Sensitivity Analysis and Design Optimization of Differential-Algebraic Equation Systems*

Linda PETZOLD, Radu SERBAN, Shengtai LI, Soumyendu RAHA and Yang CAO
Department of Mechanical and Environmental Engineering
University of California, Santa Barbara, CA 93106, USA

Abstract. We report on our progress in developing algorithms and software for sensitivity analysis and design optimization of differential-algebraic equation systems.

1. Introduction

Differential-algebraic equations (DAEs) arise in a wide variety of engineering and scientific problems, including the modeling of multibody and flexible systems. Much work has been devoted to understanding these systems and developing numerical methods and software for the simulation problem[4], although some substantial technical challenges remain.

In this paper we report on our progress in developing algorithms and software for sensitivity analysis and optimization of DAE systems. These computations are an order of magnitude more complex than simulation, require highly efficient and robust methods for simulation, and are of critical importance throughout engineering design. In section 2, we outline algorithms and software for sensitivity analysis of large-scale DAE systems, via both the forward and reverse (adjoint) modes. The new software can handle DAE systems of index up to two (in Hessenberg form), and solves for consistent initial conditions for both the state and sensitivity systems. In section 3, we formulate the general problem of design optimization and optimal control for DAE systems, and describe our algorithm based on a modified multiple shooting method for solving these problems. An important issue for this type of method is the handling of constraints. In particular, given an equality constraint, it can be included in the dynamic optimization problem either as part of the DAE, or be handled directly by the optimizer. This choice can affect both the index and stability of the resulting DAE system. In section 4, we describe our tool for analyzing the DAE structure and finding a stable partitioning of the constraints.

*This research was supported by NSF Multidisciplinary Challenge Grant CCR-98-9896198, NSF/DARPA Virtual Integrated Processing program, DOE DE-FG03-98ER25354 and by LLNL ISCR 00-15.

2. Sensitivity Analysis

Sensitivity analysis for DAE systems is important in many engineering and scientific applications. The information contained in the sensitivity trajectories is useful for parameter estimation, design optimization, optimal control, model reduction and experimental design. Here we present algorithms and software for sensitivity analysis of large-scale DAE systems of index up to two. There are two basic types of sensitivity approaches. The first, which we call the *forward mode* (sometimes referred to as the Tangent Linear Model) is very efficient for computing sensitivities of a potentially large number of output variables with respect to relatively few input variables. The second, which we call the *reverse mode* or *adjoint sensitivity analysis*, is advantageous when it is required to find the sensitivity of a single or low-dimensional output variable with respect to a large number of input variables. Thus the approaches are complementary. We have recently developed algorithms and software for both, and will outline them here.

2.1. Forward Mode

To illustrate the basic approach for sensitivity analysis, consider the general DAE system with parameters,

$$F(t, x, x', p) = 0, \quad x(0) = x_0(p) \tag{1}$$

where $x \in R^{n_x}, p \in R^{n_p}$. Here n_x is the number of time-dependent variables x as well as the dimension of the DAE system, and n_p is the number of parameters in the original DAE system. Sensitivity analysis entails finding the derivative of the solution x with respect to each parameter. This produces an additional $n_s = n_p \cdot n_y$ sensitivity equations which, together with the original system, yield

$$F(t, x, x', p) = 0, \tag{2a}$$

$$\frac{\partial F}{\partial x} s_i + \frac{\partial F}{\partial x'} s_i' + \frac{\partial F}{\partial p} = 0, \quad i = 1, ..., n_p, \tag{2b}$$

where $s_i = dx/dp_i$. Defining

$$X = \begin{bmatrix} x \\ s_1 \\ \vdots \\ s_{n_p} \end{bmatrix}, \quad \mathrm{F} = \begin{bmatrix} F(t, x, x', p) \\ \frac{\partial F}{\partial x} s_1 + \frac{\partial F}{\partial x'} s_1' + \frac{\partial F}{\partial p_1} \\ \vdots \\ \frac{\partial F}{\partial x} s_{n_p} + \frac{\partial F}{\partial x'} s_{n_p}' + \frac{\partial F}{\partial p_{n_p}} \end{bmatrix}$$

the combined system can be rewritten as

$$\mathrm{F}(t, X, X', p) = 0, \quad X(0) = \begin{bmatrix} x_0 \\ \frac{dx_0}{dp_1} \\ \vdots \\ \frac{dx_0}{dp_{n_p}} \end{bmatrix}.$$

This system can be solved by the k-th order BDF formula with step size h_{n+1} to yield a nonlinear system

$$G(X_{n+1}) = \mathrm{F}\left(t_{n+1}, X_{n+1}, X'^{(0)}_{n+1} - \frac{\alpha_s}{h_{n+1}}(X_{n+1} - X^{(0)}_{n+1}), p\right) = 0, \tag{3}$$

where $X_{n+1}^{(0)}$ and $X'^{(0)}_{n+1}$ are predicted values for X_{n+1} and X'_{n+1}, which are obtained via polynomial extrapolation of past values [4]. Also, α_s is the fixed leading coefficient which is defined in [4]. Newton's method for the nonlinear system produces the iteration

$$X_{n+1}^{(k+1)} = X_{n+1}^{(k)} - \mathbf{J}^{-1} G(X_{n+1}^{(k)}),$$

where

$$\mathbf{J} = \begin{bmatrix} J & & & & \\ J_1 & J & & & \\ J_2 & 0 & J & & \\ \vdots & \vdots & \vdots & \ddots & \\ J_{n_p} & 0 & \cdots & 0 & J \end{bmatrix} \tag{4}$$

and

$$J = \alpha \frac{\partial F}{\partial x'} + \frac{\partial F}{\partial x}, \quad J_i = \frac{\partial J}{\partial x} s_i + \frac{\partial J}{\partial p_i}$$

and $\alpha = \alpha_s / h_{n+1}$.

There are three well-established methods to solve the nonlinear system (3):

- Staggered direct method, described in [8].
- Simultaneous corrector method, described in [24].
- Staggered corrector method, described in [10].

Analysis and comparison of the performance of these three methods have been given in [10, 20], where it was found that the relative efficiencies of the methods depend on the problem and on the number of parameters. Here we describe briefly the three methods.

The staggered direct method first solves equation (3) for the state variables. After the Newton iteration for the state variables has converged, the sensitivity equations in (3) are updated with the most recent values of the state variables. Because equation (2b) is linear with a matrix J for the sensitivity equations, it is solved directly without Newton iteration. However, to solve the linear system in this way requires computation and factorization of the Jacobian matrix at each step and also extra storage for the matrix $\partial F / \partial x'$. Since the Jacobian is updated and factorized only when necessary in DAE solvers such as DASSL and DASPK [4], the additional matrix updates and factorizations may make the staggered direct method unattractive compared to the other methods. However, if the cost of a function evaluation is more than the cost of factorization of the Jacobian matrix and the number of sensitivity parameters is very large (see [20]), the staggered direct method is more efficient. We have modified the implementation of [8] to make the staggered direct method more reliable for ill-conditioned problems.

The simultaneous corrector method solves (3) as one whole nonlinear system, where Newton iteration is used. The Jacobian matrix $\mathbf{J}$ in (4) is approximated by its block diagonal in the Newton iteration. Thus, this method allows the factored corrector matrix to be reused for multiple steps. It has been shown in [24] that the resulting iteration is two-step quadratically convergent for full Newton, and convergent for modified Newton iteration.

The staggered corrector method lies in between the staggered direct method and the simultaneous corrector method. Instead of solving the linear sensitivity system directly as in the staggered direct method, a Newton iteration is used

$$s_i^{(k+1)} = s_i^{(k)} - J^{-1} G_{s_i}(s_i^{(k)}), \tag{5}$$

where G_{s_i} is the residual for the i-th sensitivity and J is the factored Jacobian matrix which is used in the Newton iteration for the state variables. Like the simultaneous corrector method, this method does not require the factorization of the Jacobian matrix at each step. One of the advantages of the staggered corrector method is that we do not need to evaluate the sensitivity equations during the iteration of solving for the state variables. This can reduce the computation time if the state variables require more iterations than the sensitivity variables. After solving for the state variables in the corrector iteration, only the diagonal part of **J** in (4) is left. We can expect that the convergence of the Newton iteration will be improved over that of using an approximate iteration matrix in the simultaneous corrector method. This has been observed in our numerical experiments.

Several approaches have been developed to calculate the sensitivity residuals that may be used with either the staggered corrector or the simultaneous corrector methods. Maly and Petzold [24] used a directional derivative finite difference approximation. For example, the ith sensitivity equation may be approximated as

$$\frac{F(t, x + \delta_i s_i, x' + \delta_i s_i', p + \delta_i e_i) - F(t, x, x', p)}{\delta_i} = 0, \tag{6}$$

where δ_i is a small scalar quantity, and e_i is the ith unit vector. Proper selection of the scalar δ_i is crucial to maintaining acceptable round-off and truncation error levels [24]. If $F(t, x, x', p)$ is already available from the state equations, which is the case in the Newton iteration of DASPK, (6) needs only one function evaluation for each sensitivity. The main drawback of this approach is that it may be inaccurate for badly scaled problems.

The selection of the increment δ_i for equation (6) in our current software is an improvement over the algorithms of [24] which was suggested by Hindmarsh [15]. The increment is given by

$$\delta_i = \Delta \max(|p_i|, 1/||u_i||_2) \tag{7}$$

where Δ is a scale factor,

$$u_i = \left(WT^{in_x + j} / WT^j : j = 1, ..., n_x \right),$$

and WT is a vector of weights determined by the relative and absolute user error tolerances and the solution x,

$$WT^j = \mathrm{RTOL}_j \cdot |x_j| + \mathrm{ATOL}_j.$$

Alternatively, the sensitivity residuals can be evaluated analytically by an automatic differentiation tool such as ADIFOR [3] or other automatic differentiation (AD) methods. We recommend using AD to evaluate the sensitivity equations. Even for some well-scaled problems, the ADIFOR-generated routine has better performance in terms of efficiency and accuracy than the finite difference approximation.

We have recently developed new software DASPK3.0 [17] for solution and forward-mode sensitivity analysis of DAE systems based on the above methods. The software makes use of the basic methods of DASSL [4] for time integration, and includes as an option the preconditioned iterative methods of DASPK [5] which are needed for solving large-scale DAE systems. DASPK3.0 provides for consistent initialization of the solutions and the sensitivities, interfaces seamlessly with automatic differentiation for the accurate evaluation of the sensitivity equations, and is capable via MPI[9] of exploiting the natural parallelism of sensitivity analysis as well as providing an efficient solution in sequential computations. The DASPK3.0 software can be found at `http://www.engineering.ucsb.edu/~cse`.

Some problems require the sensitivities of a single or small-dimensional output with respect to a large number of parameters. For these problems, particularly if the number of state variables is also large, the forward sensitivity approach is intractable. For a general DAE (1) and a function given by

$$G(p) = \int_0^T g(p, x, t)dt \tag{8}$$

or alternatively by the scalar function $g(p, x, T)$ at time T, these problems take the form: find $\frac{dG}{dp}$ or $\frac{dg}{dp}$, where p is a potentially large number of parameters.

The adjoint problem for (8) is derived as follows. Linearize the DAE (1) with respect to p to obtain

$$F_p + F_x x_p + F_{\dot{x}} \dot{x}_p = 0, \quad x_p(0) = x_{0p} \tag{9}$$

where subscripts on functions such as F or g are used to denote partial derivatives.

Introducing the multiplier λ, we get

$$\int_0^T \lambda^* (F_p + F_x x_p + F_{\dot{x}} \dot{x}_p)dt = 0 \tag{10}$$

where $*$ denotes the conjugate transpose. Integrating by parts, the last term in the above integral becomes

$$\int_0^T \lambda^* F_{\dot{x}} \dot{x}_p dt = (\lambda^* F_{\dot{x}} x_p)|_0^T - \int_0^T \left(\dot{\lambda}^* F_{\dot{x}} + \lambda^* \frac{dF_{\dot{x}}}{dt} \right) x_p dt, \tag{11}$$

where

$$\lambda^* \frac{dF_{\dot{x}}}{dt} = \left[\frac{d}{dt} \left(F_{\dot{x}}^* \bar{\lambda} \right) \right]^* = \left(F_{\dot{x}}^* \bar{\lambda} \right)_t^* + \left[\left(F_{\dot{x}}^* \bar{\lambda} \right)_x \dot{x} \right]^* + \left[\left(F_{\dot{x}}^* \bar{\lambda} \right)_{\dot{x}} \ddot{x} \right]^*. \tag{12}$$

A bar over a variable indicates that the variable is held fixed for the purpose of the current differentiation.

Without loss of generality, we can assume that F depends linearly on $\dot{x}$ and therefore the last term in (12) is zero. Indeed, any other case can be reduced to this one by introducing the additional variables $y = \dot{x}$. Note that, in the worst case, the problem size is increased by $\text{rank}(F_{\dot{x}})$. So from now on, we calculate $\lambda^* \frac{dF_{\dot{x}}}{dt}$ by

$$\lambda^* \frac{dF_{\dot{x}}}{dt} = \left[\frac{d}{dt} \left(F_{\dot{x}}^* \bar{\lambda} \right) \right]^* = \left(F_{\dot{x}}^* \bar{\lambda} \right)_t^* + \left[\left(F_{\dot{x}}^* \bar{\lambda} \right)_x \dot{x} \right]^*.$$

Thus we have

$$\int_0^T \left[\lambda^* (F_p + F_x x_p) - \left(\dot{\lambda}^* F_{\dot{x}} + \lambda^* \frac{dF_{\dot{x}}}{dt} \right) x_p \right] dt + (\lambda^* F_{\dot{x}} x_p)|_0^T = 0 \tag{13}$$

which can be written as

$$\int_0^T \left[\lambda^* F_p - \left(\dot{\lambda}^* F_{\dot{x}} + \lambda^* \frac{dF_{\dot{x}}}{dt} - \lambda^* F_x \right) x_p \right] dt + (\lambda^* F_{\dot{x}} x_p)|_0^T = 0. \tag{14}$$

Now letting

$$\begin{cases} \dot{\lambda}^* F_{\dot{x}} + \lambda^* \left[\frac{dF_{\dot{x}}}{dt} - F_x \right] &= -g_x \\ (\lambda^* F_{\dot{x}})|_{t=T} &= 0, \end{cases} \tag{15}$$

we obtain the equation for $\frac{dG}{dp}$

$$\frac{dG}{dp} = \int_0^T (g_p - \lambda^* F_p)\, dt + (\lambda^* F_{\dot{x}} x_p)|_{t=0} \tag{16}$$

which can be written alternatively as

$$\frac{dG}{dp} = \int_0^T (g_p - \lambda^* F_p)\, dt + (\lambda^* F_{\dot{x}})|_{t=0} x_{0p}. \tag{17}$$

Equations (15) are the so called *adjoint equations* for $\frac{dG}{dp}$. Note that the boundary condition

$$(\lambda^* F_{\dot{x}})|_{t=T} = 0$$

is applicable only for DAEs with index up to one. In [7], we develop boundary conditions that are valid also in the index-two (Hessenberg) case.

For $\frac{dg}{dp}$, we have

$$\frac{dg}{dp} = \frac{d}{dT}\frac{dG}{dp}$$

so

$$\frac{dg}{dp} = (g_p - \lambda^* F_p)(T) - \int_0^T \lambda_T^* F_p dt + (\lambda_T^* F_{\dot{x}})|_{t=0} x_{0p}, \tag{18}$$

where λ_T denotes $\frac{\partial \lambda}{\partial T}$. The corresponding adjoint equations are

$$\dot{\lambda}_T^* F_{\dot{x}} + \lambda_T^* \left[\frac{dF_{\dot{x}}}{dt} - F_x\right] = 0. \tag{19}$$

For DAEs of index up to one, to find the boundary condition for this equation, we write λ as $\lambda(t,T)$ because it depends on both t and T. Then

$$\lambda^*(T,T) F_{\dot{x}}|_{t=T} = 0.$$

Taking the total derivative, we obtain

$$(\lambda_t + \lambda_T)^*(T,T) F_{\dot{x}}|_{t=T} + \lambda^*(T,T)\frac{dF_{\dot{x}}}{dt} = 0.$$

Since λ_t is just $\dot{\lambda}$, and using the previously derived expression for $dF_{\dot{x}}/dt$, we have the boundary condition

$$(\lambda_T^* F_{\dot{x}})|_{t=T} = -\left[\lambda^*(T,T)\frac{dF_{\dot{x}}}{dt} + \dot{\lambda}^* F_{\dot{x}}\right]\Bigg|_{t=T}. \tag{20}$$

According to equation (15), the upper condition is

$$(\lambda_T^* F_{\dot{x}})|_{t=T} = (g_x - \lambda^* F_x)|_{t=T}. \tag{21}$$

In the case that $F_{\dot{x}}$ is invertible, we have $\lambda(T,T) = 0$, which leads to $\lambda_T = -\dot{\lambda}$.

Boundary conditions for the index-two (Hessenberg) case are derived in [7].

We have been developing software called ADJOINTDASPK for the reverse mode sensitivity problem. Our goal for the adjoint sensitivity calculation has been algorithms and software that are as reliable, efficient, and easy to use as the current algorithms and software for forward sensitivity analysis. An efficient formulation of the adjoint

equations can be generated by the current generation of automatic differentiation software. For example, TAMC [11] and ADIFOR3.0 [16] implement the forward and reverse modes that are needed for this task. ADJOINTDASPK makes use of the reliable and efficient methods in DASPK3.0 for solving the adjoint equations, and for determining a consistent set of initial conditions.

Several research issues are under investigation. For large-scale problems whose solution and forward sensitivities are best computed in DASPK via preconditioned iterative methods, we need to ensure that these same 'matrix-free' methods and corresponding preconditioners are accessible for the solution of the adjoint equations. Predictable and compact means of storing the forward solution information required by the adjoint computation are needed. We have been investigating the use of reduced order models both for storing the forward solution information and for highly efficient sensitivity solution in applications where repeated forward and reverse sensitivity solves are required. Determination of consistent initial conditions for the adjoint system for general DAE systems is also an issue that is in progress.

3. Design Optimization

We consider the differential-algebraic equation (DAE) system

$$\begin{aligned} \mathbf{F}(t, \mathbf{x}, \mathbf{x}', \mathbf{p}, \mathbf{u}(t)) &= 0 \\ \mathbf{x}(t_1, \mathbf{r}) &= \mathbf{x}_1(\mathbf{r}) \end{aligned} \tag{22}$$

where the DAE is index one (see [4] or [1]) or Hessenberg index-two Hessenberg, and the initial conditions have been chosen so that they are consistent (so that the constraints of the DAE are satisfied). The control parameters $\mathbf{p}$ and the vector-valued control function $\mathbf{u}(t)$ must be determined such that the objective functional

$$\int_{t_1}^{t_{\max}} \Psi(t, \mathbf{x}(t), \mathbf{p}, \mathbf{u}(t))\, dt \qquad \text{is minimized}$$

and some additional equality and/or inequality constraints

$$G(t, \mathbf{x}(t), \mathbf{p}, \mathbf{u}(t)) \geq 0$$

are satisfied. The optimal control function $\mathbf{u}^*(t)$ is assumed to be continuous. In several of our applications, the DAE system is large-scale. Thus, the dimension N_x of $\mathbf{x}$ is large. However, the dimension of the control parameters and of the representation of the control function $\mathbf{u}(t)$ is much smaller. To represent $\mathbf{u}(t)$ in a low-dimensional vector space, we use piecewise polynomials on $[t_1, t_{\max}]$, their coefficients being determined by the optimization. For ease of presentation we can therefore assume that the vector $\mathbf{p}$ contains both the parameters and these coefficients (we let M denote the combined number of these values) and discard the control function $\mathbf{u}(t)$ in the remainder of this section. Also, we consider that the initial states are fixed and therefore discard the dependency of $\mathbf{x}_1$ on $\mathbf{r}$. Hence we consider

$$\mathbf{F}(t, \mathbf{x}, \mathbf{x}', \mathbf{p}) = 0, \qquad \mathbf{x}(t_1) = \mathbf{x}_1, \tag{23}$$

$$\int_{t_1}^{t_{\max}} \psi(t, \mathbf{x}(t), \mathbf{p})\, dt \qquad \text{is minimized}, \tag{24}$$

$$\mathbf{g}(t, \mathbf{x}(t), \mathbf{p}) \geq 0. \tag{25}$$

There are a number of well-known methods for direct discretization of this optimal control problem, for the case that the DAEs can be reduced to ordinary differential equations (ODEs) in standard form. The *single shooting method* solves the ODEs (23) over the interval $[t_1, t_{\max}]$, with the set of controls generated at each iteration by the optimization algorithm. However, it is well-known that single shooting can suffer from a lack of stability and robustness [2]. Moreover, for this method it is more difficult to maintain additional constraints and to ensure that the iterates are physical or computable. The *finite-difference method* or *collocation method* discretizes the ODEs over the interval $[t_1, t_{\max}]$ with the ODE solutions at each discrete time and the set of controls generated at each iteration by the optimization algorithm. Although this method is more robust and stable than the single shooting method, it requires the solution of an optimization problem which for a large-scale ODE system is enormous, and it does not allow for the use of adaptive ODE or (in the case that the ODE system is the result of semi-discretization of PDEs) PDE software.

We thus consider the *multiple-shooting method* for the discretization of the optimal control problem. In this method, the time interval $[t_1, t_{\max}]$ is divided into subintervals $[t_{itx}, t_{itx+1}]$ $(itx = 1, \ldots, N_{tx})$, and the differential equations (23) are solved over each subinterval, where additional intermediate variables $\mathbf{X}_{itx}$ are introduced. On each subinterval we denote the solution at time t of (23) with initial value $\mathbf{X}_{itx}$ at t_{itx} by $\mathbf{x}(t, t_{itx}, \mathbf{X}_{itx}, \mathbf{p})$.

Continuity between subintervals is achieved via the continuity constraints

$$\mathbf{C}_1^{itx}(\mathbf{X}_{itx+1}, \mathbf{X}_{itx}, \mathbf{p}) \equiv \mathbf{X}_{itx+1} - \mathbf{x}(t_{itx+1}, t_{itx}, \mathbf{X}_{itx}, \mathbf{p}) = \mathbf{0}.$$

For the DAE solution to be defined on each multiple shooting subinterval, it must be provided with a set of initial values which are consistent (that is, the initial values must satisfy any algebraic constraints in the DAE). This is the case even for feasible methods, since the optimizer does not see the DAE constraints. To begin each interval with a consistent set of initial values, we first project the intermediate solution generated by SNOPT onto the DAE constraints, and then solve the DAE system over the subinterval. In the case of index-1 problems with well-defined algebraic variables and constraints such as the problem considered in this paper, this means that we perturb the intermediate initial values of the algebraic variables so that they satisfy the DAE constraints at the beginning of each multiple shooting subinterval.

The additional constraints (25) are required to be satisfied at the boundaries of the shooting intervals

$$\mathbf{C}_2^{itx}(\mathbf{X}_{itx}, \mathbf{p}) \equiv \mathbf{g}(t_{itx}, \mathbf{X}_{itx}, \mathbf{p}) \geq \mathbf{0}.$$

Following common practice, we write

$$\Phi(t) = \int_{t_1}^{t} \psi(\tau, \mathbf{x}(\tau), \mathbf{p})\, d\tau, \tag{26}$$

which satisfies $\Phi'(t) = \psi(t, \mathbf{x}(t), \mathbf{p})$, $\Phi(t_1) = 0$. This introduces another equation and variable into the differential system (23). The discretized optimal control problem becomes

$$\min_{\mathbf{X}_2, \ldots, \mathbf{X}_{Ntx}, \mathbf{p}} \Phi(t_{\max}) \tag{27}$$

subject to the constraints

$$\mathbf{C}_1^{itx}(\mathbf{X}_{itx+1}, \mathbf{X}_{itx}, \mathbf{p}) = \mathbf{0}, \tag{28}$$

$$\mathbf{C}_2^{itx}(\mathbf{X}_{itx}, \mathbf{p}) \geq \mathbf{0}. \tag{29}$$

This problem can be solved by an optimization code. We use the solver SNOPT [13], which incorporates a sequential quadratic programming (SQP) method (see [14]). The SQP methods require a gradient and Jacobian matrix that are the derivatives of the objective function and constraints with respect to the optimization variables. We compute these derivatives via our differential-algebraic equation (DAE) sensitivity software DASPK3.0 described earlier. Our algorithms and software for the optimal control of dynamical systems are described in detail in [29].

This basic multiple-shooting type of strategy can work very well for small-to-moderate size ODE systems, and has an additional advantage that it is inherently parallel. However, for large-scale ODE and DAE systems there is a problem because the computational complexity grows rapidly with the dimension of the ODE system. The difficulty lies in the computation of the derivatives of the continuity constraints with respect to the variables $\mathbf{X}_{itx}$. The work to compute the derivative matrix $\partial \mathbf{x}(t)/\partial \mathbf{X}_{itx}$ is of order $\mathcal{O}(N_x^2)$, and for the problems under consideration N_x can be very large (for example, for an ODE system obtained from the semi-discretization of a PDE system, N_x is the product of the number of PDEs and the number of spatial grid points). In contrast, the computational work for the single shooting method is of order $\mathcal{O}(N_x N_p)$ although the method is not as stable, robust or parallelizable.

We reduce the computational complexity of the multiple shooting method for this type of problem by modifying the method to make use of the structure of the continuity constraints to reduce the number of sensitivity solutions which are needed to compute the derivatives[12]. To do this, we recast the continuity constraints in a form where only the matrix-vector products $(\partial \mathbf{x}(t)/\partial \mathbf{X}_{itx})\mathbf{w}_j$ are needed, rather than the entire matrix $\partial \mathbf{x}(t)/\partial \mathbf{X}_{itx}$. The matrix-vector products are directional derivatives; each can be computed via a single sensitivity analysis. The number of vectors $\mathbf{w}_j$ such that the directional sensitivities are needed is small, of order $\mathcal{O}(N_p)$. Thus the complexity of the modified multiple shooting computation is reduced to $\mathcal{O}(N_x N_p)$, roughly the same as that of single shooting. Unfortunately, the reduction in computational complexity comes at a price: the stability of the modified multiple shooting algorithm suffers from the same limitations as single shooting. However, for many DAE and partial differential algebraic equation (PDAE) systems, where simulation of the forward problem is stable, this is not an issue, and the modified method is more robust for nonlinear problems. The inherent parallelism of the multiple shooting algorithm is also lost by the modified method.

In the context of the SQP method, the use of modified multiple shooting involves a transformation of the constraint Jacobian. The affected rows are those associated with the continuity constraints and any path constraints applied within the shooting intervals. Path constraints enforced at the shooting points (and other constraints involving only discretized states) are not transformed. The transformation is cast almost entirely at the user level and requires minimal changes to the optimization software, which is important because software in this area is constantly being modified and improved. Gill *et.al.* ([12]) have shown that the modified quadratic subproblem yields a descent direction for the ℓ_1 penalty function. DAOPT is a modification to the SNOPT optimization code that uses a merit function based on an ℓ_1 penalty function.

A limitation of the basic COOPT software arises for PDAE systems because the method of lines is used to solve the PDAE. This does not allow for an adaptive grid in space, which would be needed if there are steep spatial gradients moving in time. We have developed methods and software based on adaptive mesh refinement (AMR) for systems of partial differential equations[18]. The software has been designed to be

flexible, to allow the user to 'plug and play' existing non-adaptive simulation software, and to exert direct control over the adaptivity when needed. Recently we have shown how to compute sensitivity derivatives via AMR [19], and developed a capability for doing dynamic optimization including AMR for time-dependent PDE systems[21].

4. Partitioning for Structure and Stability

An important issue for shooting or multiple shooting-type methods in design optimization is the handling of constraints[27]. Given an equality constraint, it can be included in the dynamic optimization problem either as part of the DAE, or to be handled directly by the optimizer. If it is included as part of the DAE, then there is a possibility that it could alter the index (mathematical structure) of the DAE, making it potentially more difficult to solve; on the other hand, if it is an important physical constraint, then its inclusion into the DAE where it will always be enforced should help the optimizer to avoid non-physical solutions.

In our work, this question has been studied from the point of view of mathematical structure (index), and stability/conditioning of the DAE. It turns out that including some constraints with the DAE can alter the DAE stability, in a way which may be either favorable or unfavorable. Algorithms have been developed for partitioning the constraints to lead to a stable DAE system of index two or lower (which can be solved by the existing software). The logarithmic norm, which is closely related to the pseudoeigenvalues, is used to measure the stability for potential partitionings. We have developed a partitioning tool for analyzing the DAE structure and determining a stable partitioning of the constraints.

The dilemma of how best to treat the constraints arises often in the dynamic optimization of path-constrained dynamics systems. For example, a planar rigid body model of a crane manipulating a payload which must be lifted along a specified trajectory, is described below [6].

4.1. Example: Planar Model of a Crane

The crane is a mechanical system in planar Cartesian co-ordinates, x and z, denoting the horizontal and vertical positions of the payload. The other state variables are d, the horizontal distance traveled by the crane trolley from the origin and r, the paid out cable length. The equations are

$$
\begin{aligned}
M_2\ddot{x} &= -\tau\sin(\theta) && \text{(30a)}\\
M_2\ddot{z} &= -\tau\cos(\theta) + mg && \text{(30b)}\\
M_1\ddot{d} &= -C_1\dot{d} + u_1 + \tau\sin(\theta) && \text{(30c)}\\
J\ddot{r} &= -C_2\dot{r} - C_3u_2 + C_3^2\tau && \text{(30d)}\\
0 &= \theta - \tan^{-1}((x-d)/z) && \text{(30e)}\\
0 &= r^2 - (x-d)^2 - z^2 && \text{(30f)}\\
x &= \phi_1(t) && \text{(30g)}\\
z &= \phi_2(t) && \text{(30h)}
\end{aligned}
$$

where M_1, M_2, m are the masses of the trolley, cable and payload system and the payload alone, respectively. C_1, C_2, C_3 are constants, and J is the mass moment of

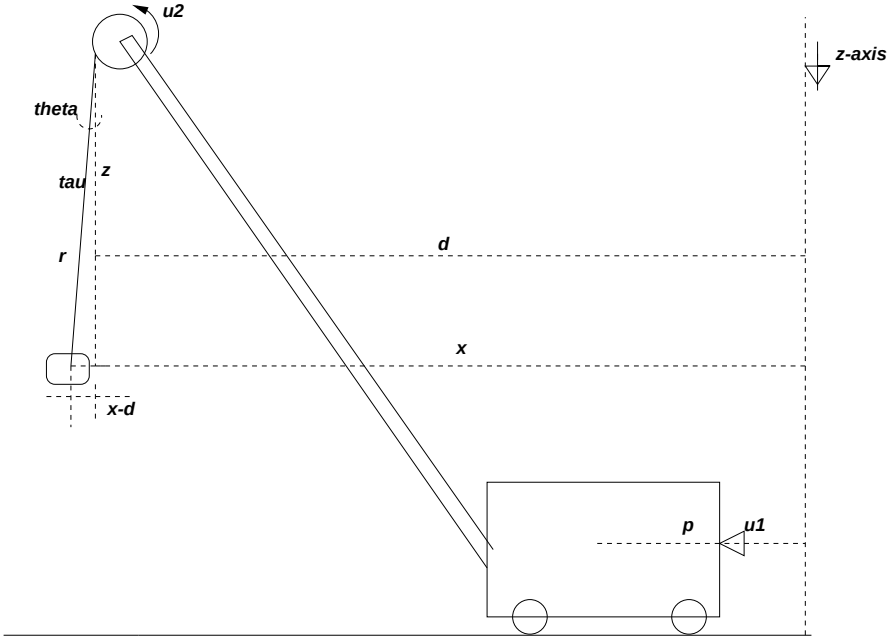

Figure 1: Planar Crane Model

inertia for the pulley paying out the cable. The algebraic variables are τ, the tension in the cable and θ, the cable angle with vertical. The horizontal force driving the trolley (u_1) and torque driving the pulley in the winch (u_2) are the two control variables. Equations (30g) and (30h) describe the specified path of the payload. The initial values were taken to be $x = 3.0, \dot{x} = 2.7, z = 21.0, \dot{z} = -2.0, d = 1.0, \dot{d} = 2.7, r = ((x-d)^2 + z^2)^{0.5}, \dot{r} = -2.0, \tau = 980.0, u_1 = 0, u_2 = 0, \theta = \tan^{-1}\left(\frac{(x-d)}{z}\right)$, on the interval of integration from $t_0 = 0$ to $t_{\text{final}} = 8$.

The path-constrained crane model 30 is an index-5 system. Thus it is not directly solvable by any of the methods described earlier.

Figure (2) illustrates the error ($\int_{t0}^{t_{final}} ||r^2 - (x-d)^2 - z^2)||_2 dt$) in simulating the planar crane problem with an index 1 partitioned DAE (pDAE) consisting of (30a - 30e) only and excluding equation (30f).

While the physical constraints inherently should be included with the pDAE, they can raise the index of the pDAE to higher than 2. As an example, equation (30f) in the crane example when included in the pDAE (30a - 30e) raises the index of the system to 3. When these constraints raise the index of the pDAE to higher than 2, a suitable index reduction method can be adopted.

For the crane model, constraints (30e) and (30f) are both scleronomic constraints, i.e., they describe some geometric structure of the physical problem. Constraint (30e) introduces the measurement of θ, the angle by which the payload deviates from the vertical. Constraint (30f) connects the payload to the pulley, by relating its horizontal and vertical positions to the length of the cable paid out. Violation of constraint (30f) in the pDAE would produce an unphysical simulation since the payload is now modeled as detached from the cable. This is evident from the plot of x, z, d and r in Figure (5).

Applying our partitioning algorithm to this model, constraint (30e) is included in the pDAE since it satisfies the index one structure and stability criterion. The angle of deflection from vertical θ is chosen as the index 1 variable. The resulting underlying ODE in x does not have large eigenvalues in $\mathcal{C}^+$ at $t = t_0$. For a practical physical application, the eigenvalues of the pDAE system can be expected to be of moderate

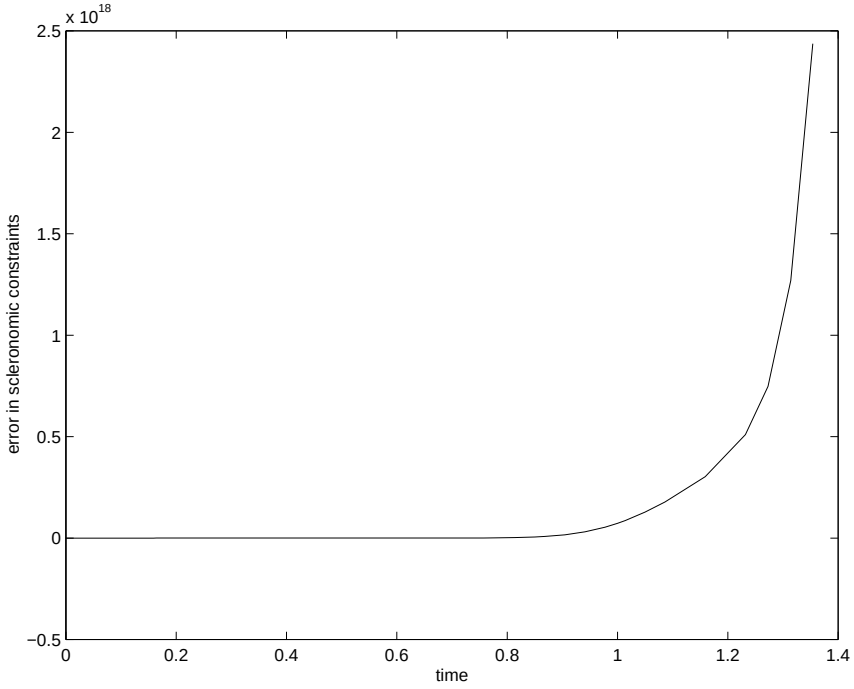

Figure 2: Partial plot of errors neglecting scleronomic constraint

size and the system can be expected to be stable. The constraint equation (30f) is an index 3 scleronomic constraint (known from the physics of the problem) and is reduced to its index 2 form via one differentiation of constraint (30f) with respect to t

$$r\dot{r} - (x - d)(\dot{x} - \dot{d}) - z\dot{z} = 0 \tag{31}$$

or to index 1 via two differentiations of constraint (30f) with respect to t. (if the available DAE integration software can integrate index 0 or 1 systems only). The other constraints (pre-programmed trajectory of the payload) (30g and 30h) are handled by the SQP method.

The search for index 2 constraints returns the empty set. To search for index 3 scleronomic constraints, the remaining algebraic equations are differentiated using automatic differentiation once with respect to time. The index-2 search algorithm is re-applied and the algorithm returns the differentiated form of constraint (30f), i. e., (31) as an index-2 constraint suitable for inclusion in the pDAE system. The tension in the cable τ is returned as the corresponding index 2 variable.

Even though the scleronomic constraint (31) slightly raises the logarithmic norm estimate for the index 2 pDAE system thus constructed, this partition leads to a more physical and well conditioned problem. The plots in Figures (3) and (4) are the results obtained from a multiple shooting type scheme using DASPK3.0 [17] as the DAE integrator and SNOPT as the NLP method [26]. In this case the path constraints are chosen as $p_1 \equiv x - (-0.0675t^2 + 2.7t + 3.0) = 0$ and $p_2 \equiv z - (-21.0 - 0.05t^2 + 2.0t) = 0$. The optimizer SNOPT stops at an acceptable point which cannot be improved further after 4 major iterations (with 188, 182, 112 and 1 minor iterations). The objective function is given by the error integral $\int_{t_0}^{t_{final}} (p_1^2 + p_2^2)^{0.5} dt$. The time interval is $0 \leq t \leq 8$. The controls (u_1 and u_2) have been modeled as quadratic polynomials over each of the 8 shooting intervals used for the problem. The tolerances for DASPK3.0 were $rtol = 10^{-7}$ and $atol = 10^{-7}$ and all tolerances for SNOPT were 10^{-5}. The control and state conti-

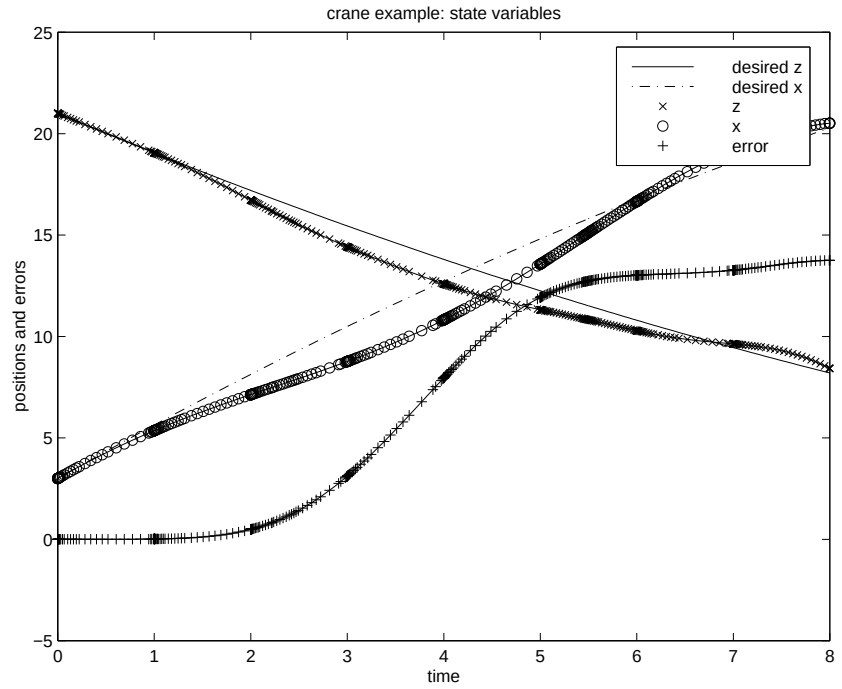

Figure 3: Crane problem using equations (30a)-(30e) and (31) in the pDAE.

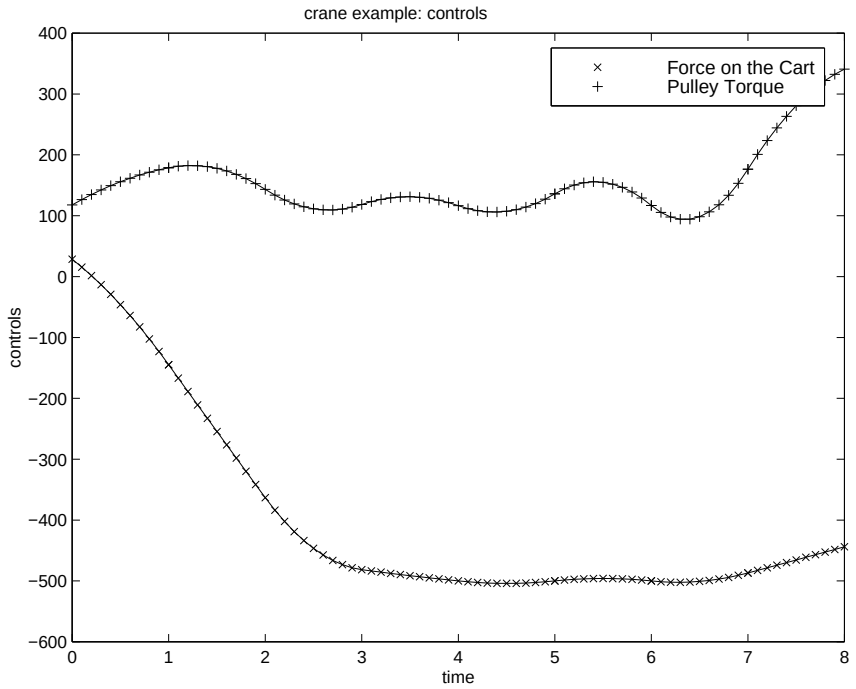

Figure 4: Crane controls using equations (30a)-(30e) and (31) in the pDAE.

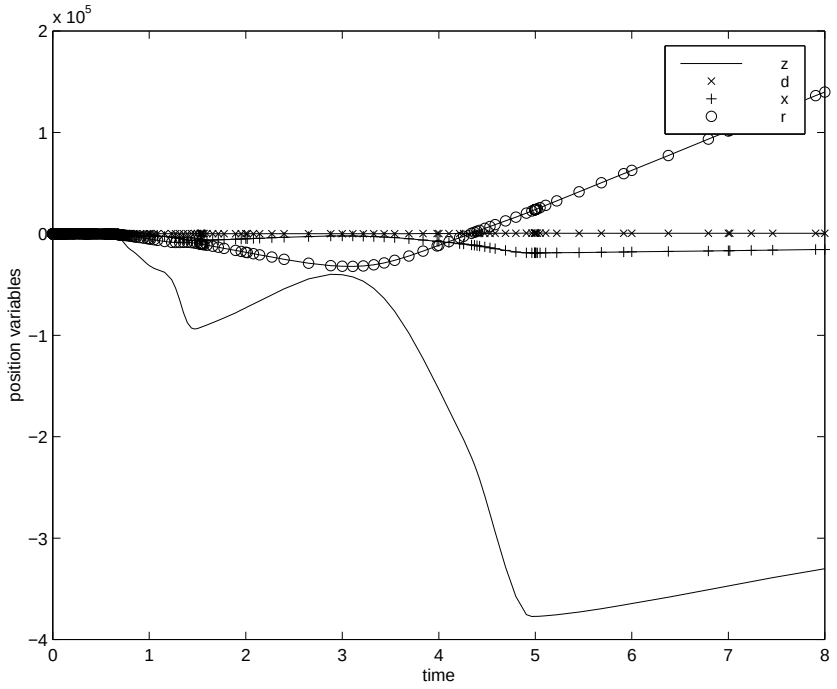

Figure 5: Crane problem using equations (30a)-(30e) in the pDAE.

nuity constraints across the shooting intervals have been satisfied to less than or equal to 0.1 when the optimizer has stopped.

With the same parameter settings for the numerical methods, a pDAE formulation leaving (30f) with the optimizer as a path constraint and treating τ as an additional control variable has been attempted. The plot in Figure (5) is obtained from the results. The solution is incorrect. After 7 major iterations in SNOPT (162, 50, 72, 8, 1, 9 and 22 minor iterations), the optimizer fails to find a descent direction. Clearly the stopping point is an unacceptable solution to the physical problem. Many of the continuity constraints in the multiple shooting method are unacceptably violated.

References

[1] U. M. Ascher and L. R. Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, SIAM, Philadelphia 1998. ISBN 0-89871-412-5.

[2] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, SIAM, Philadelphia 1995. ISBN 0-89871-354-4.

[3] C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland, ADIFOR—Generating Derivative Codes from Fortran Programs, *Scientific Programming* **1** (1992), 11-29.

[4] K. E. Brenan, S. L. Campbell, and L. R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, SIAM, Philadelphia 1995. ISBN 0-89871-353-6.

[5] P. N. Brown, A. C. Hindmarsh and L. R. Petzold, Using Krylov Methods in the Solution of Large-Scale Differential-Algebraic Systems, *SIAM J. Sci. Comput.* (1994), 1467-1488.

[6] S. L. Cambell, High Index Differential Algebraic Equations, *Mech. Struct. and Mach.* **23:2** (1995), 199-222.

[7] Y. Cao, S. Li, L. Petzold and R. Serban, Adjoint Sensitivity Analysis for Differential-Algebraic Equations: Part I, The Adjoint System, in preparation.

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

[9] N. Doss, W. Gropp, E. Luck, and A. Skjellum, A model implementation of MPI, Technical report Argonne National Laboratory, 1993.

[10] W. F. Feehery, J. E. Tolsma and P. I. Barton, Efficient sensitivity analysis of large-scale differential-algebraic systems, *Applied Numerical Mathematics* **25** (1997) 41-54.

[11] R. Giering and T. Kaminski, Recipes for adjoint code construction, *ACM Trans. Math. Software* **24** (1998), 437-474.

[12] P. E. Gill, L. O. Jay, M. W. Leonard, L. R. Petzold and V. Sharma, An SQP Method for the Optimal Control of Large-Scale Dynamical Systems, *J. Comp. Appl. Math.* **20** (2000), 197-213.

[13] P. E. Gill, W. Murray, and M. A. Saunders, SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization, Numerical Analysis Report 97-2, Department of Mathematics, University of California, San Diego, La Jolla, CA, 1997.

[14] P. E. Gill, W. Murray, and M. H. Wright, Practical Optimization, Academic Press, London and New York, 1981. ISBN 0-12-283952-8.

[15] A. C. Hindmarsh, personal communication.

[16] P. Hovland, Argonne National Laboratory, personal communication, 1999.

[17] S. Li and L.R. Petzold, Software and Algorithms for Sensitivity Analysis of Large-Scale Differential Algebraic Systems, to appear, *J. Comp. Appl. Math.*.

[18] S. Li, Adaptive Mesh Methods and Software for Time-Dependent PDEs, Ph.D. thesis, Department of Computer Science, University of Minnesota, 1998.

[19] S. Li, L. R. Petzold and J. M. Hyman, Solution Adapted Nested Grid Refinement and Sensitivity Analysis for Parabolic PDE Systems, in preparation.

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

[21] S. Li, R. Serban and L. Petzold, Optimal control for time-dependent partial differential equations with implicit adaptive mesh refinement, in preparation.

[22] M. Lo and S.Ross, Low Energy Interplanetary Transfers Using Invariant Manifolds of L1, L2, and Halo Orbits, AAS/AIAA Space Flight Mechanics Meeting, Monterey, Ca, 9-11 February 1998.

[23] R. Serban, W. S. Koon, M. Lo, J. E. Marsden, L. R. Petzold, S. D. Ross and R. S. Wilson, Optimal Control for Halo Orbit Missions, Proc. IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control, March 16-18, 2000, Princeton Univ.

[24] T. Maly and L. R. Petzold, Numerical methods and software for sensitivity analysis of differential-algebraic systems, *Applied Numerical Mathematics* **20** (1996), 57-79.

[25] S. E. Mattson, and G. and Söderlind, Index reduction in differential-algebraic equations using dummy derivatives, *SIAM J. Sci. Comput.* **14:3** (1993), 677-692.

[26] L.R. Petzold, J. B. Rosen, P. E. Gill, L. O. Jay and K. Park, Numerical Optimal Control of Parabolic PDEs using DASOPT, Large Scale Optimization with Applications, Part II: Optimal Design and Control, Eds. L. Biegler, T. Coleman, A. Conn and F. Santosa, IMA Volumes in Mathematics and its Applications, 93 (1997), 271-300.

[27] S. Raha, Constraint Partitioning for Solution of Path-Constrained Dynamic Optimization Problems, Ph.D. Thesis, University of Minnesota, Scientific Computation, 2000.

[28] L. Raja, R. Kee, R. Serban, and L.R. Petzold Dynamic Optimization of Chemically Reacting Stagnation Flows, 1998 Electrochemical Society Conference, Boston, Ma.

[29] R. Serban, COOPT - Control and Optimization of Dynamic Systems - Users' Guide, UCSB, Department of Mechanical and Environmental Engineering, Report UCSB-ME-99-1,1999.