t^0) = 0$.

The variational equations (5) are linear. The direct method solves simultaneously the model equation (1) together with the variational equations (5) to obtain both concentrations and sensitivities. The combined system (1)–(5) has the Jacobian

$$\frac{\partial[f, JS_1 + f_{p_1}, \ldots, JS_m + f_{p_m}]}{\partial[y, S_1, \ldots, S_m]} = \begin{pmatrix} J & 0 & \cdots & 0 \\ (JS_1)_y + J_{p_1} & J & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ (JS_m)_y + J_{p_m} & 0 & \cdots & J \end{pmatrix}, \qquad (7)$$

where the subscripts in the component sub-matrices denote partial differentiation. The eigenvalues of the combined Jacobian (7) are the eigenvalues of J (the Jacobian of the model equations), with different multiplicities; therefore, if model (1) is stiff the sensitivity equations (5) are also stiff. To maintain stability, an implicit time-stepping method is needed. Implicit methods solve linear systems with the "prediction" matrix $(I - h\gamma J)$, where h is the stepsize and γ a parameter determined by the method. In the naive approach one would have to solve linear algebraic systems of dimension $n(m+1) \times n(m+1)$ corresponding to Eq. (7). The direct-decoupled method (Dunker, 1984) exploits the special structure of the combined Jacobian (7); specifically, one only computes the LU factorization of the $n \times n$ model prediction matrix $I - h\gamma J = P^{\mathrm{T}} \cdot L \cdot U$. Then the $n(m+1) \times n(m+1)$ prediction matrix for Jacobian (7) has the LU factorization

$$\begin{pmatrix} P^{\mathrm{T}} \cdot L & 0 & \cdots & 0 \\ -h\gamma[(JS_1)_y + J_{p_1}]\, U^{-1} & P^{\mathrm{T}} \cdot L & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ -h\gamma[(JS_m)_y + J_{p_m}]\, U^{-1} & 0 & \cdots & P^{\mathrm{T}} \cdot L \end{pmatrix} \times \begin{pmatrix} U & 0 & \cdots & 0 \\ 0 & U & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & U \end{pmatrix}. \qquad (8)$$

The direct-decoupled method was developed and is traditionally presented in the context of BDF time-stepping schemes (Dunker, 1984; Caracotsios and Stewart, 1985; Leis and Kramer, 1986). In what follows we review this approach, then we extend the direct-decoupled philosophy to Runge–Kutta and Rosenbrock integrators.

2.1. Direct-decoupled backward differentiation formulas

Model (5) is approximated by the k-step, order k linear multistep formula

$$y^{i+1} = \sum_{j=0}^{k-1} \alpha_j y^{i-j} + h\beta f(t^{i+1}, y^{i+1}; p), \quad h = t^{i+1} - t^i. \quad (9)$$

The formula coefficients α_j, β are determined such that the method has order k of consistency. Relation (9) is a nonlinear system of equations which implicitly defines y^{i+1}, and which must be solved by a Newton-type iterative scheme. Typically, the solution $y^{i+1}_{\{q+1\}}$ at iteration $q+1$ is computed as

$$[I - h\beta J(t^i, y^i; p)](y^{i+1}_{\{q+1\}} - y^{i+1}_{\{q\}}) = \sum_{j=0}^{k-1} \alpha_j y^{i-j} + h\beta f(t^{i+1}, y^{i+1}_{\{q\}}; p) - y^{i+1}_{\{q\}}, \qquad (10)$$

which requires one factorization of $I - h\beta J$ and one backsubstitution per iteration.

Discretization of the continuous sensitivity equation: In this approach (Dunker, 1984; Caracotsios and Stewart, 1985; Leis and Kramer, 1986) one discretizes the continuous sensitivity equation (5) with the same BDF method used for discretizing model (9)

$$S_\ell^{i+1} = \sum_{j=0}^{k-1} \alpha_j S_\ell^{i-j} + h\beta J(t^{i+1}, y^{i+1}; p) S_\ell^{i+1} + h\beta f_{p_\ell}(t^{i+1}, y^{i+1}; p), \quad 1 \leqslant \ell \leqslant m. \qquad (11)$$

Note that system (11) is linear, therefore it admits a noniterative solution

$$[I - h\beta J(t^{i+1}, y^{i+1}; p)] S_\ell^{i+1} = \sum_{j=0}^{k-1} \alpha_j S_\ell^{i-j} + h\beta f_{p_\ell}(t^{i+1}, y^{i+1}; p), \quad 1 \leqslant \ell \leqslant m. \qquad (12)$$

The direct-decoupled method solves Eq. (10) first for the new-time solution y^{i+1}. The matrix $I - h\beta J(t^{i+1}, y^{i+1}; p)$ is computed and factorized; and systems Eq. (12) are solved for $\ell = 1$ through m. Note that all systems (12) use the same matrix factorization, and this factorization is also reused in Eq. (10) for computing y^{i+2} during the next time step. Therefore, the solution together with m sensitivities are obtained at the cost of a single matrix factorization per time step.

Discrete sensitivity equation: In this approach one considers directly the sensitivities of the numerical solution. A discrete equation involving these entities is obtained by taking the derivative of Eq. (9) with respect to p_ℓ ; it is easy to see that the process leads again to Eq. (11). Therefore, the numerical solutions of the variational equations S_ℓ^i are also the sensitivities of the numerical solution $\mathrm{d}y^i/\mathrm{d}p_\ell$.

2.2. Direct-decoupled Runge–Kutta methods

A general s-stage Runge–Kutta method is defined as (Hairer and Wanner, 1991, Section IV.3)

$$y^{i+1} = y^i + h \sum_{j=1}^{s} b_j k_j^0, \quad T^j = t^i + c_j h,$$

$$Y^j = y^i + h \sum_{q=1}^{s} a_{jq} k_q^0, \quad k_j^0 = f(T^j, Y^j; p), \qquad (13)$$

where the coefficients a_{jq}, b_j and c_j are prescribed for the desired accuracy and stability properties. The stage derivative values k_j^0 are defined implicitly, and require solving a (set of) nonlinear system(s). Newton-type methods solve coupled linear systems of dimension (at most) $n \times s$.

Discretization of the continuous sensitivity equation: Application of this method to the sensitivity equation (5) gives

$$S_\ell^{i+1} = S_\ell^i + h \sum_{j=1}^{s} b_j k_j^\ell \quad \text{for } 1 \leqslant \ell \leqslant m,$$

$$k_j^\ell = J(T^j, Y^j; p)\left(S_\ell^i + h \sum_{q=1}^{s} a_{jq} k_q^\ell\right) + f_{p_\ell}(T^j, Y^j; p). \qquad (14)$$

System (14) is linear and does not require an iterative procedure.

Discrete sensitivity equation: Clearly, Eq. (14) can be obtained by differentiating Eq. (13) with respect to p_ℓ and setting $k_j^\ell = \mathrm{d}k_j^0/\mathrm{d}p_\ell$, therefore the Runge–Kutta numerical solutions of the sensitivity equations are equal to the sensitivities of the Runge–Kutta numerical solution of the model equation.

Computational considerations: One first solves Eq. (13) for concentrations, which gives all stage derivative vectors $k_1^0, \ldots, k_s^0$. Next one solves Eq. (14) for sensitivities; each set $k_1^\ell, \ldots, k_s^\ell$ is the solution of an $(ns) \times (ns)$ linear system. The matrix of this system is the same for all $1 \leqslant \ell \leqslant m$, but is different than the matrix used in the Newton iteration that solves Eq. (13). One can reuse the matrix factorization of Eq. (13) if the linear systems (14) are solved by an iterative method. However, due to the repeated LU factorizations or the repeated Jacobian evaluations involved the standard Runge–Kutta methods do not seem suitable for direct sensitivity calculations.

2.3. Direct-decoupled Rosenbrock methods

An s-stage Rosenbrock method (Hairer and Wanner, 1991, Section IV.7) computes the next-step solution by the formulas

$$y^{i+1} = y^i + \sum_{j=1}^{s} b_j k_j^0, \quad T^j = t^i + \alpha_j h,$$

$$Y^j = y^i + \sum_{q=1}^{j-1} a_{jq} k_q^0,$$

$$\left[\frac{1}{h\gamma_{jj}} I - J(t^i, y^i; p)\right] k_j^0 = f(T^j, Y^j; p) + \sum_{q=1}^{j-1} \frac{c_{jq}}{h} k_q^0 + h\gamma_j \frac{\partial f}{\partial t}(t^i, y^i; p), \qquad (15)$$

where s is the number of stages. The formula coefficients (b_j, a_{jq}, c_{jq}, α_j, γ_{jj}, and γ_j) give the order of consistency and the stability properties. At each stage of the method, a linear system of equations with unknowns k_j^0 and matrix $1/(h\gamma_{jj})I - J$ must be solved. Methods for which $\gamma_{11} = \cdots = \gamma_{ss}$ are of particular interest since in this case only one LU matrix decomposition is required per integration step (Sandu et al., 1997b, Verwer et al., 1999, Djouad et al., 2002). Form (15) is advantageous for implementation purposes and is equivalent with the standard formulation (Hairer and Wanner, 1991, Section IV.7).

Discretization of the continuous sensitivity equation: To apply method (15) to the combined sensitivity equations (1)–(5), we need to make explicit use of the combined Jacobian (7). One step of the method updates the solution with Eq. (15) and the sensitivities using

$$S_\ell^{i+1} = S_\ell^i + \sum_{j=1}^{s} b_j k_j^\ell \quad \text{for } 1 \leqslant \ell \leqslant m, \qquad (16)$$

$$\begin{aligned}\left[\frac{1}{h\gamma_{jj}} I - J(t^i, y^i; p)\right] k_j^\ell &= J(T^j, Y^j; p)\left(S_\ell^i + \sum_{q=1}^{j-1} a_{jq} k_q^\ell\right) + f_{p_\ell}(T^j, Y^j; p) \\ &\quad + \sum_{q=1}^{j-1} \frac{c_{jq}}{h} k_q^\ell \\ &\quad + \left(\frac{\partial J}{\partial p_\ell}(t^i, y^i; p)\right) k_j^0 + \left(\frac{\partial J}{\partial y}(t^i, y^i; p) \times S_\ell^i\right) k_j^0 \\ &\quad + h\gamma_j J_t(t^i, y^i; p) S_\ell^i + h\gamma_j f_{p_\ell,t}(t^i, y^i; p),\end{aligned}$$

where $J_t = \partial J/\partial t, f_{p_\ell,t} = \partial f_{p_\ell}/\partial t$. If $\gamma_{11} = \cdots = \gamma_{ss}$, a single $n \times n$ matrix LU decomposition is required per step to obtain both the concentrations and the sensitivities.

Discrete sensitivity equation: We note that the derivative of method (15) with respect to p_ℓ leads also to Eq. (16). Consequently, the sensitivities of the numerical solution coincide with the numerical solutions of the sensitivity equations.

Computational considerations: Formula (16) requires the evaluation of the Hessian, i.e. the derivatives of the Jacobian with respect to y, as well as the Jacobian derivatives with respect to the parameters; these entities are 3-tensors. In addition, an extra Jacobian evaluation and one Jacobian-vector multiplication is needed at each stage. The need for Jacobian derivatives prevented Rosenbrock methods to be used extensively in direct sensitivity calculations. However, the new automatic differentiation and symbolic differentiation technologies make the computation of these derivatives feasible.

To avoid computing the derivatives of the Jacobian one can approximate Jacobian (7) by diag$(J, \ldots, J)$ and use a Rosenbrock-W integration formula (Hairer and

Wanner, 1991, Section IV.7). The resulting method gives consistent approximations of the sensitivities of the continuous solution, but these are now different than the sensitivities of the numerical solution.

Remark 1. Seefeld and Stockwell (1999) present an extension of the direct-decoupled method to systems with time-varying parameters. In their approach, a time-dependent parameter is expressed as the product of a time-varying profile and a time-independent multiplier: $p_j(t) = p_j^*(t) \times q_j$ with the nominal value of the multiplier chosen to be unity. The direct-decoupled method is then used to compute the derivatives of the concentrations with respect to the time-independent multiplier.

3. Adjoint sensitivity analysis

The direct-decoupled method is known to be very effective when the sensitivities of a large number of output variables are computed with respect to a small number of input parameters. The adjoint method provides an efficient alternative to the direct-decoupled method for evaluating the sensitivity of a scalar response function with respect to the initial conditions and model parameters. Mathematical foundations of the adjoint sensitivity for nonlinear dynamical systems and various classes of response functionals are presented by Cacuci (1981a, b). The construction of the adjoint operators associated with linear and nonlinear dynamics and applications to atmospheric modeling are described in detail by Marchuk (1995) and Marchuk et al. (1996). Menut et al. (2000) and Vautard et al. (2000) use the adjoint modeling for sensitivity studies in atmospheric chemistry. A review of the adjoint method applied to four-dimensional variational atmospheric chemistry data assimilation is presented in the work of Wang et al. (2001).

We will refer to the dynamical model (1) as a forward (direct) model. Given a scalar response function

$$g = g(y(t^{\mathrm{F}}, p)) \tag{17}$$

we are interested to evaluate the sensitivities

$$\frac{\partial g}{\partial p_\ell} = \sum_{j=1}^{n} \left(\frac{\partial g}{\partial y_j} \frac{\partial y_j}{\partial p_\ell} \right)_{(t=t^{\mathrm{F}})} = \left\langle \frac{\partial g}{\partial y}, S_\ell \right\rangle_{n(t=t^{\mathrm{F}})} \quad \text{for } 1 \leqslant \ell \leqslant m, \tag{18}$$

where $\langle \cdot, \cdot \rangle_n$ denotes the inner product in $\mathscr{R}^n$, $\langle u, v \rangle_n = v^{\mathrm{T}} u$.

In the typical case where the cost functional is given by a time integral (3), the sensitivity problem (3)–(4) may be reduced to problem (17)–(18) by augmenting the state vector y with a new component y_{n+1} whose time evolution is governed by the equations

$$\frac{\mathrm{d}y_{n+1}}{\mathrm{d}t} = \hat{g}(y(t,p)), \quad y_{n+1}(t^0) = 0. \tag{19}$$

Then

$$y_{n+1}(t^{\mathrm{F}}, p) = \int_{t^0}^{t^{\mathrm{F}}} \hat{g}(y(t,p))\,\mathrm{d}t \tag{20}$$

and the adjoint sensitivity is applied to the augmented system (1)–(19), with the state vector $Y = (y^{\mathrm{T}}, y_{n+1})^{\mathrm{T}}$ and the response functional $g = y_{n+1}(t^{\mathrm{F}}, p)$.

In the adjoint sensitivity one must distinguish between the continuous and the discrete adjoint modeling (Sirkes and Tziperman, 1997). While in general the derivation of the continuous adjoint model is presented, often in practice it is a discrete adjoint model that is implemented. This distinction is of particular importance in the context of stiff chemical reactions systems that require sophisticated numerical integrators.

3.1. Continuous adjoint sensitivity

Traditionally, the adjoint model equations are derived using the Hilbert spaces theory and variational techniques (Wang et al., 2001). The continuous adjoint model is obtained from the continuous forward model using the linearization technique. To first-order approximation, a perturbation δp in the input parameters leads to a perturbation in the response functional

$$\delta g = g(y(t^{\mathrm{F}}, p+\delta p)) - g(y(t^{\mathrm{F}}, p)) = \left\langle \frac{\partial g}{\partial y}, \delta y \right\rangle_{n(t=t^{\mathrm{F}})}, \tag{21}$$

where the state perturbation $\delta y(t^{\mathrm{F}})$ is obtained by solving the tangent linear model problem

$$\frac{\mathrm{d}\delta y}{\mathrm{d}t} = J(t,y;p)\delta y + f_p(t,y;p)\delta p, \quad t^0 \leqslant t \leqslant t^{\mathrm{F}}, \tag{22}$$

$$\delta y(t^0) = \delta y^0. \tag{23}$$

In the direct sensitivity approach, for each parameter variation $\delta p_l, 1 \leqslant l \leqslant m$, a new system (22)–(23) must be solved to compute $\delta y(t^{\mathrm{F}})$. The adjoint model is used to express explicitly the perturbation δg in terms of parameter variations δp as follows: we introduce the adjoint variable $\lambda(t) \in \mathscr{R}^n$ (to be precisely defined later), take the scalar product of Eq. (22) with λ and integrate on $[t^0, t^{\mathrm{F}}]$ to obtain:

$$\int_{t^0}^{t^{\mathrm{F}}} \left\langle \lambda, \frac{\mathrm{d}\delta y}{\mathrm{d}t} \right\rangle_n \mathrm{d}t = \int_{t^0}^{t^{\mathrm{F}}} \langle \lambda, J(t,y;p)\delta y + f_p(t,y;p)\delta p \rangle_n \,\mathrm{d}t. \tag{24}$$

Integrating by parts the left-hand side of Eq. (24), using matrix transposition on the right-hand side, and rearranging the terms give the equivalent formulation

$$\langle \lambda, \delta y \rangle_n \big|_{t^0}^{t^F} = \int_{t^0}^{t^F} \left\langle \frac{d\lambda}{dt} + J^T(t, y; p)\lambda, \delta y \right\rangle_n + \langle f_p^T(t, y; p)\lambda, \delta p \rangle_m \, dt. \tag{25}$$

Therefore, if λ is defined as the solution of the adjoint problem

$$\frac{d\lambda}{dt} = -J^T(t, y; p)\lambda, \tag{26}$$

$$\lambda(t^F) = \frac{\partial g}{\partial y}(t^F), \tag{27}$$

the perturbation in the response functional can be expressed from Eqs. (21), (23), and (25) as

$$\delta g = \langle \lambda(t^0), \delta y^0 \rangle_n + \int_{t^0}^{t^F} \langle f_p^T(t, y; p)\lambda, \delta p \rangle_m \, dt. \tag{28}$$

Evaluation of $\lambda(t)$, $t^0 \leqslant t \leqslant t^F$ requires *only one forward integration* from t^0 to t^F of model (1) to compute $y(t), t^0 < t \leqslant t^F$, followed by *a single backward integration* from t^F to t^0 of the n-dimensional adjoint model (26)–(27). $\lambda(t^0) \in \mathscr{R}^n$ represent the sensitivities of the response g with respect to the initial conditions y^0. Having available $\lambda(t)$, sensitivities with respect to additional model parameters p_l are

$$\frac{\partial g}{\partial p_l} = \int_{t^0}^{t^F} \langle \lambda(t), f_{p_l}(t, y; p) \rangle_n \, dt = \int_{t^0}^{t^F} f_{p_l}^T(t, y; p)\lambda(t) \, dt \tag{29}$$

and may be computed using an appropriate numerical quadrature scheme.

Remark 2. Modeling chemical kinetic systems requires the specification of time-dependent model parameters. For example, photolytic reaction rates are determined by the solar radiation and thermal reactions rates depend on the temperature, therefore the reaction rates are implicitly a function of time. The adjoint modeling may be used to evaluate the sensitivity of the response functional with respect to time-dependent model parameters. The adjoint functions $\lambda(t)$ are also called *influence functions* and represent the sensitivity of the response functional with respect to the variations in the model state at time t; if the parameters are time dependent, $p = p(t)$, the *impulse response functions* $f_{p_\ell}^T(t, y; p(t))\lambda(t)$ represent the variation in g due to a unit impulse (Dirac function) in the parameter p_ℓ at time t (Bryson and Ho, 1975).

Remark 3. In practice, it is often of interest to evaluate the sensitivity of multiple response functionals (e.g. concentrations of various pollutants at a given time) with respect to the model parameters. One should notice that in the direct approach a new set of Eqs. (5) must be solved for each additional parameter, whereas in the adjoint approach a new set of adjoint equations (26)–(27) must be solved for each additional response. The sensitivities of a vector-valued function $G(y(t^F)) \in R^k$ may be obtained by a backward integration of the adjoint model

$$\frac{d\Lambda}{dt} = -J^T(t, y; p)\Lambda, \tag{30}$$

$$\Lambda(t^F) = \left(\frac{\partial G}{\partial y}\right)^T (t^F), \tag{31}$$

where $\Lambda(t) \in R^{n \times k}$ are the adjoint variables. The matrix approach to the adjoint sensitivity analysis of atmospheric models with multiple responses is discussed by Ustinov (2001).

3.2. The relationship with the Green's function method

The adjoint method presented above can be related to the Green's function. The Green's function method of sensitivity analysis in chemical kinetics is discussed in detail by Hwang et al. (1978), Dougherty et al. (1979), and Vuilleumier et al. (1997). The Green's function matrix $K(t, \tau) \in \mathscr{R}^{n \times n}$ associated to Eq. (5) satisfies

$$\frac{d}{dt}K(t, \tau) - J(t, y; p)K(t, \tau) = 0, \quad t > \tau, \tag{32}$$

$$K(\tau, \tau) = I. \tag{33}$$

The sensitivities $S_\ell(t)$ may be expressed in terms of the Green's function (Hwang et al., 1978)

$$S_\ell(t) = K(t, t^0)S_\ell(t^0) + \int_{t^0}^{t} K(t, \tau) f_{p_\ell}(\tau) \, d\tau, \quad 1 \leqslant \ell \leqslant m. \tag{34}$$

To obtain $S_\ell(t^F)$, the differential equations for the adjoint Green's function

$$\frac{d}{d\tau}K^*(\tau, t^F) + K^*(\tau, t^F)J(\tau, y; p) = 0, \quad \tau < t^F, \tag{35}$$

$$K^*(t^F, t^F) = I \tag{36}$$

are solved backwards in time from t^F to t^0 and using the identity

$$K^*(\tau, t) = K(t, \tau), \tag{37}$$

the right-hand side integrals in Eq. (34) are computed with an appropriate numerical quadrature scheme (Hwang et al., 1978; Dougherty et al. 1979). In this way, all $S_\ell(t^F), 1 \leqslant l \leqslant m$ may be evaluated by solving $n+1$ sets (Eqs. (1) and(35)–(36)) of stiff ODE's of dimension n.

A fundamental observation is that to evaluate sensitivities (18) explicit knowledge of $S_\ell(t^F), 1 \leqslant l \leqslant m$ is not needed; only the inner product

$\langle \partial g/\partial y(t^{\mathrm{F}}), S_\ell(t^{\mathrm{F}}) \rangle_n$ is required. From Eqs. (18) and (34), we obtain

$$\begin{aligned}\frac{\partial g}{\partial p_\ell} &= \left\langle \frac{\partial g}{\partial y}(t^{\mathrm{F}}), S_\ell(t^{\mathrm{F}}) \right\rangle_n \\ &= \left\langle \frac{\partial g}{\partial y}(t^{\mathrm{F}}), K(t^{\mathrm{F}}, t^0) S_\ell(t^0) \right\rangle_n \\ &\quad + \int_{t^0}^{t^{\mathrm{F}}} \left\langle \frac{\partial g}{\partial y}(t^{\mathrm{F}}), K(t^{\mathrm{F}}, \tau) f_{p_\ell}(\tau, y; p) \right\rangle_n \mathrm{d}\tau, \qquad (38)\end{aligned}$$

which may be written using Eq. (37) and matrix transposition in the equivalent form

$$\begin{aligned}\frac{\partial g}{\partial p_\ell} &= \left\langle K^{*\mathrm{T}}(t^0, t^{\mathrm{F}}) \frac{\partial g}{\partial y}(t^{\mathrm{F}}), S_\ell(t^0) \right\rangle_n \\ &\quad + \int_{t^0}^{t^{\mathrm{F}}} \left\langle K^{*\mathrm{T}}(\tau, t^{\mathrm{F}}) \frac{\partial g}{\partial y}(t^{\mathrm{F}}), f_{p_\ell}(\tau, y; p) \right\rangle_n \mathrm{d}\tau. \qquad (39)\end{aligned}$$

If we define the *adjoint variables* $\lambda(\tau) \in \mathscr{R}^n$ as

$$\lambda(\tau) = K^{*\mathrm{T}}(\tau, t^{\mathrm{F}}) \frac{\partial g}{\partial y}(t^{\mathrm{F}}), \qquad (40)$$

then Eq. (39) is written in a form that reveals the similarities with Eq. (28)

$$\frac{\partial g}{\partial p_\ell} = \langle \lambda(t^0), S_\ell(t^0) \rangle_n + \int_{t^0}^{t^{\mathrm{F}}} \langle \lambda(\tau), f_{p_\ell}(\tau, y; p) \rangle_n \, \mathrm{d}\tau, \quad 1 \leqslant l \leqslant m. \qquad (41)$$

From Eqs. (35)–(36) and definition (40) it follows that $\lambda(\tau)$ is the solution of the *adjoint model* problem (26)–(27).

3.3. Discrete adjoint sensitivity

The discretization of system (1) with a selected numerical method provides a numerical approximation $y^N \approx y(t^{\mathrm{F}})$ through a sequence of N intermediate states

$$y^{i+1} = F^i(y^i; p), \quad i = 0, \ldots, N-1, \qquad (42)$$

where F^i represents a one-step numerical integration formula which advances the solution from t^i to t^{i+1}. This establishes an explicit relationship between the evaluated response functional $g(y^N)$ and the model parameters. Using interpolation techniques, we may assume that the parameters p are discrete (time independent) and are determined by their values at the interpolation nodes.

The discrete adjoint model equations are obtained directly from the discrete forward model equations (42) as we now explain. To present a compact, yet explicit derivation of the discrete adjoint sensitivity formulae, we consider an augmented state vector $Y(t) = (y^{\mathrm{T}}(t), p^{\mathrm{T}}(t))^{\mathrm{T}}$, where $p(t) \equiv p$. We rewrite the discrete equations (42) as

$$y^{i+1} = F^i(Y^i), \quad i = 0, \ldots, N-1, \qquad (43)$$

and attach the parameters equations

$$p^0 = p, \qquad p^{i+1} = p^i, \quad i = 0, \ldots, N-1. \qquad (44)$$

The sensitivity with respect to the initial conditions $Y^0 = ((y^0)^{\mathrm{T}}, (p^0)^{\mathrm{T}})^{\mathrm{T}}$ is given by

$$\begin{pmatrix} \dfrac{\partial g(y^N)}{\partial y^0} \\ \dfrac{\partial g(y^N)}{\partial p} \end{pmatrix} = \frac{\partial g(y^N)}{\partial Y^0} = \left(\frac{\partial Y^N}{\partial Y^0}\right)^{\mathrm{T}} \begin{pmatrix} \dfrac{\partial g}{\partial y}(y^N) \\ 0_m \end{pmatrix}. \qquad (45)$$

A successive application of the chain rule followed by transposition gives

$$\left(\frac{\partial Y^N}{\partial Y^0}\right)^{\mathrm{T}} = \left(\frac{\partial Y^1}{\partial Y^0}\right)^{\mathrm{T}} \left(\frac{\partial Y^2}{\partial Y^1}\right)^{\mathrm{T}} \cdots \left(\frac{\partial Y^{N-1}}{\partial Y^{N-2}}\right)^{\mathrm{T}} \times \left(\frac{\partial Y^N}{\partial Y^{N-1}}\right)^{\mathrm{T}}, \qquad (46)$$

and by differentiating equations (43)–(44) we obtain

$$\left(\frac{\partial Y^{i+1}}{\partial Y^i}\right) = \begin{pmatrix} F_y^i(y^i, p^i) & F_p^i(y^i, p^i) \\ 0_{(m,\,n)} & I_{(m,\,m)} \end{pmatrix}, \quad i = 0, \ldots, N-1. \qquad (47)$$

Therefore, if we define the adjoint variables at $t^N = t^{\mathrm{F}}$

$$\lambda^N = \frac{\partial g}{\partial y}(y^N), \quad v^N = 0_m \qquad (48)$$

and evaluate the adjoint variables at $t^i, i = N-1, \ldots, 1, 0$ using the recursive relations

$$\begin{pmatrix} \lambda^i \\ v^i \end{pmatrix} = \left(\frac{\partial Y^{i+1}}{\partial Y^i}\right)^{\mathrm{T}} \begin{pmatrix} \lambda^{i+1} \\ v^{i+1} \end{pmatrix} = \begin{pmatrix} F_y^i(y^i, p^i)^{\mathrm{T}} \lambda^{i+1} \\ F_p^i(y^i, p^i)^{\mathrm{T}} \lambda^{i+1} + v^{i+1} \end{pmatrix}, \qquad (49)$$

we obtain from Eqs. (45) to (49) the sensitivities

$$\frac{\partial g(y^N)}{\partial y^0} = \lambda^0, \quad \frac{\partial g(y^N)}{\partial p} = v^0. \qquad (50)$$

3.4. Practical issues of the adjoint code implementation

The adjoint method relies on the linearization of the forward model and the nonlinearity introduced by the chemistry may have a direct impact on the time interval length and the qualitative aspects of the adjoint sensitivity analysis. Since for nonlinear problems the adjoint equations depend on the forward model state, in order to perform the adjoint computations the forward model state must be available in reverse order. Therefore, a large amount of memory must be allocated for storing the state during the forward run.

As noticed in the previous sections in practice two strategies may be used to implement the adjoint code.

In the continuous approach, the continuous adjoint model is first derived from the linearized continuous forward model equations; then a numerical method of choice is used to integrate the adjoint model. In this approach, the complexity of the numerical method used during the forward integration does not interfere with the adjoint computations. While during the forward integration one has to solve a *stiff nonlinear* ODE system, during the adjoint integration a *stiff linear* system of ODE's must be solved. Therefore, highly stable implicit methods may be efficiently implemented as no iterations are needed for solving the adjoint.

The second method is the discrete adjoint approach, where the explicit dependence of the state vector evolution on the input parameters is obtained by the numerical integration of the forward model. The discrete adjoint model is then derived directly from the linearized discrete forward model equations. In this way, the discrete adjoint model provides an exact gradient (sensitivity) relative to the numerical computation of the response functional. This approach appears to be more suitable for the variational data assimilation where a minimization process must be performed since the minimization routine will receive the exact gradient of the evaluated cost function. However, implementing an efficient discrete adjoint code may be a difficult task if sophisticated numerical methods are used. Additional issues related with the consistency of a numerical scheme and the consistency of its adjoint are discussed by Sei and Symes (1995).

Hand-generated adjoint codes are tedious to write and often subject to errors. Automatic differentiation tools (Giering and Kaminski, 1998; Rostaing et al., 1993) may facilitate the adjoint code generation, but they must be used with caution and the correctness of the automatic-generated adjoint code must be carefully verified. The state of the art stiff ODE solvers are usually written in a form which is not optimal for the adjoint code generation and often one has to rewrite them in a form that is suitable for the adjoint compilers.

There is no general rule to decide which approach (discrete vs. continuous) should be used to implement the adjoint model. The performance of the adjoint code is problem dependent and a selection can be made only after an extensive analysis and testing for the particular problem to be solved have been performed.

For atmospheric chemistry data assimilation and sensitivity studies, a discrete adjoint model was used by Fisher and Lary (1995) for a Burlich–Stoer integration scheme (Stoer and Burlich, 1980) and by Elbern et al. (1997) for a quasi-steady-state approximation (QSSA) method. A continuous adjoint chemistry model with a fourth-order Rosenbrock solver (Hairer and Wanner, 1991) for the forward/backward integration was successfully applied to 4D-Var chemical data assimilation by Errera and Fonteyn (2001) in a hybrid adjoint approach (discrete adjoint for the transport integration, continuous adjoint for the chemistry integration).

4. The KPP tools

In this section, we present the KPP software tools that are useful in derivative computations. A detailed discussion of the basic KPP capabilities can be found in our previous work (Damian-Iordache et al., 1995; 2002). Here, we focus on the new features introduced in the release 1.2 that allow an efficient sensitivity analysis of chemical kinetic systems.

KPP builds simulation code for chemical systems driven by the law of mass action kinetics

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}y &= S\cdot \mathrm{diag}[k_1(t),\ldots,k_R(t)]\cdot p(y)\\ &= S\cdot\omega(t,y) = f(t,y),\end{aligned} \qquad (51)$$

where S is the stoichiometric matrix, $k_i(t)$ the ith reaction rate coefficient, $p=[p_1,\ldots,p_R]^{\mathrm{T}}$ the vector of reactant products and $\omega=[\omega_1,\ldots,\omega_R]^{\mathrm{T}}$ the vector of reaction velocities.

4.1. The derivative function

KPP orders the variable species such that the sparsity pattern of the Jacobian is maintained after an LU decomposition and generates the following function to compute the vector A_VAR of component time derivatives from the concentrations of variable V, radical R and fixed F species, and the vector of rate coefficients RCT.

```
SUBROUTINE FunVar ( V, R, F, RCT, A_VAR ).
```

4.2. The Jacobian

The Jacobian of the derivative function is also constructed by KPP via the command

```
#JACOBIAN [ OFF | ON | SPARSE ].
```

The option OFF inhibits the generation of the Jacobian subroutine; the option ON generates the Jacobian in full matrix format, while the option SPARSE generates the following routine for the Jacobian in sparse row-compressed format

```
SUBROUTINE JacVar_SP( V, R, F, RCT, JS ).
```

Implicit numerical integrators solve systems of the form $(I-h\gamma J)x=b$, where h is the step size, γ a method-dependent parameter, J the Jacobian, and $I-h\gamma J$ the "prediction" matrix. KPP generates the following sparse linear algebra subroutines:

```
SUBROUTINE KppDecomp(N,P,IER)
```

performs an in-place, nonpivoting, sparse LU decomposition of the prediction matrix P (IER returns a nonzero value if singularity is detected). Using this factorization the sparse backward and forward substitutions are performed by

```
SUBROUTINE KppSolve(P,X).
```

Similarly, a new subroutine required for the adjoint computations

```
SUBROUTINE KppSolveTR ( P, b, X )
```

solves the linear system $P^{\mathrm{T}}X = b$ with the transposed coefficient matrix, and uses the same LU factorization as KppSolve. The sparse subroutines KppDecomp and KppSolve are extremely efficient (Sandu et al., 1996).

Two other KPP-generated subroutines are useful for direct and adjoint sensitivity analysis

```
SUBROUTINE JacVar_SP_Vec ( JVS, U, V )
```

computes the sparse Jacobian times vector product ($V \leftarrow \mathrm{JVS} \cdot U$), and the subroutine

```
SUBROUTINE JacVarTR_SP_Vec ( JVS, U, V )
```

computes the sparse Jacobian transposed times vector product ($V \leftarrow \mathrm{JV\,S}^{\mathrm{T}} \cdot U$).

4.3. The stoichiometric formulation

KPP can generate the elements of the derivative function and the Jacobian in the stoichiometric formulation (51); this means that the product between the stoichiometric matrix, rate coefficients, and reactant products is not explicitly performed. The option that controls the code generation is

```
#STOICMAT [ OFF | ON ].
```

The `ON` value of the switch instructs KPP to generate code for the stoichiometric matrix, the vector of reactant products in each reaction, and the partial derivative of the time derivative function with respect to rate coefficients. These elements are discussed below.

The stoichiometric matrix is usually very sparse; the total number of nonzero entries in the stoichiometric matrix is the constant `NSTOICM` in KPP-generated code. KPP produces the stoichiometric matrix in sparse, column-compressed format. Elements are stored in columnwise order in the one-dimensional vector of values `STOICM(1:NSTOICM)`; their row indices are stored in `IROW_STOICM(1:NSTOICM)`; the vector `CCOL_STOICM(1:NVAR+1)` contains pointers to the start of each column; for example, column j starts in the sparse vector at position `CCOL_STOICM(`j`)` and ends at `CCOL_STOICM(`$j+1$`)` -1. The last value `CCOL_STOICM(NVAR` $+1$`)` $=$ `NSTOICM` $+$ `1` is not necessary but simplifies the future handling of sparse data structures.

The following subroutine computes the reactant products for each reaction, i.e. `A_RPROD` is the vector $p(y)$ in formulation (51).

```
SUBROUTINE ReactantProd ( V, R, F, A_RPROD ).
```

In the stoichiometric formulation (51) the Jacobian is

$$J(t,y) = \frac{\partial f(t,y)}{\partial y} = S \cdot \mathrm{diag}[k_1(t), \ldots, k_R(t)] \cdot \frac{\partial p(y)}{\partial y} = S \cdot \mathrm{diag}[k_1(t), \ldots, k_R(t)] \cdot J^{\mathrm{RP}}(y). \tag{52}$$

The following subroutine computes the Jacobian of reactant products vector; i.e. `JV_RPROD` is the matrix $J^{\mathrm{RP}} = \partial p(y)/\partial y$ above

```
SUBROUTINE JacVarReactantProd
  ( V, R, F, JV_RPROD ).
```

The matrix `JV_RPROD` is of course sparse and is computed and stored in row-compressed sparse format. The number of nonzeros is stored in the parameter `NJVRP`, the column indices in the vector `ICOL_JVRP` and the beginning of each row in `CROW_JVRP`.

4.4. The derivatives with respect to reaction coefficients

The stoichiometric formulation allows a direct computation of the derivatives with respect to rate coefficients. From Eq. (51) one sees that the partial derivative of the time derivative function with respect to a reaction coefficient is given by the corresponding column in the stoichiometric matrix times the corresponding entry in the vector of reactant products

$$f(t,y) = S \cdot \mathrm{diag}[k_1(t), \ldots, k_R(t)] \cdot p(y) \Rightarrow \frac{\partial f_{1:\mathtt{NVAR}}}{\partial k_j} = S_{1:\mathtt{NVAR},j} \cdot p_j(y).$$

The following subroutine computes the partial derivative of the time derivative function with respect to a set of reaction coefficients; this is more efficient than computing each derivative separately.

```
SUBROUTINE dFunVar_dRcoeff
  ( V, R, F, NCOEFF, JCOEFF, DFDR ).
```

A total of `NCOEFF` derivatives are taken with respect to reaction coefficients `JCOEFF(1)–JCOEFF(NCOEFF)`. `JCOEFF`, therefore, is a vector of integers containing the indices of reaction coefficients with respect to which we differentiate. The subroutine returns the `NVAR` $\times$ `NCOEFF` matrix `DFDR`; each column of this matrix contains the derivative of the function with respect to one rate coefficient. Specifically,

$$\mathtt{DFDR}_{1:\mathtt{NVAR},j} = \frac{\partial f_{1:\mathtt{NVAR}}}{\partial k_{\mathtt{JCOEFF(j)}}}, \quad 1 \leqslant j \leqslant \mathtt{NCOEFF}.$$

The partial derivative of the Jacobian with respect to the rate coefficient k_j can be obtained from Eq. (52) as the external product of column j of the stoichiometric matrix with row j of J^{RP}

$$\frac{\partial J_{1:\mathtt{NVAR},1:\mathtt{NVAR}}}{\partial k_j} = S_{1:\mathtt{NVAR},j} \cdot J^{\mathrm{RP}}_{j,1:\mathtt{NVAR}}.$$

In practice, one needs the product of this Jacobian partial derivative with a user vector, i.e.

$$\frac{\partial J_{1:\mathtt{NVAR},1:\mathtt{NVAR}}}{\partial k_j} \cdot U_{1:\mathtt{NVAR}} = S_{1:\mathtt{NVAR},j} \cdot (J^{\mathrm{RP}}_{j,1:\mathtt{NVAR}} \cdot U_{1:\mathtt{NVAR}}).$$

This is computed by the KPP-generated subroutine

```
SUBROUTINE dJacVar_dRcoeff
  ( V, R, F, U, NCOEFF, JCOEFF, DJDR ).
```

`U` is the user-supplied vector. A total of `NCOEFF` derivatives are taken with respect to reaction coefficients `JCOEFF(1)–JCOEFF(NCOEFF)`. The subroutine returns the `NVAR`×`NCOEFF` matrix `DJDR`; each column of this matrix contains the derivative of the Jacobian with respect to one rate coefficient times the user vector. Specifically,

$$\mathtt{DJDR}_{1:\mathtt{NVAR},j} = \frac{\partial J_{1:\mathtt{NVAR},1:\mathtt{NVAR}}}{\partial k_{\mathtt{JCOEFF(j)}}} \cdot U_{1:\mathtt{NVAR}}, \quad 1 \leqslant j \leqslant \mathtt{NCOEFF}.$$

4.5. The Hessian

The Hessian contains second-order derivatives of the time derivative functions. More exactly, the Hessian is a 3-tensor such that

$$H_{i,j,k}(t,y;p) = \frac{\partial^2 f_i(t,y_1,\ldots,y_n;p_1,\ldots,p_m)}{\partial y_j \partial y_k}, \quad 1 \leqslant i,j,k \leqslant n. \tag{53}$$

For each component i there is a Hessian matrix $H_{i,:,:}$, since the time derivative function is smooth these Hessian matrices are symmetric

$$H_{i,j,k}(t,y;p) = \frac{\partial^2 f_i(t,y;p)}{\partial y_j \partial y_k} = \frac{\partial^2 f_i(t,y;p)}{\partial y_k \partial y_j} = H_{i,k,j}(t,y;p). \tag{54}$$

An alternative way to look at the Hessian is to consider the 3-tensor as the derivative of the Jacobian (6) with respect to individual species concentrations

$$H_{i,j,k} = \frac{\partial J_{i,j}(t,y_1,\ldots,y_n;p)}{\partial y_k}, \qquad J_{i,j} = \frac{df_i(t,y;p)}{dy_j}, \quad 1 \leqslant i,j,k \leqslant n. \tag{55}$$

Clearly, the Hessian is a very sparse tensor (considerably sparser than the Jacobian). KPP computes the number of nonzero Hessian entries (and saves this in the variable NHESS). The Hessian itself is represented in coordinate sparse format; the real vector HESS holds the values, and the integer vectors IHESS_$\{I,J,K\}$ the indices of nonzero entries

```
HESS =[ 1, 2, ..., NHESS ] IHESS_{I,J,K}
   =[ 1, 2, ..., NHESS ]
```

such that the nonzero Hessian entries are stored as

$$H_{i,j,k} = \mathrm{HESS}(m), \quad \{i,j,k\} = \mathrm{IHESS\_}\{I,J,K\}(m) \quad \text{for } H_{i,j,k} \neq 0 \text{ and } 1 \leqslant m \leqslant \mathrm{NHESS}.$$

The sparsity coordinate vectors IHESS_$\{I,J,K\}$ are computed by KPP and initialized statically; these vectors are constant as the sparsity pattern of the Hessian does not change during the computation.

Note that due to the symmetry relation (54) it is enough to store only the upper part of each component's Hessian matrix; the sparse structures for the Hessian store only those entries $H_{i,j,k}$ for which $j \leqslant k$.

The subroutine

```
SUBROUTINE HessVar ( V, R, F, HESS )
```

computes the Hessian (in coordinate sparse format) given the concentrations of the variable species V. Note that only the entries $H_{i,j,k}$ with $j \leqslant k$ are computed and stored.

The subroutine

```
SUBROUTINE HessVar_Vec ( HESS, U1, U2, HU )
```

computes the Hessian times user vectors product, which is a vector and can be regarded as the derivative of Jacobian-vector product times vector

$$\mathrm{HU} \leftarrow (\mathrm{HESS} \times U2) \cdot U1 = \frac{\partial (J(y) \cdot U1)}{\partial y} \cdot U2.$$

Similarly, the subroutine

```
SUBROUTINE HessVarTR_Vec ( HESS, U1, U2, HTU ).
```

computes the Hessian transposed times user vectors product (which equals the derivative of Jacobian transposed-vector product times vector)

$$\mathrm{HTU} \leftarrow (\mathrm{HESS} \times U2)^{\mathrm{T}} \cdot U1 = \frac{\partial (J^{\mathrm{T}}(y) \cdot U1)}{\partial y} \cdot U2.$$

5. Building direct-decoupled code with KPP

In this section, we present a tutorial example for the direct-decoupled code implementation in the KPP framework. For simplicity, we consider the problem of evaluating the sensitivities $S_\ell(t) \in \mathscr{R}^n$ of the model state (concentrations) at time moments t, $t^0 \leqslant t < t^{\mathrm{F}}$, with respect to the concentration of a species ℓ at a previous time instance t^0

$$S_\ell(t) = \frac{\partial[y_1(t),\ldots,y_n(t)]}{\partial y_\ell(t^0)}, \quad \ell = 1,\ldots,n. \tag{56}$$

The forward numerical integration, $y^i \to y^{i+1}$, is performed using the first-order linearly implicit Euler method (Hairer and Wanner, 1991, Section IV.9)

$$\left(J(t^i,y^i) - \frac{1}{h}I\right)(y^i - y^{i+1}) = f(t^i,y^i),$$
$$i = 0,\ldots,N-1, \tag{57}$$

with a constant step size h; the final time is reached for $t^N = t^0 + Nh = t^F$. The implementation of one forward integration step requires the solution of a system with the sparse Jacobian, and is readily implemented using KPP-generated routines.

To calculate sensitivities with respect to initial values we follow formulas (16) in the one-stage, autonomous form. The Hessian is denoted by $H = \partial J/\partial y$.

$$y^{i+1} = y^i - k^0, \qquad S_\ell^{i+1} = S_\ell^i - k^\ell \quad \text{for } 1 \leqslant \ell \leqslant m,$$

$$\left(J(t^i,y^i) - \frac{1}{h}I\right)k^0 = f(t^i,y^i),$$

$$\left(J(t^i,y^i) - \frac{1}{h}I\right)k^\ell = J(t^i,y^i)S_\ell^i + (H(t^i,y^i) \times S_\ell^i)k^0$$
$$\text{for } 1 \leqslant \ell \leqslant m.$$

For the implementation one needs the following KPP-generated routines: sparse Jacobian decomposition and substitution, sparse Jacobian times vector multiplication, and sparse Hessian times vectors multiplication. Note that an implementation of the direct-decoupled code for computing sensitivities with respect to rate coefficients can be easily obtained following the same pattern and using the KPP-generated subroutines for the function and Jacobian derivatives with respect to the rate coefficients.

6. Building adjoint code with KPP

First applications of the KPP software tools to the adjoint code generation for chemical kinetics systems were presented by Daescu et al. (2000), who reported a superior performance over the adjoint code generated with the general purpose adjoint compiler TAMC (Giering and Kaminski, 1998) for a two-stage Rosenbrock method. In this section, we present a tutorial example for the continuous and discrete adjoint code implementation in the KPP framework. For simplicity, we consider the problem of evaluating the sensitivities of a response functional $g(y(t^F))$ with respect to the model state at previous instants in time, $t^0 \leqslant t < t^F$

$$S(t) = \left(\frac{\partial g(t^F)}{\partial y_1(t)},\ldots,\frac{\partial g(t^F)}{\partial y_n(t)}\right)^T, \tag{58}$$

using the numerical scheme (57). The adjoint variables are initialized with $\lambda^N = \lambda(t^F) = \partial g(y(t^F))/\partial y$.

6.1. Continuous adjoint implementation

One step of the backward integration, $\lambda^{i+1} \to \lambda^i$, of the continuous adjoint model (26) using the linearly implicit Euler method is written (use Eq. (57) for Eq. (26) with $h \leftarrow -h$)

$$\left(J^T(t^{i+1},y^{i+1}) - \frac{1}{h}I\right)(\lambda^{i+1} - \lambda^i)$$
$$= J^T(t^{i+1},y^{i+1})\lambda^{i+1} \tag{59}$$

and after rearranging we obtain

$$\left(J(t^{i+1},y^{i+1}) - \frac{1}{h}I\right)^T \lambda^i = -\frac{1}{h}\lambda^{i+1}. \tag{60}$$

Since the adjoint equations are linear, fully implicit methods may be implemented at the same computational cost as linearly implicit methods. If the backward integration is performed with the implicit Euler method then

$$\left(J(t^i,y^i) - \frac{1}{h}I\right)^T \lambda^i = -\frac{1}{h}\lambda^{i+1} \tag{61}$$

with the only difference from Eq. (59) being that the Jacobian matrix in Eq. (61) is now evaluated at (t^i,y^i). Note that Eqs. (59) and (61) need only a sparse LU decomposition and a backsubstitution with the transposed Jacobian, both operations being implemented by KPP.

6.2. Discrete adjoint implementation

The discrete adjoint code is implemented according to formulae (43) and (49). Differentiating Eq. (57) with respect to y^i we obtain

$$H^i \times (y^i - y^{i+1}) + \left(J^i - \frac{1}{h}I\right)\left(I - \frac{\partial y^{i+1}}{\partial y^i}\right) = J^i, \tag{62}$$

where $J^i = J(t^i,y^i)$ and $H^i = \partial J^i/\partial y^i$. From Eq. (62) we obtain explicitly

$$\frac{\partial y^{i+1}}{\partial y^i} = I - \left(J^i - \frac{1}{h}I\right)^{-1}(J^i + H^i \times (y^{i+1} - y^i))$$
$$= -\left(J^i - \frac{1}{h}I\right)^{-1}\left(\frac{1}{h}I + H^i \times (y^{i+1} - y^i)\right) \tag{63}$$

(where $\times$ stands for 3-tensor times vector product). From Eqs. (49) and (63) it results

$$\lambda^i = \left(\frac{\partial y^{i+1}}{\partial y^i}\right)^{\mathrm{T}} \lambda^{i+1} = -\left(\frac{1}{h} I + H^i \times (y^{i+1} - y^i)\right)^{\mathrm{T}} \left(J^i - \frac{1}{h} I\right)^{-\mathrm{T}} \lambda^{i+1}. \tag{64}$$

Eq. (64) represents the discrete adjoint integration step associated with the linearly implicit Euler forward integration. Following Daescu et al. (2000), we introduce two new variables k and z

$$k = y^{i+1} - y^i, \quad \left(J^i - \frac{1}{h} I\right)^{\mathrm{T}} z = \lambda^{i+1}, \tag{65}$$

with which Eq. (64) becomes

$$\lambda^i = -\frac{1}{h} z - (H^i \times k)^{\mathrm{T}} z. \tag{66}$$

The linear system in Eq. (65) is solved for z efficiently through calls to the KPP-generated sparse linear algebra routines KppDecomp and KppSolve_TR. The right-hand side term in Eq. (66) is evaluated by HessVarTR_Vec taking full advantage of the Hessian sparsity.

Remark 4. By comparison of Eqs. (65) and (66) with (60) and (61) it can be seen that discrete adjoint model is a more demanding computational process and its efficient implementation is not a trivial task. For this reason, the use of discrete adjoints in atmospheric chemistry applications has been limited to explicit or low-order linearly implicit numerical methods (Fisher and Lary, 1995; Elbern et al., 1997; Daescu et al., 2000).

Remark 5. For linear dynamics $J = J(t)$ the second-order derivatives in Eq. (66) are identically zero. Therefore, the discrete adjoint model (66) is equivalent with the continuous adjoint model (61).

7. The KPP numerical library

The KPP numerical library is extended with a set of numerical integrators and drivers for direct-decoupled and for adjoint sensitivity computations.

Several Rosenbrock methods are implemented for direct-decoupled sensitivity analysis, namely Ros1, Ros2, Ros3, Rodas3, and Ros4. The implementations distinguish between sensitivities with respect to initial values and sensitivities with respect to parameters for efficiency. In addition, the BDF direct-decoupled integrator Odessa (Leis and Kramer, 1986) is available with the KPP sparse linear algebra routines.

Drivers are present for the general computation of sensitivities with respect to all initial values (general_ddm_ic) and with respect to several (user-defined) rate coefficients (general_ddm_rc). Note that KPP produces the building blocks for the simulation and also for the sensitivity calculations; it also provides application programming templates. Some minimal programming may be required from the users in order to construct their own application from the KPP building blocks.

The continuous adjoint model can be easily constructed using KPP-generated routines and is integrated with any user selected numerical method. The discrete adjoint models associated with the Ros1, Ros2, and Rodas3 integrators are provided for variable step size integration. Drivers for adjoint sensitivity and data assimilation applications are also included.

8. Conclusions and future work

The analysis of comprehensive chemical reactions mechanisms, parameter estimation techniques, and variational chemical data assimilation applications require the development of efficient sensitivity methods for chemical kinetics systems. Popular methods for chemical sensitivity analysis include the direct-decoupled method and automatic differentiation.

In this paper, we review the theory of the direct and the adjoint methods for sensitivity analysis in the context of chemical kinetic simulations. The direct method integrates the model and its sensitivity equations simultaneously and can efficiently evaluate the sensitivities of all concentrations with respect to few model parameters. An efficient numerical implementation is the direct-decoupled method, traditionally formulated using BDF formulas. We extended the direct-decoupled approach to Runge–Kutta and Rosenbrock stiff integration methods. The adjoint method integrates the adjoint of the tangent linear model backwards in time and can efficiently evaluate the sensitivities of a scalar response function with respect to a large number of model parameters. Sensitivities with respect to time-dependent model parameters may be obtained through a single backward integration of the adjoint model.

The kinetic preprocessor (KPP) developed by the authors is a symbolic engine that translates a given chemical mechanism into Fortran or C kinetic simulation code. Efficiency is obtained by carefully exploiting the sparsity structure of the Jacobian. A comprehensive suite of stiff numerical integrators is also provided.

The second part of this paper presents the comprehensive set of software tools for sensitivity analysis that were developed and implemented in the new release of the KPP-1.2. KPP was extended to symbolically generate code for the stoichiometric formulation of kinetic systems; for a direct sparse multiplication of Jacobian transposed times vector; for the solution of linear systems involving the transposed Jacobian; for the derivatives of rate function and its Jacobian with respect to reaction coefficients; for the Hessian, i.e. the derivatives of the Jacobian with respect to the concentrations; and for the sparse tensor product of Hessian with user defined vectors. The Hessian and the derivatives with respect to rate coefficients are sparse entities; KPP analyzes their sparsity and produces simulation code together with efficient sparse data structures.

The use of these software tools to build sensitivity simulation code is outlined, following the theoretical overview of direct and adjoint sensitivity analysis. Direct-decoupled code is constructed in a straightforward way; BDF and Runge–Kutta direct-decoupled integrators, as well as specific drivers are included in the KPP numerical library. The need for Jacobian derivatives prevented Rosenbrock methods to be used extensively in direct sensitivity calculations; however, the proposed symbolic differentiation technology makes the computation of these derivatives feasible. The continuous and discrete adjoint models are completely generated by the KPP software taking full advantage of the sparsity of the chemical mechanism. Flexible direct-decoupled and adjoint sensitivity code implementations are achieved, and various numerical integration methods can be employed with minimal user intervention.

In the companion paper (Daescu et al., 2003), we present an extensive set of numerical experiments and demonstrate the efficiency of the KPP software as a tool for direct/adjoint sensitivity applications. These applications include direct-decoupled and adjoint sensitivity calculations with respect to initial conditions, emissions, and reaction rate coefficients; time-dependent sensitivity analysis; variational data assimilation and parameters estimation.

Acknowledgements

The authors are grateful to NSF support for this work through the award ITR AP&IM-0205198. The work of A. Sandu was also supported in part by the NSF CAREER award ACI-0093139. D.N. Daescu acknowledges the support from the Supercomputing Institute for Digital Simulation and Advanced Computation of the University of Minnesota.

References

Bryson, A.E., Ho, Y.-C., 1975. Applied Optimal Control: Optimization, Estimation, and Control. Hemisphere Publishing Corporation, Washington, DC.

Byrne, G.D., Dean, A.M., 1993. The numerical solution of some chemical kinetics models with VODE and CHEMKIN II. Computers and Chemistry 17 (3), 297–302.

Cacuci, D.G., 1981a. Sensitivity theory for nonlinear systems. I. Nonlinear functional analysis approach. Journal of Mathematical Physics 22, 2794–2802.

Cacuci, D.G., 1981b. Sensitivity theory for nonlinear systems. II. Extensions to additional classes of responses. Journal of Mathematical Physics 22, 2803–2812.

Caracotsios, M., Stewart, W.E., 1985. Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations. Computers and Chemical Engineering 9 (4), 359–365.

Carter, W.P.L., 2000. Documentation of the SAPRC-99 chemical mechanism for VOC reactivity assessment. Final Report to California Air Resources Board Contract No. 92-329 and 95-308, May 2000.

Daescu, D.N., Carmichael, G.R., Sandu, A., 2000. Adjoint implementation of Rosenbrock methods applied to variational data assimilation problems. Journal of Computational Physics 165 (2), 496–510.

Daescu, D.N., Sandu, A., Carmichael, G.R., 2003. Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: II—Numerical validation and applications. Atmospheric Environment, in press, doi:10.1016/j.atmosenv.2003.08.020.

Damian-Iordache, V., Sandu, A., Damian-Iordache, M., Carmichael, G.R., Potra, F.A., 1995. KPP—a symbolic preprocessor for chemistry kinetics—user's guide. Technical Report, The University of Iowa, Iowa City, IA 52246.

Damian-Iordache, V., Sandu, A., Damian-Iordache, M., Carmichael, G.R., Potra, F.A., 2002. The kinetic preprocessor KPP—a software environment for solving chemical kinetics. Computers and Chemical Engineering 26 (11), 1567–1579.

Djouad, R., Sportisse, B., Audiffren, N., 2002. Numerical simulation of aqueous-phase atmospheric models: use of a non-autonomous Rosenbrock method. Atmospheric Environment 36, 873–879.

Dougherty, E.P., Hwang, J.-T., Rabitz, H., 1979. Further developments and applications of the Green's function method of sensitivity analysis in chemical kinetics. Journal of Chemical Physics 71 (4), 1794–1808.

Dunker, A.M., 1984. The decoupled direct method for calculating sensitivity coefficients in chemical kinetics. Journal of Chemical Physics 81, 2385–2393.

Elbern, H., Schmidt, H., Ebel, A., 1997. Variational data assimilation for tropospheric chemistry modeling. Journal of Geophysical Research 102 (D13), 15967–15985.

Errera, Q., Fonteyn, D., 2001. Four-dimensional variational chemical assimilation of CRISTA stratospheric measurements. Journal of Geophysical Research 106 (D11), 12253–12265.

Fisher, M., Lary, D.J., 1995. Lagrangian four-dimensional variational data assimilation of chemical species. Quarterly Journal of the Royal Meteorological Society 121, 1681–1704.

Giering, R., Kaminski, T., 1998. Recipes for adjoint code construction. ACM Transactions on Mathematical Software 24 (4), 437–474.

Hairer, E., Wanner, G., 1991. Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. Springer, Berlin.

Hwang, J.-T., Dougherty, E.P., Rabitz, S., Rabitz, H., 1978. The Green's function method of sensitivity analysis in chemical kinetics. Journal of Chemical Physics 69 (11), 5180–5191.

Jacobson, M.Z., Turco, R.P., 1994. SMVGEAR: a sparse-matrix, vectorized Gear code for atmospheric models. Atmospheric Environment 17, 273–284.

Leis, J.R., Kramer, M.A., 1986. ODESSA—an ordinary differential equation solver with explicit simultaneous sensitivity analysis. ACM Transactions on Mathematical Software 14 (1), 61–67.

Marchuk, G.I., 1995. Adjoint Equations and Analysis of Complex Systems. Kluwer Academic Publishers, Dordrecht.

Marchuk, G.I., Agoshkov, I.V., Shutyaev, P.V., 1996. Adjoint Equations and Perturbation Algorithms in Nonlinear Problems. CRC Press, Boca Raton, FL.

Menut, L., Vautard, R., Beekmann, M., Honoré, C., 2000. Sensitivity of photochemical pollution using the adjoint of a simplified chemistry-transport model. Journal of Geophysical Research—Atmospheres 105-D12 (15), 15379–15402.

Rostaing, N., Dalmas, S., Galligo, A., 1993. Automatic differentiation in Odyssée. Tellus 45, 558–568.

Sandu, A., Potra, F.A., Damian-Iordache, V., Carmichael, G.R., 1996. Efficient implementation of fully implicit methods for atmospheric chemistry. Journal of Computational Physics 129, 101–110.

Sandu, A., van Loon, M., Potra, F.A., Carmichael, G.R., Verwer, J.G., 1997a. Benchmarking stiff ODE solvers for atmospheric chemistry equations I—implicit vs. explicit. Atmospheric Environment 31, 3151–3166.

Sandu, A., Blom, J.G., Spee, E., Verwer, J.G., Potra, F.A., Carmichael, G.R., 1997b. Benchmarking stiff ODE solvers for atmospheric chemistry equations II—Rosenbrock solvers. Atmospheric Environment 31, 3459–3472.

Seefeld, S., Stockwell, W.R., 1999. First-order sensitivity analysis of models with time-dependent parameters: an application to PAN and ozone. Atmospheric Environment 33, 2941–2953.

Sei, A., Symes, W.W., 1995. A note on consistency and adjointness for numerical schemes. Technical Report 95527, Rice university, Houston, TX.

Sirkes, Z., Tziperman, E., 1997. Finite difference of adjoint or adjoint of finite difference? Monthly Weather Review 49, 5–40.

Stoer, J., Burlich, R., 1980. Introduction to Numerical Analysis. Springer, New York, NY.

Ustinov, E.A., 2001. Adjoint sensitivity analysis of atmospheric dynamics: application to the case of multiple observables. Journal of the Atmospheric Sciences 58, 3340–3348.

Vautard, R., Beekmann, M., Menut, L., 2000. Applications of adjoint modeling in atmospheric chemistry: sensitivity and inverse modeling. Environmental Modeling and Software 15, 703–709.

Verwer, J.G., Spee, E., Blom, J.G., Hunsdorfer, W., 1999. A second order Rosenbrock method applied to photochemical dispersion problems. SIAM Journal on Scientific Computing 20, 1456–1480.

Vuilleumier, L., Harley, R.A., Brown, N.J., 1997. First- and second-order sensitivity analysis of a photochemically reactive system (a Green's function approach). Environmental Science and Technology 31, 1206–1217.

Wang, K.Y., Lary, D.J., Shallcross, D.E., Hall, S.M., Pyle, J.A., 2001. A review on the use of the adjoint method in four-dimensional atmospheric-chemistry data assimilation. Quarterly Journal of the Royal Meteorological Society 127 (576, Part B), 2181–2204.