SENSITIVITY ANALYSIS IN INDEX-1 DIFFERENTIAL ALGEBRAIC EQUATIONS BY ESDIRK METHODS

Morten Rode Kristensen *** **John Bagterp Jørgensen** *,1
Per Grove Thomsen ** **Michael Locht Michelsen** ***
Sten Bay Jørgensen ***

* *2-control ApS, Høffdingsvej 34, 2500 Valby, Denmark*
** *Informatics and Mathematical Modeling, Technical University of Denmark, 2800 Lyngby, Denmark*
*** *Department of Chemical Engineering, Technical University of Denmark, 2800 Lyngby, Denmark*

Abstract: Dynamic optimization by multiple shooting requires integration and sensitivity calculation. A new semi-implicit Runge-Kutta algorithm for numerical sensitivity calculation of index-1 DAE systems is presented. The algorithm calculates sensitivities with respect to problem parameters and initial conditions, exploiting the special structure of the sensitivity equations. The algorithm is a one-step method which makes it especially efficient compared to multiple-step methods when frequent discontinuities are present. These advantages render the new algorithm particularly suitable for dynamic optimization and nonlinear model predictive control. The algorithm is tested on the Dow Chemicals benchmark problem. *Copyright ©2005 IFAC.*

Keywords: Differential algebraic equations, Sensitivity computation, Nonlinear model predictive control, Parameter estimation

1. INTRODUCTION

Sensitivity analysis for DAE systems is important in many engineering and scientific applications. It is useful in gradient based optimization of systems described by differential algebraic equations. Such applications include parameter estimation (prediction-error-estimation), dynamic optimization (optimal control) and experimental design. Furthermore, sensitivity analysis finds applications within model reduction. In this paper, we present an algorithm for solution of index-1 DAE systems and simultaneous computation of the sensitivities. The algorithm is an extension of the ESDIRK algorithm (Alexander, 2003; Kristensen *et al.*, 2004*a*; Kristensen *et al.*, 2004*b*) and is motivated by optimal control problems solved using multiple shooting (Binder *et al.*, 2001; Jørgensen *et al.*, 2004). The overall efficiency of the multiple shooting optimal control algorithm depends critically on the efficiency of the procedure for integration and sensitivity calculation as most of the computational effort is spent in this part of the program.

[1] Corresponding author. E-mail: jbj@2-control.com. Phone: +45 70 222 404

The algorithm is applicable to index-1 differential algebraic equations in semi-explicit form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{z}(t), \mathbf{u}) \tag{1a}$$
$$\mathbf{0} = \mathbf{g}(t, \mathbf{x}(t), \mathbf{z}(t), \mathbf{u}) \tag{1b}$$
$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{1c}$$

$\frac{\partial \mathbf{g}}{\partial \mathbf{z}}$ is assumed to be non-singular. The solution of this system is denoted: $\mathbf{x}(t) = \mathbf{x}(t; \mathbf{x}_0, \mathbf{u})$ and $\mathbf{z}(t) = \mathbf{z}(t; \mathbf{x}_0, \mathbf{u})$. The state sensitivities, $\mathbf{s}_{x,x_0}(t) = \frac{\partial}{\partial \mathbf{x}_0}\mathbf{x}(t; \mathbf{x}_0, \mathbf{u})$ and $\mathbf{s}_{z,x_0}(t) = \frac{\partial}{\partial \mathbf{x}_0}\mathbf{z}(t; \mathbf{x}_0, \mathbf{u})$, are given by the linear differential algebraic system

$$\dot{\mathbf{s}}_{x,x_0} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\mathbf{s}_{x,x_0} + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{z}}\right)\mathbf{s}_{z,x_0} \quad (2a)$$

$$\mathbf{0} = \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)\mathbf{s}_{x,x0} + \left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right)\mathbf{s}_{z,x0} \quad (2b)$$

$$\mathbf{s}_{x,x_0}(0) = \mathbf{I} \quad (2c)$$

Similarly, the parameter sensitivities, denoted $\mathbf{s}_{x,u}(t) = \frac{\partial}{\partial \mathbf{u}}\mathbf{x}(t;\mathbf{x}_0,\mathbf{u})$ and $\mathbf{s}_{z,u}(t) = \frac{\partial}{\partial \mathbf{u}}\mathbf{z}(t;\mathbf{x}_0,\mathbf{u})$, are given by the linear differential algebraic system

$$\dot{\mathbf{s}}_{x,u} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\mathbf{s}_{x,u} + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{z}}\right)\mathbf{s}_{z,u} + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right) \quad (3a)$$

$$\mathbf{0} = \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)\mathbf{s}_{x,u} + \left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right)\mathbf{s}_{z,u} + \left(\frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right) \quad (3b)$$

$$\mathbf{s}_{x,u}(0) = \mathbf{0} \quad (3c)$$

The method presented is an explicit singly diagonal Runge-Kutta algorithm which simultaneously solves (1)-(3) for the state and algebraic trajectories, $\mathbf{x}(t)$ and $\mathbf{z}(t)$, as well as the state and parameter sensitivities, $\mathbf{s}_{x,x_0}(t)$ and $\mathbf{s}_{x,u}(t)$. The algorithm has been extended with a predictive step-length controller, and efficiently reuses the Jacobian used in solving (1) for solution of the sensitivity equations, (2)-(3). In contrast to BDF methods, which are multi-step methods, the ESDIRK method is a one-step method which implies that it is efficient for problems with frequent discontinuities; e.g. zero-order-hold parameterized optimal control problems.

2. OPTIMAL CONTROL BY MULTIPLE SHOOTING

As a motivating application for sensitivity computations for differential algebraic equations, consider the optimal Bolza problem

$$\min_{\{\mathbf{x}(t),\mathbf{u}(t)\}} \phi = \int_{t_0}^{t_f} h(t,\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}(t))dt + L(\mathbf{x}(t_f)) \quad (4a)$$

$$s.t. \quad \dot{\mathbf{x}}(t) = \mathbf{f}(t,\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}(t)) \quad (4b)$$

$$\mathbf{0} = \mathbf{g}(t,\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}(t)) \quad (4c)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \quad (4d)$$

in which the controls are discretized using zero-order-hold

$$\mathbf{u}(t) = \mathbf{u}_k \quad t_k \le t < t_{k+1},\ k = 0,1,\ldots,N-1 \quad (5)$$

The corresponding discrete-time problem which can be solved by SQP techniques is

$$\min \phi = \sum_{k=0}^{N-1} H_k(\mathbf{x}_k,\mathbf{u}_k) + L(\mathbf{x}_N) \quad (6a)$$

$$s.t. \ \mathbf{x}_{k+1} = \mathbf{F}_k(\mathbf{x}_k,\mathbf{u}_k) \quad (6b)$$

The operator, $\mathbf{F}_k(\mathbf{x}_k,\mathbf{u}_k) = \mathbf{x}(t_{k+1})$, is computed as the solution of an index-1 DAE system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t,\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}_k) \quad (7a)$$

$$\mathbf{0} = \mathbf{g}(t,\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}_k) \quad (7b)$$

$$\mathbf{x}(t_k) = \mathbf{x}_k \quad (7c)$$

and the operator

$$H_k(\mathbf{x}_k,\mathbf{u}_k) = \int_{t_k}^{t_{k+1}} h(t,\mathbf{x}(t),\mathbf{z}(t),\mathbf{u})dt \quad (8)$$

is computed by quadrature along the solution of the index-1 DAE system. Alternatively, it may be computed by augmenting the DAE system by an extra state, i.e. $\dot{y} = h(t,\mathbf{x}(t),\mathbf{z}(t),\mathbf{u})$ and $y(t_k) = 0$. In addition to evaluation of $H_k(\mathbf{x}_k,\mathbf{u}_k)$ and $\mathbf{F}_k(\mathbf{x}_k,\mathbf{u}_k)$, the solution of (6) by SQP methods require evaluation of the gradients of $\mathbf{F}_k(\mathbf{x}_k,\mathbf{u}_k)$

$$\mathbf{A}_k = \nabla_{x_k}\mathbf{F}_k(\mathbf{x}_k,\mathbf{u}_k) \quad (9a)$$

$$\mathbf{B}_k = \nabla_{u_k}\mathbf{F}_k(\mathbf{x}_k,\mathbf{u}_k) \quad (9b)$$

as well as the gradients of $H_k(\mathbf{x}_k,\mathbf{u}_k)$

$$\mathbf{q}_k = \nabla_{x_k}H_k(\mathbf{x}_k,\mathbf{u}_k) \quad (10a)$$

$$\mathbf{r}_k = \nabla_{u_k}H_k(\mathbf{x}_k,\mathbf{u}_k) \quad (10b)$$

The gradients, $\mathbf{A}_k$ and $\mathbf{B}_k$, are computed as the state and parameter sensitivities of the index-1 DAE system (7), i.e. $\mathbf{A}_k = \mathbf{s}_{x,x_0}(t_{k+1})^T$ and $\mathbf{B}_k = \mathbf{s}_{x,u}(t_{k+1})^T$. $\mathbf{q}_k$ and $\mathbf{r}_k$ may be computed in a similar manner. This motivates the need for efficient solution of index-1 DAE systems and their sensitivities in optimal control including nonlinear model predictive control.

3. THE ESDIRK METHOD FOR DAE SYSTEMS

The algorithm presented here for joint state and sensitivity integration is based on an ESDIRK method (Kristensen *et al.*, 2004*a*; Alexander, 2003). The algorithm has been implemented in the code ESDIRK34 which treats index-1 DAE systems of the form (1). The ESDIRK integration scheme for the differential equations is

$$\mathbf{X}_i = \mathbf{x}_n + h\sum_{j=1}^{i} a_{ij}\mathbf{f}(t_n + c_j h, \mathbf{X}_j, \mathbf{Z}_j, \mathbf{u})$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\sum_{i=1}^{4} b_i\mathbf{f}(t_n + c_i h, \mathbf{X}_i, \mathbf{Z}_i, \mathbf{u}) \quad (11)$$

$$\mathbf{e}_{n+1} = h\sum_{i=1}^{4} d_i\mathbf{f}(t_n + c_i h, \mathbf{X}_i, \mathbf{Z}_i, \mathbf{u})$$

in which $\mathbf{X}_i$ denotes the solution at the ith internal stage ($i = 1,2,3,4$) of step n and $\mathbf{e}$ is the error vector. $\mathbf{Z}_j$ must satisfy the algebraic equations

$$\mathbf{0} = \mathbf{g}(t_n + c_j h, \mathbf{X}_j, \mathbf{Z}_j, \mathbf{u}), \quad j = 1,2,3,4 \quad (12)$$

The coefficients in (11) are presented in the Butcher tableau

$$\begin{array}{c|cccc} 0 & 0 & & & \\ c_2 & a_{21} & \gamma & & \\ c_3 & a_{31} & a_{32} & \gamma & \\ 1 & b_1 & b_2 & b_3 & \gamma \\ \hline \mathbf{x}_{n+1} & b_1 & b_2 & b_3 & \gamma \\ \hline \mathbf{e}_{n+1} & d_1 & d_2 & d_3 & d_4 \end{array} = \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \\ \hline & \mathbf{d}^T \end{array} \quad (13)$$

According to the tableau the solution to the first internal stage is explicitly given as the solution to the last stage of the previous step. Solution of the nonlinear equations in (11) requires an iterative procedure in which the equations (12) are solved simultaneously. Following the integration scheme the full set of equations for the ith internal stage is

$$\begin{aligned}\begin{bmatrix}\mathbf{X}_i\\ \mathbf{0}\end{bmatrix} &= \begin{bmatrix}\mathbf{x}_n\\ \mathbf{0}\end{bmatrix} + h\sum_{j=1}^{i} a_{ij}\cdot\begin{bmatrix}\mathbf{f}(T_j,\mathbf{X}_j,\mathbf{Z}_j,\mathbf{u})\\ \mathbf{g}(T_j,\mathbf{X}_j,\mathbf{Z}_j,\mathbf{u})\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{x}_n\\ \mathbf{0}\end{bmatrix} + h\sum_{j=1}^{i-1} a_{ij}\cdot\begin{bmatrix}\mathbf{f}(T_j,\mathbf{X}_j,\mathbf{Z}_j,\mathbf{u})\\ \mathbf{0}\end{bmatrix}\\ &\quad + h\gamma\cdot\begin{bmatrix}\mathbf{f}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u})\\ \mathbf{g}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u})\end{bmatrix}\end{aligned}$$

in which $T_i = t_n + c_i h$. In residual form this becomes

$$\begin{aligned}&\mathbf{R}(\mathbf{X}_i,\mathbf{Z}_i) =\\ &\begin{bmatrix}\mathbf{X}_i\\ \mathbf{0}\end{bmatrix} - \begin{bmatrix}\mathbf{x}_n\\ \mathbf{0}\end{bmatrix} - h\sum_{j=1}^{i-1} a_{ij}\cdot\begin{bmatrix}\mathbf{f}(T_j,\mathbf{X}_j,\mathbf{Z}_j,\mathbf{u})\\ \mathbf{0}\end{bmatrix}\\ &- h\gamma\cdot\begin{bmatrix}\mathbf{f}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u})\\ \mathbf{g}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u})\end{bmatrix} = 0\end{aligned}$$

The residual is equated to zero and solved via Newton iterations

$$\nabla\mathbf{R}(\mathbf{X}_i^{(k)},\mathbf{Z}_i^{(k)})^T\cdot\begin{bmatrix}\Delta\mathbf{X}_i\\ \Delta\mathbf{Z}_i\end{bmatrix} = -\mathbf{R}(\mathbf{X}_i^{(k)},\mathbf{Z}_i^{(k)})$$

$$\begin{bmatrix}\mathbf{X}_i^{(k+1)}\\ \mathbf{Z}_i^{(k+1)}\end{bmatrix} = \begin{bmatrix}\mathbf{X}_i^{(k)}\\ \mathbf{Z}_i^{(k)}\end{bmatrix} + \begin{bmatrix}\Delta\mathbf{X}_i\\ \Delta\mathbf{Z}_i\end{bmatrix}$$

in which the gradient of $\mathbf{R}$ is

$$\begin{aligned}&\nabla\mathbf{R}(\mathbf{X}_i,\mathbf{Z}_i)\\ &= \begin{bmatrix}\mathbf{I} - h\gamma\nabla_{\mathbf{x}}\mathbf{f}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u}) & -h\gamma\nabla_{\mathbf{x}}\mathbf{g}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u})\\ -h\gamma\nabla_{\mathbf{z}}\mathbf{f}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u}) & -h\gamma\nabla_{\mathbf{z}}\mathbf{g}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u})\end{bmatrix}\\ &\simeq \begin{bmatrix}\mathbf{I} - h\gamma\nabla_{\mathbf{x}}\mathbf{f}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u}) & -h\gamma\nabla_{\mathbf{x}}\mathbf{g}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u})\\ -h\gamma\nabla_{\mathbf{z}}\mathbf{f}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u}) & -h\gamma\nabla_{\mathbf{z}}\mathbf{g}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u})\end{bmatrix}\\ &= \mathbf{M}\end{aligned}$$

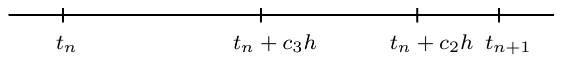
in which $\mathbf{M}$ denotes the iteration matrix. This matrix is formed from the Jacobian of the DAE system (1):

$$\mathbf{J} = \begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{f}\\ \nabla_{\mathbf{x}}\mathbf{g} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix}$$

The solutions to the second, third and fourth internal stages are calculated in the above fashion. Since the solution to stage four equals the solution at the end point, no further calculations are needed and

$$\begin{bmatrix}\mathbf{X}_4\\ \mathbf{Z}_4\end{bmatrix} = \begin{bmatrix}\mathbf{x}_{n+1}\\ \mathbf{z}_{n+1}\end{bmatrix} \tag{14}$$

Each Newton iteration requires starting guesses for all variables. Extrapolated values based on previous information is used as starting guesses for the algebraic variables. The distribution of nodes within each integration step is illustrated below:



For stage 2 the solution at t_n is used as starting guess, i.e. $\mathbf{Z}_2^{(1)} = \mathbf{z}_n$. Experiments has been made using an extrapolated value based on information from the previous integration step, but without convincing results. For stage 3 interpolation is made using $\mathbf{z}_n$ and $\mathbf{Z}_2$, and the starting guess for stage 4 is based on extrapolation from $\mathbf{Z}_2$ and $\mathbf{Z}_3$.

3.1 *Sensitivities*

ESDIRK34 uses a staggered direct approachto sensitivity analysis (Caracotsios and Stewart, 1985). In the staggered direct approach the state and sensitivity integration is performed in a staggered fashion within each integration step. First, the model states of the DAE system are integrated from t_n to t_{n+1} after which the linear sensitivity equations are solved directly using the current factorization of the iteration matrix.

3.1.1. State Sensitivities The state sensitivities are derived by differentiation of the discrete formula (11) with respect to the initial states. If we let the algebraic states follow the integration scheme, the ith stage gradient ($i = 2, 3, 4$) is calculated as follows for one integration step

$$\begin{aligned}\nabla_{\mathbf{x}_n}\begin{bmatrix}\mathbf{X}_i\\ \mathbf{0}\end{bmatrix} =& \nabla_{\mathbf{x}_n}\begin{bmatrix}\mathbf{x}_n\\ \mathbf{0}\end{bmatrix} +\\ & h\sum_{j=1}^{i} a_{ij}\nabla_{\mathbf{x}_n}\begin{bmatrix}\mathbf{f}(T_j,\mathbf{X}_j,\mathbf{Z}_j,\mathbf{u})\\ \mathbf{g}(T_j,\mathbf{X}_j,\mathbf{Z}_j,\mathbf{u})\end{bmatrix}\end{aligned}$$

which yields

$$\begin{aligned}&\begin{bmatrix}\nabla_{\mathbf{x}_n}\mathbf{X}_i & \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\\ &\quad + ha_{i1}\begin{bmatrix}\nabla_{\mathbf{x}_n}\mathbf{x}_n & \nabla_{\mathbf{x}_n}\mathbf{z}_n\end{bmatrix}\cdot\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix}\\ &\quad + h\sum_{j=2}^{i-1} a_{ij}\begin{bmatrix}\nabla_{\mathbf{x}_n}\mathbf{X}_j & \nabla_{\mathbf{x}_n}\mathbf{Z}_j\end{bmatrix}\cdot\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix}\\ &\quad + h\gamma\begin{bmatrix}\nabla_{\mathbf{x}_n}\mathbf{X}_i & \nabla_{\mathbf{x}_n}\mathbf{Z}_i\end{bmatrix}\cdot\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix}\end{aligned}$$

The last term is moved to the left hand side, and the result is

$$\begin{aligned}&\begin{bmatrix}\nabla_{\mathbf{x}_n}\mathbf{X}_i & \nabla_{\mathbf{x}_n}\mathbf{Z}_i\end{bmatrix}\cdot\overbrace{\left[\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} - h\gamma\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix}\right]}^{\simeq\mathbf{M}}\\ &= \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix} + ha_{i1}\begin{bmatrix}\mathbf{I} & \nabla_{\mathbf{x}_n}\mathbf{z}_n\end{bmatrix}\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix}\\ &\quad + h\sum_{j=2}^{i-1} a_{ij}\begin{bmatrix}\nabla_{\mathbf{x}_n}\mathbf{X}_j & \nabla_{\mathbf{x}_n}\mathbf{Z}_j\end{bmatrix}\cdot\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix}\end{aligned}$$

The derivatives of $\mathbf{f}$ and $\mathbf{g}$ with respect to $\mathbf{X}_i$ and $\mathbf{Z}_i$ are approximated by the derivatives at t_n, i.e.:

$$\begin{aligned}&\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u}) & \nabla_{\mathbf{x}}\mathbf{g}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u})\\ \nabla_{\mathbf{z}}\mathbf{f}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u}) & \nabla_{\mathbf{z}}\mathbf{g}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u})\end{bmatrix}\\ &\quad\simeq \begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u}) & \nabla_{\mathbf{x}}\mathbf{g}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u})\\ \nabla_{\mathbf{z}}\mathbf{f}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u}) & \nabla_{\mathbf{z}}\mathbf{g}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u})\end{bmatrix}\end{aligned}$$

for $i = 2, 3, 4$. Since $\mathbf{X}_4 = \mathbf{x}_{n+1}$ and $\mathbf{Z}_4 = \mathbf{z}_{n+1}$, the stage gradients $\nabla_{\mathbf{x}_n}\mathbf{X}_4$ and $\nabla_{\mathbf{x}_n}\mathbf{Z}_4$ equals the desired state sensitivities.

3.1.2. Parameter Sensitivities The parameter sensitivities are derived in a similar fashion. By differentiation of (11) with respect to the parameter vector $\mathbf{u}$ the ith stage gradient is

$$\nabla_{\mathbf{u}}\begin{bmatrix}\mathbf{X}_i\\ \mathbf{0}\end{bmatrix} = \nabla_{\mathbf{u}}\begin{bmatrix}\mathbf{x}_n\\ \mathbf{0}\end{bmatrix} + h\sum_{j=1}^{i} a_{ij}\nabla_{\mathbf{u}}\begin{bmatrix}\mathbf{f}(T_j,\mathbf{X}_j,\mathbf{Z}_j,\mathbf{u})\\ \mathbf{g}(T_j,\mathbf{X}_j,\mathbf{Z}_j,\mathbf{u})\end{bmatrix}$$

which may be expressed as

$$\begin{aligned}&\begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{X}_i & \mathbf{0}\end{bmatrix} = \\ &ha_{i1}\left[\begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{x}_n & \nabla_{\mathbf{u}}\mathbf{z}_n\end{bmatrix}\cdot\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix} + \begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{f} & \nabla_{\mathbf{u}}\mathbf{g}\end{bmatrix}\right]\\ &+ h\sum_{j=2}^{i-1} a_{ij}\left[\begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{X}_j & \nabla_{\mathbf{u}}\mathbf{Z}_j\end{bmatrix}\cdot\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix} + \begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{f} & \nabla_{\mathbf{u}}\mathbf{g}\end{bmatrix}\right]\\ &+ h\gamma\left[\begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{X}_i & \nabla_{\mathbf{u}}\mathbf{Z}_i\end{bmatrix}\cdot\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix} + \begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{f} & \nabla_{\mathbf{u}}\mathbf{g}\end{bmatrix}\right]\end{aligned}$$

The last term containing $\nabla_{\mathbf{u}}\mathbf{X}_i$ and $\nabla_{\mathbf{u}}\mathbf{Z}_i$ is moved to the left hand side, and the result is

$$\begin{aligned}&\begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{X}_i & \nabla_{\mathbf{u}}\mathbf{Z}_i\end{bmatrix}\cdot\overbrace{\left[\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} - h\gamma\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix}\right]}^{\simeq\mathbf{M}} = \\ &ha_{i1}\left[\begin{bmatrix}\mathbf{0} & \nabla_{\mathbf{u}}\mathbf{z}_n\end{bmatrix}\cdot\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix} + \begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{f} & \nabla_{\mathbf{u}}\mathbf{g}\end{bmatrix}\right] + \\ &h\sum_{j=2}^{i-1} a_{ij}\left[\begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{X}_j & \nabla_{\mathbf{u}}\mathbf{Z}_j\end{bmatrix}\cdot\begin{bmatrix}\nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g}\\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g}\end{bmatrix} + \right.\\ &\left.\begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{f} & \nabla_{\mathbf{u}}\mathbf{g}\end{bmatrix}\right] + h\gamma\begin{bmatrix}\nabla_{\mathbf{u}}\mathbf{f} & \nabla_{\mathbf{u}}\mathbf{g}\end{bmatrix}\end{aligned}$$

The stage gradients $\nabla_{\mathbf{u}}\mathbf{X}_4$ and $\nabla_{\mathbf{u}}\mathbf{Z}_4$ equal the desired parameter sensitivities. Again, the derivatives of $\mathbf{f}$ and $\mathbf{g}$ with respect to $\mathbf{X}_i$ and $\mathbf{Z}_i$ are approximated by the derivatives at t_n, and a similar approximation is made for the derivatives with respect to $\mathbf{u}$:

$$\begin{matrix}\nabla_{\mathbf{u}}\mathbf{f}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u}) \simeq \nabla_{\mathbf{u}}\mathbf{f}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u})\\ \nabla_{\mathbf{u}}\mathbf{g}(T_i,\mathbf{X}_i,\mathbf{Z}_i,\mathbf{u}) \simeq \nabla_{\mathbf{u}}\mathbf{g}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u})\end{matrix}, \quad i = 2,3,4$$

Using these approximations, only one evaluation of the partial derivatives is needed in each integration step. The derivations above apply to each individual integration step $[t_n, t_n + h]$. The sensitivities are propagated over successive integration steps using a simple recursion. If $\mathbf{A}_k = \nabla_{\mathbf{x}_k}\mathbf{x}_{k+1}$ and $\mathbf{B}_k = \nabla_{\mathbf{u}_k}\mathbf{x}_{k+1}$ denote the state and parameter sensitivity matrices for the interval $[t_k, t_{k+1}]$, corresponding to a predefined sampling interval, then the sensitivities are updated recursively as

$$\mathbf{A}_k \leftarrow \mathbf{A}_k\mathbf{A} \qquad \mathbf{B}_k \leftarrow \mathbf{B}_k\mathbf{A} + \mathbf{B} \tag{15}$$

with $\mathbf{A}_k = \mathbf{I}$ and $\mathbf{B}_k = \mathbf{0}$ at $t = t_k$, assuming that the initial conditions are independent of the parameters.

Algorithm 1 ESDIRK34 (DAE) with Sensitivities

```
$t = t_k$, $\mathbf{A}_k = \mathbf{I}$, and $\mathbf{B}_k = 0$.
while $t \le t_k + T_s$ do
  If $t + h > t_k + T_s$ then $h = t_k + T_s - t$ end if.
  Compute the Jacobian and the partial derivatives with respect to parameters
  Compute iteration matrix
    $\mathbf{M} = \begin{bmatrix}\mathbf{I} - h\gamma\nabla_{\mathbf{x}}\mathbf{f}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u}) & -h\gamma\nabla_{\mathbf{x}}\mathbf{g}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u})\\ -h\gamma\nabla_{\mathbf{z}}\mathbf{f}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u}) & -h\gamma\nabla_{\mathbf{z}}\mathbf{g}(t_n,\mathbf{x}_n,\mathbf{z}_n,\mathbf{u})\end{bmatrix}$   (16)
  and $LU$-factorize $\mathbf{M}$.
  Compute the internal stages iteratively for $i = 2, 3, 4$ using the $LU$-factorization of $\mathbf{M}$.
  while $tol \le \|\mathbf{R}\|$ do
    Compute the residual vector $\mathbf{R}$ and solve for $\begin{bmatrix}\Delta\mathbf{X}_i\\ \Delta\mathbf{Z}_i\end{bmatrix}$
      $\mathbf{R}(\mathbf{X}_i,\mathbf{Z}_i) = \begin{bmatrix}\mathbf{X}_i\\ \mathbf{0}\end{bmatrix} - \begin{bmatrix}\mathbf{x}_n\\ \mathbf{0}\end{bmatrix} - h\sum_{j=1}^{i} a_{ij}\cdot\begin{bmatrix}\mathbf{f}(T_j,\mathbf{X}_j,\mathbf{Z}_j,\mathbf{u})\\ \mathbf{g}(T_j,\mathbf{X}_j,\mathbf{Z}_j,\mathbf{u})\end{bmatrix}$   (17)
      $\mathbf{M}^T\cdot\begin{bmatrix}\Delta\mathbf{X}_i\\ \Delta\mathbf{Z}_i\end{bmatrix} = -\mathbf{R}(\mathbf{X}_i^{(k)},\mathbf{Z}_i^{(k)})$   (18)
      $\begin{bmatrix}\mathbf{X}_i^{(k+1)}\\ \mathbf{Z}_i^{(k+1)}\end{bmatrix} = \begin{bmatrix}\mathbf{X}_i^{(k)}\\ \mathbf{Z}_i^{(k)}\end{bmatrix} + \begin{bmatrix}\Delta\mathbf{X}_i\\ \Delta\mathbf{Z}_i\end{bmatrix}$   (19)
  end while
  Compute the error estimate $\mathbf{e}_{n+1}$ and tolerance monitor $r$:
    $\mathbf{e}_{n+1} = \sum_{j=1}^{4} hd_j\mathbf{f}_j$   (20)
    $r = \sqrt{\frac{1}{n_d}\sum_{i=1}^{n_d}\left(\frac{(e_{n+1})_i}{atol_i + |(x_n)_i|\, rtol_i}\right)^2}$   (21)
  if $r \le 1$ then
    Accept the step, update the time $t \leftarrow t + h$, and update the solution: $\mathbf{x}_{n+1} = \mathbf{X}_4$, $\mathbf{z}_{n+1} = \mathbf{Z}_4$.
    Compute the sensitivities $(\mathbf{A},\mathbf{B})$ for the interval $[t, t+h]$ using Algorithm 2. Update the sensitivities $(\mathbf{A}_k,\mathbf{B}_k)$: $\mathbf{A}_k \leftarrow \mathbf{A}_k\mathbf{A}$, $\mathbf{B}_k \leftarrow \mathbf{B}_k\mathbf{A} + \mathbf{B}$.
    Compute a new step size $h$ using the error controller.
  else
    Compute a new step size $h$ using the error controller.
  end if
end while
```

4. IMPLEMENTATION ISSUES

The method described has been implemented in the Fortran code ESDIRK34. The overall structure of the code is outlined in Algorithm 1. ESDIRK34 uses a staggered direct approach to sensitivity analysis. Once the convergence criterion in the Newton iteration is satisfied, the sensitivities are calculated directly according to Algorithm 2 exploiting the linearity of the sensitivity equations. This approach has several desirable properties:

Algorithm 2 Approximate Sensitivities

```
Solve for $\mathbf{V}$
    $$\mathbf{MV} = \begin{bmatrix} \nabla_{\mathbf{x}}\mathbf{f} & \nabla_{\mathbf{x}}\mathbf{g} \\ \nabla_{\mathbf{z}}\mathbf{f} & \nabla_{\mathbf{z}}\mathbf{g} \end{bmatrix} \quad (22)$$
using the $LU$ factorization of $\mathbf{M}$.
Compute
    $\mathbf{G}_1 = \left[\nabla_{\mathbf{x}}\mathbf{f} - \nabla_{\mathbf{x}}\mathbf{g}\left(\nabla_{\mathbf{z}}\mathbf{g}\right)^{-1}\nabla_{\mathbf{z}}\mathbf{f} \;\; 0\right]$   (23a)
    $\mathbf{G}_2 = \left(\left[\mathbf{I} \;\; \mathbf{0}\right] + ha_{21}\mathbf{G}_1\right)\mathbf{V}$   (23b)
    $\mathbf{G}_3 = \left(\left[\mathbf{I} \;\; \mathbf{0}\right] + ha_{31}\mathbf{G}_1 + ha_{32}\mathbf{G}_2\right)\mathbf{V}$   (23c)
    $\mathbf{K} = \left[\mathbf{I} \;\; \mathbf{0}\right] + ha_{41}\mathbf{G}_1 + ha_{42}\mathbf{G}_2 + ha_{43}\mathbf{G}_3$   (23d)
Solve for $\left[\nabla_{\mathbf{x}_n}\mathbf{X}_4 \;\; \nabla_{\mathbf{x}_n}\mathbf{Z}_4\right]$
    $\left[\nabla_{\mathbf{x}_n}\mathbf{X}_4 \;\; \nabla_{\mathbf{x}_n}\mathbf{Z}_4\right] \cdot \mathbf{M} = \mathbf{K}$   (23e)
using the $LU$ factorization of $\mathbf{M}$
Set
    $\mathbf{A} = \nabla_{\mathbf{x}_n}\mathbf{X}_4$   (23f)
Compute
    $\mathbf{L}_1 = \left[\nabla_{\mathbf{u}}\mathbf{f} - \nabla_{\mathbf{u}}\mathbf{g}\left(\nabla_{\mathbf{z}}\mathbf{g}\right)^{-1}\nabla_{\mathbf{z}}\mathbf{f} \;\; 0\right]$   (24a)
    $\mathbf{L}_2 = \left[ha_{21}\mathbf{L}_1 + h\gamma\left[\nabla_{\mathbf{u}}\mathbf{f} \;\; \nabla_{\mathbf{u}}\mathbf{g}\right]\right]\mathbf{V} + \left[\nabla_{\mathbf{u}}\mathbf{f} \;\; \nabla_{\mathbf{u}}\mathbf{g}\right]$   (24b)
    $\mathbf{L}_3 = \left[ha_{31}\mathbf{L}_1 + ha_{32}\mathbf{L}_2 + h\gamma\left[\nabla_{\mathbf{u}}\mathbf{f} \;\; \nabla_{\mathbf{u}}\mathbf{g}\right]\right]\mathbf{V} + \left[\nabla_{\mathbf{u}}\mathbf{f} \;\; \nabla_{\mathbf{u}}\mathbf{g}\right]$   (24c)
    $\mathbf{H} = ha_{41}\mathbf{L}_1 + ha_{42}\mathbf{L}_2 + ha_{43}\mathbf{L}_3 + h\gamma\left[\nabla_{\mathbf{u}}\mathbf{f} \;\; \nabla_{\mathbf{u}}\mathbf{g}\right]$   (24d)
Solve for $\left[\nabla_{\mathbf{u}}\mathbf{X}_4 \;\; \nabla_{\mathbf{u}}\mathbf{Z}_4\right]$
    $\left[\nabla_{\mathbf{u}}\mathbf{X}_4 \;\; \nabla_{\mathbf{u}}\mathbf{Z}_4\right] \cdot \mathbf{M} = \mathbf{H}$   (24e)
using the $LU$ factorization of $\mathbf{M}$
Set
    $\mathbf{B} = \nabla_{\mathbf{u}}\mathbf{X}_4$   (24f)
```

- Sensitivities are not calculated in steps, where the error control fails, thereby avoiding wasted work.
- The same Jacobian evaluation is used for the iteration matrix and for the sensitivity algorithm.
- The same LU factorization of the iteration matrix is used in the Newton iteration and in the direct solution of the sensitivities.

From Algorithm 2 it is seen that in each step, the extra effort of calculating sensitivities is effectively reduced to evaluating $\left[\nabla_{\mathbf{u}}\mathbf{f} \;\; \nabla_{\mathbf{u}}\mathbf{g}\right]$, performing three back substitutions using the already factorized iteration matrix, and carrying out a number of matrix-matrix operations. As pointed out by Li *et al.* (2000), a drawback of the staggered direct method is the need for frequent factorizations of the iteration matrix. Because the iteration matrix is used not only to converge the nonlinear system of state equations but also directly in the calculation of sensitivities, ESDIRK34 factorizes the matrix in each step. Most standard DAE solvers (which are not prepared for sensitivity analysis) use an adaptive strategy for choosing when to refactorize. There is always a trade-off between the rate of convergence in the Newton iterations and the frequency of Jacobian updates and factorizations. Since in ESDIRK34 a factorization is performed in each step, good convergence is attained, and no restrictions need to be placed on the step size changes in terms of variations from step to step, noting that a change in step size calls for a refactorization of the iteration matrix. This property simplifies the construction of the integration error and convergence controller compared to a high-performance implementation without sensitivity capabilities.

In terms of error norms, convergence criteria, stage value predictors for the Newton iteration etc., ESDIRK34 uses standard techniques; the error-norm is a mixed relative-absolute norm (21), for reasons of robustness the convergence is based on the residuals rather than the displacements (Houbak *et al.*, 1985), and the stage values for the 3rd and 4th internal integration stages are efficiently predicted using information from the second step (remembering the distribution of the quadrature nodes).

The integration error is controlled using a predictive controller for step size selection as suggested by Gustafsson (1992). The local truncation error $\boldsymbol{e}_{n+1}$ computed by the error estimator is related to the step size h_n by

$$\boldsymbol{e}_{n+1} = \boldsymbol{\phi}(t_n)h_n^{p+1} + \mathcal{O}\left(h_n^{p+2}\right) \tag{25}$$

in which $p = 3$ is the order of ESDIRK34. Asymptotically, the norm r_{n+1} of the error $\boldsymbol{e}_{n+1}$ is given by

$$r_{n+1} = \varphi_n h_n^{p+1} \tag{26}$$

in which $\varphi_n = ||\boldsymbol{\phi}(t_n) + \mathcal{O}\left(h_n\right)||$. In the conventional asymptotic error controller, the disturbance, φ_n, is assumed to be constant (or slowly varying). A dead-beat disturbance estimate may be obtained by $\hat{\varphi}_n = \varphi_{n-1} = r_n/h_{n-1}^{p+1}$. Hence, the solution accuracy $\varepsilon = r_{n+1}$ is achieved by selecting the step size by the formula

$$h_n = \left(\frac{\varepsilon}{\hat{\varphi}_n}\right)^{1/(p+1)} = \left(\frac{\varepsilon}{r_n}\right)^{1/(p+1)} h_{n-1} \tag{27}$$

Based on empirical evidence, Gustafsson (1992) noted that a better model for the disturbance φ_n is a logarithmic linear model, i.e.

$$\Delta \log \varphi_{n+1} = \Delta \log \varphi_n \tag{28}$$

in which $\Delta \log \varphi_n = \log \varphi_n - \log \varphi_{n-1}$. Using the disturbance model (28), the process model (26), and the measurement $\varphi_{n-1} = r_n/h_{n-1}^{p+1}$, Gustafsson (1992) constructed an observer for the model (28). In conjunction with a dead-beat predictive controller selecting the step size, this observer results in the following control law for computation of the step size

$$h_n = \frac{h_{n-1}}{h_{n-2}}\left(\frac{\varepsilon}{r_n}\right)^{k_2/(p+1)}\left(\frac{r_{n-1}}{r_n}\right)^{k_1/(p+1)} h_{n-1} \tag{29}$$

k_1 and k_2 are the gain parameters of the observer, while ε is the desired tolerance (including a safety factor). Gustafsson (1992) suggests $k_1 = k_2 = 1$ which corresponds to a dead-beat observer. This predictive

Table 1. Performance comparison between ESDIRK34 and sLIMEX.

Code	sLIMEX	ESDIRK34
NSTEP	28	77
NFUN	360	868
NJAC	28	75
NJACT	388	75
NLU	166	77
NBACK	4482	862
NSENS	360	77
CPU Time (sec)	0.07	0.09

integration error controller, along with a number of extensions and safety nets has been implemented in ESDIRK34. The benefits of model based step size controllers for integrators in optimization applications have recently been confirmed by Meeker *et al.* (2004).

5. METHOD COMPARISON

The ESDIRK34 algorithm was tested on the Dow Chemicals benchmark problem (Caracotsios and Stewart, 1985; Li *et al.*, 2000) consisting of 6 differential equations and 4 algebraic equations. Sensitivities were calculated for 9 kinetic rate constants. It was verified that correct results were obtained.

A performance comparison was made between ESDIRK34 and the code sLIMEX (Schlegel *et al.*, 2004) which is based on a one-step extrapolation method. sLIMEX uses a simultaneous corrector method for sensitivity computation. The results are summarized in Table 1 showing comparable performance in terms of CPU time with more function evaluations required by ESDIRK34 but significantly less Jacobian evaluations due to the efficient reuse of Jacobian information for the sensitivity integration.

6. CONCLUSION

A new algorithm for sensitivity analysis of index-1 DAEs has been presented. The algorithm is based on an ESDIRK method of the Runge-Kutta family, and it has been implemented in the code ESDIRK34. The algorithm has order 3 and is A- and L-stable as well as stiffly accurate. The state and sensitivity integrations are performed separately, thereby enabling the linearity of the sensitivity equations to be exploited. A key feature of the new algorithm is the reuse of the Jacobian evaluations for both iteration matrices and sensitivity residuals. In this way the extra effort of calculating sensitivities is minimized. Case studies have shown that clear advantages exist for the use of one step methods such as ESDIRK34, when frequent discontinuities are present in the solution. This makes ESDIRK34 suitable for use in dynamic optimization and nonlinear model predictive control. In addition ESDIRK has been applied for parameter estimation in dynamic models (Kristensen, 2004).

REFERENCES

Alexander, R. (2003). Design and implementation of DIRK integrators for stiff systems. *Applied Numerical Mathematics* **46**, 1–17.

Binder, T., L. Blank, H.G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J.P. Schlöder and O. von Stryk (2001). Introduction to model based optimization of chemical processes on moving horizons. In: *Online Optimization of Large Scale Systems* (M. Grötschel, S.O. Krumke and J. Rambau, Eds.). pp. 295–339. Springer. Berlin.

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

Gustafsson, K. (1992). Control of Error and Convergence in ODE Solvers. PhD thesis. Department of Automatic Control, Lund Institute of Technology.

Houbak, N., S. P. Nørsett and P. G. Thomsen (1985). Displacement or residual test in the application of implicit methods for stiff problems. *IMA Journal of Numerical Analysis* **5**(3), 297–305.

Jørgensen, J. B., J. B. Rawlings and S. B. Jørgensen (2004). Numerical solution of uncontrained nonlinear optimal control problems. Manuscript for Computers and Chemical Engineering.

Kristensen, M. (2004). Parameter estimation in nonlinear dynamical systems. Master's thesis. Department of Chemical Engineering, Technical University of Denmark.

Kristensen, M. R., J. B. Jørgensen, P. G. Thomsen and S. B. Jørgensen (2004*a*). An ESDIRK method with sensitivity analysis capabilities. *Computers and Chemical Engineering* **28**, 2695–2707.

Kristensen, M.R., J.B. Jørgensen, P.G. Thomsen and S.B. Jørgensen (2004*b*). Efficient sensitivity computation for nonlinear model predictive control. In: *NOLCOS 2004, 6th IFAC Symposium on Nonlinear Control Systems* (Frank Allgöwer, Ed.). IFAC. Stuttgart, Germany. pp. 723–728.

Li, S., L. Petzold and W. Zhu (2000). Sensitivity analysis of differential-algebraic equations: A comparison of methods on a special problem. *Applied Numerical Mathematics* **32**, 161–174.

Meeker, K., C. Homescu, L. Petzold, H. El-Samad and M. Khammash (2004). Digital filter stepsize control in DASPK and its effect on control optimization performance. Technical report. Department of Computer Science, University of California, Santa Barbara, USA.

Schlegel, M., W. Marquardt, R. Ehrig and U. Nowak (2004). Sensitivity analysis of linearly-implicit differential-algebraic systems by one-step extrapolation. *Applied Numerical Mathematics* **48**(1), 83–102.