NUMERICAL OPTIMAL CONTROL OF PARABOLIC PDES USING DASOPT*

LINDA PETZOLD†, J. BEN ROSEN‡, PHILIP E. GILL§, LAURENT O. JAY¶, AND KIHONG PARK‖

Abstract. This paper gives a preliminary description of DASOPT, a software system for the optimal control of processes described by time-dependent partial differential equations (PDEs). DASOPT combines the use of efficient numerical methods for solving differential-algebraic equations (DAEs) with a package for large-scale optimization based on sequential quadratic programming (SQP). DASOPT is intended for the computation of the optimal control of time-dependent nonlinear systems of PDEs in two (and eventually three) spatial dimensions, including possible inequality constraints on the state variables. By the use of either finite-difference or finite-element approximations to the spatial derivatives, the PDEs are converted into a large system of ODEs or DAEs. Special techniques are needed in order to solve this very large optimal control problem. The use of DASOPT is illustrated by its application to a nonlinear parabolic PDE boundary control problem in two spatial dimensions. Computational results with and without bounds on the state variables are presented.

Key words. differential-algebraic equations, optimal control, nonlinear programming, sequential quadratic programming, partial differential equations.

AMS subject classifications. 34A09, 34H05, 49J20, 49J15, 49M37, 49D37, 65F05, 65K05, 90C, 90C30, 90C06, 90C90

1. Introduction. We describe a numerical method (DASOPT) for finding the solution of a general optimal control problem. We assume that the problem is described with an objective function that must be minimized subject to constraints involving a system of DAEs and (possibly) inequality constraints. The numerical method uses the general-purpose packages DASPKSO (§4) and SNOPT (§3) in an essential way, and takes full advantage of their capabilities.

In the method proposed, large-scale nonlinear programming is used to solve the optimization/optimal control problem. The original time interval is divided into subintervals in a multiple-shooting type approach that provides a source of parallelism. (For other approaches, see, e.g., Dickmanns and Well [11], Kraft [20], Hargraves and Paris [19], Pesch [28], Lamour [21], Betts and Huffman [3], von Stryk and Bulirsch [35], Bulirsch *et al.* [9], von Stryk [34], Betts [2], Brenan [6], Schulz, Bock and Steinbach [30], Tanartkit and Biegler [32], Pantelides, Sargent and Vassiliadis [27], and Gritsis, Pantelides and Sargent [18].)

*This research was partially supported by National Science Foundation grants CCR-95-27151 and DMI-9424639, National Institute of Standards and Technology contract 60 NANB2D 1272, Department of Energy grant FG02-92ER25130, Office of Naval Research grants N00014-90-J-1242 and N00014-96-1-0274, the Minnesota Supercomputing Institute, and the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory Cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

†Department of Computer Science and Army High Performance Computing Research Center, University of Minnesota, Minneapolis, Minnesota 55455.

‡Department of Computer Science, University of Minnesota, Minneapolis, Minnesota 55455, and Department of Computer Science and Engineering, University of California, San Diego, La Jolla, California 92093-0114.

§Department of Mathematics, University of California, San Diego, La Jolla, California 92093-0112.

¶Department of Computer Science, University of Minnesota, Minneapolis, Minnesota 55455.

‖School of Mechanical Engineering, Kookmin University, Seoul, Korea.

The associated finite-dimensional optimization problem is characterized by: (a) many variables and constraints; (b) sparse constraint and objective derivatives; and (c) many constraints active at the solution. The optimization problem is solved using the package SNOPT (§3), which is specifically designed for this type of problem. SNOPT uses a sequential quadratic programming (SQP) method in conjunction with a limited-memory quasi-Newton approximation of the Lagrangian Hessian. There has been considerable interest elsewhere in extending SQP methods to the large structured problems. Much of this work has focused on reduced-Hessian methods, which maintain a dense quasi-Newton approximation to a smaller dimensional *reduced* Hessian (see, e.g., Biegler, Nocedal and Schmidt [4], Eldersveld [12], Tjoa and Biegler [33], and Schultz [29]). Our preference for approximating the full Hessian is motivated by substantial improvements in reliability and efficiency compared to earlier versions of SNOPT based on the reduced-Hessian approach.

The function and derivative computations for the optimization involve computing the solution of a large-scale DAE system, and solution sensitivities with respect to the initial conditions and the control parameters. The general-purpose package DASPKSO (§4) is used to compute the DAE solution and sensitivities. The sensitivity equations can be solved very efficiently, and in parallel with the original DAE.

In §5, a typical application is described, consisting of a nonlinear parabolic PDE in two spatial dimensions, with boundary control of the interior temperature distribution. This application serves as an initial test problem for DASOPT, and has the important feature that the size of the problem is readily increased by simply using a finer spatial grid size. It is shown in §5 how the PDE is reduced to a suitable finite-dimensional optimization problem. The numerical results, obtained by DASOPT for ten related cases, are summarized in §6. These results are displayed in ten figures that show, as a function of time, the optimal control and the temperatures at interior points obtained with different constraints and degrees of nonlinearity.

We assume that the continuous problem is given in the form

$$\begin{aligned} \underset{u,v}{\text{minimize}} \quad & \phi(u) = \int_0^{t_{\max}} \psi(v,u,t)\,dt \\ \text{subject to} \quad & v(0) = v_0, \\ & f(v,v',u,t) = 0, \quad t \in [0,t_{\max}], \qquad (1.1a) \\ & g(v,u,t) \geq 0, \quad t \in [0,t_{\max}]. \qquad (1.1b) \end{aligned}$$

It is assumed that given the initial condition v_0 and the control function $u = u(t)$, $t \in [0, t_{\max}]$, the state vector function $v = v(t)$ is uniquely determined by the DAE system (1.1a). Conditions on f that ensure this are discussed, for example, in Brenan, Campbell and Petzold [7]. We also assume that the control $u(t)$ satisfies some standard conditions needed for the existence of an optimal control (see, e.g., Leitman [23]).

For simplicity of presentation, we assume that v_0 is given and fixed. However, there is no difficulty in treating v_0 as a vector of parameters to be determined by the optimization. Note also that $\phi(u)$ is most easily computed by adding the single differential equation

$$\nu' = \psi(v,u,t), \quad \nu(0) = 0 \qquad (1.2)$$

to the system (1.1a). Then $\phi(u) = \nu(t_{\max})$. It follows that the control function $u(t)$ determines the objective function $\phi(u)$.

Throughout this paper, the optimal control is assumed to be continuous, which is typical of the processes that we will be investigating. Additional restrictions on $u(t)$ and $v(t)$ are specified by the inequalities (1.1b). These will almost always include upper and lower bounds on $u(t)$, and may include similar bounds on the state vector $v(t)$. In general, it is computationally much easier to enforce constraints on $u(t)$ than constraints that involve $v(t)$.

In the applications considered here, the size of the DAE system (1.1a) may be large. However, typically the dimension of the control vector $u(t)$ will be much smaller. In order to be able to represent $u(t)$ in a low-dimensional vector space, it will be represented by a spline function, or a piecewise polynomial on $[0, t_{\max}]$. The coefficients of this spline or piecewise polynomial are determined by the optimization. If $p \in I\!R^{n_p}$ denotes the vector of coefficients, then both $u(t)$ and the objective $\phi(u)$ are completely determined by p, with

$$u(t) = \bar{u}(p, t), \quad \phi(u) = \theta(p). \tag{1.3}$$

The optimization problem given by (1.1) can then be considered as that of minimizing $\theta(p)$, subject to the inequality constraints (1.1b).

2. Discretizing the control problem. There are a number of alternative methods for discretizing the control problem. The first, known as the single shooting, or "nested" method, minimizes over the control variables and solves the DAE system (1.1a) over $[0, t_{\max}]$, given the set of control variable approximations generated at each iteration of the optimization algorithm. This approach can be used in conjunction with adaptive DAE software, and when it converges, it can be very efficient. However, it is well-known that single shooting can suffer from a lack of robustness and stability (see, e.g., Ascher, Mattheij and Russell [1]). For some nonlinear problems it can generate intermediate iterates that are nonphysical and/or not computable. For some well-conditioned boundary-value problems, it can generate unstable initial-value DAEs. Two classes of algorithms have been proposed to remedy these problems. One is the multiple shooting method, in which the initial time interval is divided into subintervals and the DAE (1.1a) is solved over each subinterval. Continuity is achieved between the subintervals by adding the continuity conditions as constraints in the optimization problem. The other is the collocation method, in which the solution and its derivative are approximated via a collocation formula defined directly on a fine grid over the whole interval. In this case, the optimization is performed over both the control variables and the discretized solution variables.

In the DASOPT project, our aim is to develop software for the optimization of several classes of nonlinear time-dependent PDEs. We have chosen to implement the multiple shooting method (with single shooting as a special case). This method was selected not only because of its stability and robustness, but also because it allows the use of existing adaptive DAE and PDE software. Another substantial benefit is that the resulting optimization problems are more tractable than those generated by the collocation method—especially in the optimization of PDE systems. A disadvantage of the straightforward implementation of multiple shooting considered here is that it may be necessary to compute n_v^2 sensitivities at each optimization iteration, where n_v is the dimension of v (and the number of DAEs in (1.1a)). A more sophisticated approach that has the complexity of single shooting and the stability and robustness of multiple shooting will be the subject of a future paper. The reader should recognize that the timing results for the test problem in §6 are not optimal, but reflect the current status of the DASOPT software.

For multiple shooting, the total time interval $[0, t_{\max}]$ is divided into N equal subintervals of length Δt each. Then

$$t_k = k\Delta t, \quad k = 0, 1, \ldots, N, \tag{2.1}$$

with $t_N = N\Delta t = t_{\max}$. The system of DAEs (1.1a) is now solved as an independent subproblem over each subinterval $[t_k, t_{k+1}]$, with its own initial conditions. A continuous solution over $[0, t_{\max}]$ is obtained by matching the initial conditions at t_k with the final values obtained from the previous subinterval $[t_{k-1}, t_k]$. This matching is included in the optimization, where the initial values of v for each subinterval are additional optimization variables.

To be more specific, let $v_k(t)$ denote the solution of the DAE system (1.1a) on the time subinterval $[t_k, t_{k+1}]$, with the initial conditions

$$v_0(0) = v_0, \quad v_k(t_k) = \bar{v}_k, \quad k = 1, 2, \ldots, N-1. \tag{2.2}$$

The value of $\bar{v}_0 = v_0$ is given, and the $\bar{v}_k$, $k = 1, 2, \ldots, N-1$, are to be determined. Let the vector $\bar{u}_k$ denote the coefficients of the spline or polynomial $u_k(\bar{u}_k, t)$ that represents $u(t)$ for $t \in [t_k, t_{k+1}]$. For example, in the application discussed in §5, if n_u denotes the dimension of u, then each $u_k(t)$ is the quadratic polynomial

$$u_k(t) = \bar{u}_{k0} + \bar{u}_{k1}(t - t_k) + \bar{u}_{k2}(t - t_k)^2, \quad \text{for } \; t \in [t_k, t_{k+1}], \tag{2.3}$$

with $\bar{u}_{k0}$, $\bar{u}_{k1}$, and $\bar{u}_{k2}$ each of order n_u. It follows that $u_k(t)$ can be represented by the $3n_u$ vector $\bar{u}_k$ formed from $\bar{u}_{k0}$, $\bar{u}_{k1}$, and $\bar{u}_{k2}$. The N vectors $\bar{u}_k$, $k = 0, 1, \ldots, N-1$ are determined by the optimization. The continuity of the $u_k(t)$ and their first derivatives is imposed by the linear equality constraints

$$\left.\begin{array}{rcl} \bar{u}_{k+1,0} & = & \bar{u}_{k0} + \bar{u}_{k1}\Delta t + \bar{u}_{k2}(\Delta t)^2 \\ \bar{u}_{k+1,1} & = & \bar{u}_{k1} + 2\bar{u}_{k2}\Delta t, \end{array}\right\} \qquad k = 0, 1, \ldots, N-2. \tag{2.4}$$

Bounds on the $u_k(t)$ at $t = t_k$ (and any additional points) give linear inequalities on the $\bar{u}_{kl}$.

Given $\bar{v}_k$ and $\bar{u}_k$, the DAE system (1.1a) gives $v_k(t_{k+1})$. Making this dependence explicit we have

$$v_k(t_{k+1}) = s(\bar{v}_k, \bar{u}_k). \tag{2.5}$$

The matching conditions, to enforce continuity of $v(t)$ at the subinterval boundaries, then become

$$s(\bar{v}_k, \bar{u}_k) - \bar{v}_{k+1} = 0, \quad k = 0, 1, \ldots, N-1. \tag{2.6}$$

The last of these constraints involves the vector $\bar{v}_N$ at the point $t_{\max}$. This vector does not specify an initial value for the differential equation, but imposes a condition on $s(\bar{v}_{N-1}, \bar{u}_{N-1})$ arising from either an explicit condition on $v(t_{\max})$ or a condition on v from the inequality constraint $g \geq 0$ below. If these constraints are not present, $\bar{v}_N$ can be a free variable in the optimization. Note that since the DAE solutions over each subinterval are independent, they can be computed in parallel.

The inequality constraints (1.1b) can now be imposed explicitly at each subinterval boundary, as requirements on the vectors $\bar{v}_k$ and $\bar{u}_k$. These become

$$g(\bar{v}_k, u_k(t_k), t_k) \geq 0, \quad k = 0, 1, \ldots, N-1, \tag{2.7a}$$

$$g(\bar{v}_N, u_{N-1}(t_N), t_N) \geq 0. \tag{2.7b}$$

Finally the objective function is determined by solving the ODE (1.2) as an additional part of the DAE system (1.1a). That is, we solve

$$\nu_k(t_k) = 0, \quad \nu_k' = \psi(v_k(t), u_k(t), t), \tag{2.8}$$

for $t \in [t_k, t_{k+1}]$. This gives the objective function as $\sum_{k=0}^{N-1} \nu_k(t_{k+1})$.

Let p denote the vector of variables associated with the finite-dimensional optimization problem. This vector has the form

$$p = (\bar{u}_0, \bar{v}_1, \bar{u}_1, \bar{v}_2, \ldots, \bar{u}_{N-1}, \bar{v}_N)^T,$$

with the total number of optimization variables given by $n_p = N(n_v + n_{\bar{u}})$ where $n_{\bar{u}}$ is the dimension of each $\bar{u}_k$. The discretized problem may be written in the general form

$$\begin{array}{ll} \underset{p \in \mathbb{R}^{n_p}}{\text{minimize}} & \theta(p) \\ \text{subject to} & b_l \leq \left\{ \begin{array}{c} p \\ Ap \\ r(p) \end{array} \right\} \leq b_u, \end{array} \tag{2.9}$$

where r is a vector of nonlinear functions, A is a constant matrix that defines the linear constraints, and b_l and b_u are constant upper and lower bounds. The vector r comprises the matching conditions (2.6) and the components of g (2.7). The components of b_l and b_u are set to define the appropriate constraint right-hand side. For example, $(b_l)_i = (b_u)_i = 0$ for the matching conditions, and $(b_l)_i = 0$, $(b_u)_i = +\infty$ for components of g. The matrix A contains the linear equality constraints associated with the continuity conditions (2.4) and any linear inequality constraints on $\bar{u}_k$ resulting from upper and lower bounds on $u(t)$. Upper and lower bounds on $v(t)$ are imposed directly as bounds on $\bar{v}_k$.

The optimization requires, in addition to the function evaluations, that both the gradient of the objective function and the Jacobian of the constituent functions be computed at each major iteration. We need the Jacobian of $s(\bar{v}_k, \bar{u}_k)$, which is typically dense. Since $s \in \mathbb{R}^{n_v}$, $\bar{v}_k \in \mathbb{R}^{n_v}$ and $\bar{u}_k \in \mathbb{R}^{n_{\bar{u}}}$, $n_v(n_v + n_{\bar{u}})$ sensitivity evaluations are required. The value of n_v may be large, so this may be the most significant part of the total computation. This is illustrated in §5, where n_v is the total number of spatial grid points in the two-dimensional PDE. A modification of the multiple shooting method that has complexity comparable to that of single shooting is under development and will be the subject of a future paper.

The gradients of $\theta(p)$ with respect to the $\bar{v}_k$ and $\bar{u}_k$ are computed similarly and they involve the sensitivities required for the Jacobian as well, so this is also an $O(n_v(n_v + n_{\bar{u}}))$ calculation.

3. Solving the optimization problem. In this section we discuss the application of the general-purpose sparse nonlinear optimizer SNOPT to solve the discretized optimal control problem. The discretized problem of §2 has several important characteristics: (a) many variables and constraints; (b) sparse constraint and objective derivatives; (c) objective and constraint functions (and their first derivatives) that are expensive to evaluate; and (d) many constraints binding at the solution. SQP methods are particularly well suited to problems with these characteristics.

At a constrained minimizer p^*, the objective gradient $\nabla\theta$ can be written as a linear combination of the constraint gradients. The multipliers in this linear combination are known as the *Lagrange multipliers*. The Lagrange multipliers for an

upper bound constraint are nonpositive, the multipliers for a lower bound constraint are nonnegative. The vector of Lagrange multipliers associated with the *nonlinear* constraints of (2.9) is denoted by π^*.

As their name suggests, SQP methods are a class of optimization methods that solve a quadratic programming subproblem at each iteration. Each QP subproblem minimizes a quadratic model of a certain modified Lagrangian function subject to linearized constraints. A merit function is reduced along each search direction to ensure convergence from any starting point. The basic structure of an SQP method involves *major* and *minor* iterations. The major iterations generate a sequence of iterates (p_k, π_k) that converge to (p^*, π^*). At each iterate a QP subproblem is used to generate a search direction towards the next iterate (p_{k+1}, π_{k+1}). Solving such a subproblem is itself an iterative procedure, with the *minor* iterations of an SQP method being the iterations of the QP method. (For an overview of SQP methods, see, for example, Gill, Murray and Wright [17].)

Each QP subproblem minimizes a quadratic model of the *modified Lagrangian*

$$\mathcal{L}(p, p_k, \pi_k) = \theta(p) - \pi_k^T d_L(p, p_k), \tag{3.1}$$

which is defined in terms of the *constraint linearization*,

$$r_L(p, p_k) = r(p_k) + J(p_k)(p - p_k),$$

and the *departure from linearity*, $d_L(p, p_k) = r(p) - r_L(p, p_k)$.

Given estimates (p_k, π_k) of (p^*, π^*), an improved estimate is found from $(\widehat{p}_k, \widehat{\pi}_k)$, the solution of the following QP subproblem:

$$\begin{aligned} \underset{p \in \mathbb{R}^n}{\text{minimize}} \quad & \theta(p_k) + \nabla\theta(p_k)^T(p - p_k) + \tfrac{1}{2}(p - p_k)^T H_k (p - p_k) \\ \text{subject to} \quad & b_l \leq \begin{Bmatrix} p \\ Ap \\ r(p_k) + J(p_k)(p - p_k) \end{Bmatrix} \leq b_u, \end{aligned}$$

where H_k is a positive-definite approximation to $\nabla^2_p \mathcal{L}(p_k, p_k, \pi_k)$.

Once the QP solution $(\widehat{p}_k, \widehat{\pi}_k)$ has been determined, the major iteration proceeds by determining new variables (p_{k+1}, π_{k+1}) as

$$\begin{pmatrix} p_{k+1} \\ \pi_{k+1} \end{pmatrix} = \begin{pmatrix} p_k \\ \pi_k \end{pmatrix} + \alpha_k \begin{pmatrix} \widehat{p}_k - p_k \\ \widehat{\pi}_k - \pi_k \end{pmatrix},$$

where α_k is found from a line search that enforces a sufficient decrease in an augmented Lagrangian merit function (see Gill, Murray and Saunders [15]).

In this SQP formulation, the objective and constraint derivatives $\nabla\theta$ and J are required once each major iteration. They are needed to define the objective and constraints of the QP subproblem. The constraint derivatives have a structure determined by the multiple shooting scheme. For example, the Jacobian of the constraints (2.6) that impose the matching conditions is of the form

$$\begin{pmatrix} U_0 & -I & & & & & & & \\ & V_1 & U_1 & -I & & & & & \\ & & & V_2 & U_2 & -I & & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & & V_{N-1} & U_{N-1} & -I \end{pmatrix},$$

where $V_i = \partial s/\partial \bar{v}_i$ and $U_i = \partial s/\partial \bar{u}_i$. The structure of the derivatives for the inequality constraints $g \geq 0$ (2.7) will depend upon the particular application.

The QP algorithm is of reduced-gradient type, with the QP reduced Hessian being computed at the first feasible minor iterate. The QP solver must repeatedly solve linear systems formed from rows and columns of the structured derivatives. In the current version of SNOPT, these sparse systems are solved using the general-purpose sparse LU package LUSOL (see Gill *et al.* [16]). Current research is directed towards other factorization methods that more fully exploit the block-diagonal structure of the derivatives (see, e.g., Steinbach [31]).

SQP methods are most robust when the derivatives of the objective and constraint functions are computed exactly. As described in §4, the function and derivative computations involve computing the solution of a large-scale DAE system, and solution sensitivities with respect to the initial conditions and the control parameters. For problems associated with large-scale PDE systems, the derivatives require computing the sensitivity of solutions to the PDE at each spatial grid point with respect to initial conditions at every other spatial grid point.

The definition of the QP Hessian H_k is crucial to the success of an SQP method. In SNOPT, H_k is a positive-definite approximation to $G = \nabla^2_p \mathcal{L}(p_k, p_k, \pi_k)$, the Hessian of the modified Lagrangian. The exact Hessian is highly structured. For example, if there are no nonlinear constraints other than the matching conditions, $\nabla^2_p \mathcal{L}(p, p_k, \pi_k)$ has the form:

$$G = \begin{pmatrix} G_{00} & & & & & & & \\ & G_{11} & G_{21}^T & & & & & \\ & G_{21} & G_{22} & & & & & \\ & & & G_{33} & G_{43}^T & & & \\ & & & G_{43} & G_{44} & & & \\ & & & & & \ddots & & \\ & & & & & & G_{N-2,N-2} & G_{N-1,N-2}^T \\ & & & & & & G_{N-1,N-2} & G_{N-1,N-1} \end{pmatrix},$$

where the diagonal block 2×2 matrix involving G_{ii}, $G_{i+1,i}$ and $G_{i+1,i+1}$ represents the Hessian terms associated with variables $\bar{v}_i$ and $\bar{u}_i$. In SNOPT, H_k is a limited-memory quasi-Newton approximate Hessian. On completion of the line search, let the change in p and the gradient of the modified Lagrangian be

$$\delta_k = p_{k+1} - p_k \quad \text{and} \quad y_k = \nabla\mathcal{L}(p_{k+1}, p_k, \pi_{k+1}) - \nabla\mathcal{L}(p_k, p_k, \pi_{k+1}).$$

The approximate Hessian is updated using the BFGS quasi-Newton update,

$$H_{k+1} = H_k - \rho_k q_k q_k^T + \theta_k y_k y_k^T,$$

where $q_k = H_k \delta_k$, $\rho_k = 1/q_k^T \delta_k$ and $\theta_k = 1/y_k^T \delta_k$. If necessary, δ_k and y_k are redefined to ensure that H_{k+1} is positive definite (see Gill, Murray and Saunders [15] for more details).

The limited-memory scheme used in SNOPT is based on the observation that the SQP computation can be arranged so that the approximate Hessian H_k is only required to perform matrix-vector *products* of the form $H_k u$. This implies that H_k need not be stored explicitly, but may be regarded as an operator involving an initial

diagonal matrix H_r and a sum of rank-two matrices held implicitly in outer-product form. With this approach, a preassigned fixed number (say ℓ) of these updates are stored and products $H_k u$ are computed using $O(\ell)$ inner-products. For a discussion of limited-memory methods see, e.g., Gill and Murray [14], Nocedal [26]), Buckley and LeNir [8], and Gilbert and Lemaréchal [13].

Currently, SNOPT uses a simple limited-memory implementation of the BFGS quasi-Newton method. As the iterations proceed, the two vectors (q_k, y_k) defining the current update are added to an expanding list of most recent updates. When ℓ updates have been accumulated, the storage is "reset" by discarding all information accumulated so far. Let r and k denote the indices of two major iterations such that $r \leq k \leq r + \ell$ (i.e., iteration k is in the sequence of ℓ iterations following a reset at iteration r). During major iteration k, products of the form $H_k u$ are computed with work proportional to $k - r$:

$$H_k u = H_r u + \sum_{j=r}^{k-1} \rho_j (y_j^T u) y_j - \rho_j (q_j^T u) q_j,$$

where H_r is a positive-definite diagonal. On completion of iteration $k = r + \ell$, the diagonals of H_k are saved to form the new H_r (with $r = k + 1$).

4. DAE Sensitivity Analysis. Many engineering and scientific problems are described by systems of differential-algebraic equations (DAEs). Parametric sensitivity analysis of the (DAE) model yields information useful for parameter estimation, optimization, process sensitivity, model simplification and experimental design. Consequently, algorithms that perform such an analysis in an efficient and rapid manner are invaluable to researchers in many fields. In this section we present two such codes: DASSLSO and DASPKSO. The codes are modifications of the DAE solvers DASSL and DASPK ([7]). The DASPKSO code is used in DASOPT to compute the sensitivities of the solution to the DAE system. The algorithms used in these sensitivity codes have several novel features. They make use of an adaptive difference directional derivative approximation to (or alternatively a user supplied expression for) the sensitivity equations. The ability to adapt the increment as time progresses is important because the solution and sensitivities can sometimes change drastically. The sensitivity equations are solved simultaneously with the original system, yielding a nonlinear system at each time step. We will outline the algorithms here; further details on the algorithms, codes, theory and numerical results can be found in [24]. The new codes are easy to use, highly efficient, and well-suited for large-scale problems.

First, we briefly give some background on the algorithms in DASSL and DASPK. Further details can be found in [7]. DASSL is a code for solving initial-value DAE systems of the form

$$F(v, v', t) = 0, \quad v(0) = v_0.$$

The DAE system must be *index-one*. For *semi-explicit* DAE systems (ODEs coupled with nonlinear constraints) of the form

$$v_1' = f_1(v_1, v_2, t) \tag{4.1a}$$

$$0 = f_2(v_1, v_2, t), \tag{4.1b}$$

the system is index-one if $\partial f_2 / \partial v_2$ is nonsingular in a neighborhood of the solution. The initial conditions given to DASSL must always be *consistent*. For semi-explicit

DAE systems (4.1), this means that the initial conditions must satisfy the constraints (4.1b). Given a consistent set of initial conditions, DASSL solves the DAE over the given time interval via an implicit, adaptive-stepsize, variable-order numerical method. The dependent variables and their derivatives are discretized via backward differentiation formulas (BDF) of orders one through five. At each time step this yields a nonlinear system that is solved using a modified Newton iteration. The linear system at each Newton iteration is solved via either a dense or banded direct linear system solver, depending on the option selected by the user.

DASSL has been highly successful for solving a wide variety of small to moderate-sized DAE systems. For large-scale DAE systems such as those arising from PDEs in two or three dimensions, DASPK can be much more effective. DASPK uses the time-stepping methods of DASSL (and includes the DASSL algorithm as a user option). It solves the nonlinear system at each time step using an inexact Newton method. This means that the linear systems at each iteration are not necessarily solved exactly. In fact, they are solved approximately via a preconditioned GMRES iterative method. The user must provide a preconditioner, which is usually dependent on the class of problems being solved.

4.1. Sensitivity for DAEs—the basic approach. Consider the general DAE system with parameters,

$$F(v, v', p, t) = 0, \qquad v(0) = v_0,$$

where $v \in \mathbb{R}^{n_v}$, $p \in \mathbb{R}^{n_p}$. Here, n_v and n_p are the dimension and the number of parameters in the original DAE system, respectively. Sensitivity analysis entails finding the derivative of the above system with respect to each parameter. This produces an additional $n_s = n_p \times n_v$ sensitivity equations that, together with the original system, yield

$$\begin{aligned} F(v, v', p, t) &= 0 \\ \frac{\partial F}{\partial v} s_i + \frac{\partial F}{\partial v'} s_i' + \frac{\partial F}{\partial p_i} &= 0, \quad i = 1, 2, \ldots, n_p, \end{aligned} \tag{4.2}$$

where $s_i = dv/dp_i$ and $s_i' = dv'/dp_i$. Given the vector of combined unknowns $V = (\, v \quad s_1 \quad \cdots \quad s_{n_p} \,)^T$ and the vector-valued function

$$\mathbf{F} = \begin{pmatrix} F(v, v', p, t) \\ \dfrac{\partial F}{\partial v} s_1 + \dfrac{\partial F}{\partial v'} s_1' + \dfrac{\partial F}{\partial p_1} \\ \vdots \\ \dfrac{\partial F}{\partial v} s_{n_p} + \dfrac{\partial F}{\partial v'} s_{n_p}' + \dfrac{\partial F}{\partial p_{n_p}} \end{pmatrix},$$

the combined system can be rewritten as

$$\mathbf{F}(V, V', p, t) = 0, \quad V(0) = \begin{pmatrix} v_0 \\ \dfrac{dv_0}{dp_1} \\ \vdots \\ \dfrac{dv_0}{dp_{n_p}} \end{pmatrix}.$$

We note that the initial conditions for this DAE system must be chosen to be consistent, and that this implies that the initial conditions for the sensitivity equations must be consistent as well.

Approximating the solution to the combined system by a numerical method, for example the implicit Euler method with stepsize h, yields the nonlinear system

$$G(V_{n+1}) = \mathbf{F}(V_{n+1}, (V_{n+1} - V_n)/h, p, t_{n+1}) = 0. \tag{4.3}$$

Newton's method for the nonlinear system produces the iteration

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

where

$$\mathbf{J} = \begin{pmatrix} J & & & & \\ J_1 & J & & & \\ J_2 & 0 & J & & \\ \vdots & \vdots & \ddots & \ddots & \\ J_{n_p} & 0 & \dots & 0 & J \end{pmatrix}, \tag{4.4}$$

with

$$J = \frac{1}{h}\frac{\partial F}{\partial v'} + \frac{\partial F}{\partial v} \quad \text{and} \quad J_i = \frac{\partial J}{\partial v} s_i + \frac{\partial J}{\partial p_i}.$$

A number of codes for ODEs and DAEs solve the sensitivity system (4.2), or its special case for ODEs, directly (see [10]). If the partial derivative matrices are not available analytically, they are approximated by finite differences. The nonlinear system is usually solved by a so-called *staggered scheme*, in which the first block is solved for the state variables v via Newton's method, and then the block-diagonal linear system for the sensitivities s is solved at each time step.

4.2. Directional derivative sensitivity approximation. Although the direct solution of (4.2) is successful for many problems, in the context of DASSL/DASPK, there are three difficulties with this approach. First, for efficiency, DASSL was designed to use its approximation to the system Jacobian over as many time steps as possible. However, sensitivity implementations using the staggered scheme described above must re-evaluate the Jacobian at every step in order to ensure an accurate approximation to the sensitivity equations. Second, if the Jacobian has been approximated via finite differences, which is most often the case, large errors may be introduced into the sensitivities. Finally, in DASPK, the Jacobian matrices are never formed explicitly. Making use of the fact that the GMRES iterative method requires only products of the Jacobian matrix with a given vector, these matrix-vector products are approximated via a directional derivative difference approximation.

To eliminate these problems, we focus on approximating the sensitivity system (4.2) directly, rather than via the matrices $\partial F/\partial v$, $\partial F/\partial v'$, and $\partial F/\partial p$. In the simplest case, the user can specify directly the residual of the sensitivity system at the same time as the residual of the original system. Eventually, we intend to incorporate the automatic differentiation software ADIFOR [5] for this purpose. Alternatively, we can approximate the right-hand side of the sensitivity equations using a directional

derivative finite-difference approximation. As an example, define $s_i = dv/dp_i$ and solve

$$\begin{aligned} F(v, v', p, t) &= 0, \\ \frac{1}{\delta_i}(F(v + \delta_i s_i, v' + \delta_i s_i', p + \delta_i e_i, t) - F(v, v', p, t)) &= 0, \quad i = 1, 2, \ldots, n_p, \end{aligned}$$

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; the adaptive determination of the increment δ_i is discussed in greater detail by Maly and Petzold [24]. Approximations to the sensitivity equations are generated at the same time as the residual of the original system, via n_p additional calls to the user function routine. The resulting system is discretized by a numerical method (in DASSL/DASPK this is the BDF method of orders 1-5), yielding an iteration matrix of the form (4.4).

In general, for a Newton or Newton-Krylov iteration, one should be able to approximate the iteration matrix $\mathbf{J}$ by its block diagonal part provided that the error matrix for the Newton/modified Newton steps is nilpotent. To illustrate this idea, consider the problem formulation (4.3)

$$G(V) = 0$$

and apply a Newton step

$$V^{(k+1)} = V^{(k)} - \widehat{\mathbf{J}}^{-1} G(V^{(k)}), \tag{4.5}$$

where the Newton matrix $\mathbf{J}$ has been approximated by its block-diagonal part, $\widehat{\mathbf{J}}$. The true solution V^* satisfies

$$V^* = V^* - \widehat{\mathbf{J}}^{-1} G(V^*). \tag{4.6}$$

Subtracting (4.6) from (4.5) and defining $e^k = V^{(k+1)} - V^*$, the iteration errors satisfy

$$V^{(k+1)} - V^* = e^{k+1} \approx e^k - \widehat{\mathbf{J}}^{-1}\mathbf{J}e^k = (I - \widehat{\mathbf{J}}^{-1}\mathbf{J})e^k.$$

The error matrix has the form

$$I - \widehat{\mathbf{J}}^{-1}\mathbf{J} = \begin{pmatrix} 0 & & & & \\ J^{-1}J_1 & 0 & & & \\ J^{-1}J_2 & 0 & 0 & & \\ \vdots & \vdots & \ddots & \ddots & \\ J^{-1}J_{n_p} & 0 & \ldots & 0 & 0 \end{pmatrix}.$$

Maly and Petzold [24] show that because this matrix is nilpotent, the Newton iteration in DASSLSO achieves 2-step quadratic convergence for nonlinear problems. Using the block-diagonal part $\widehat{\mathbf{J}}$ as the preconditioner in the GMRES iteration in DASPKSO has resulted in excellent performance.

4.3. Sensitivity Analysis of Derived Quantities. In addition to the sensitivity analysis modifications to DASSL and DASPK, a stand alone routine (SENSD) has

been constructed that performs a sensitivity analysis of a derived quantity. This routine approximates the analytic sensitivity equations by finite differencing the derived quantity $Q(v, v', p, t)$ ($p \in I\!R^{n_p}$, $v \in I\!R^{n_v}$ and $Q \in I\!R^{n_q}$), using

$$\frac{dQ(v, v', p, t)}{dp_i} = \frac{\partial Q}{\partial v}\frac{dv}{dp_i} + \frac{\partial Q}{\partial v'}\frac{dv'}{dp_i} + \frac{\partial Q}{\partial p_i}.$$

Expanding $Q(v, v', p, t)$ in a Taylor's series about v gives

$$Q(v + \delta_i s_i, v' + \delta_i s_i', p + \delta_i e_i) = Q(t, v, v', p) + \delta_i \frac{\partial Q}{\partial v} s_i + \delta_i \frac{\partial Q}{\partial p_i} + \delta_i \frac{\partial Q}{\partial v'} s_i' + O(\delta_i^2),$$

so that

$$\frac{dQ(v, v', p, t)}{dp_i} \approx \frac{1}{\delta_i}\left(Q(v + \delta_i s_i, v' + \delta_i s_i', p + \delta_i e_i, t) - Q(v, v', p, t)\right).$$

This is one of many possible finite difference schemes that can be used. In the code, central differencing is also an option. The routine SENSD can be called after a successful return from a call to DASSLSO or DASPKSO and must be provided with a function (DRVQ) which defines the derived quantity Q.

5. Formulation of a PDE test problem. In order to test DASOPT on a realistic model problem, we formulated a boundary control heating problem in two spatial dimensions. This model problem is described by a nonlinear parabolic PDE. It is a two-dimensional generalization of the model problem described in [22]. A rectangular domain in space is heated by controlling the temperature on its boundaries. It is desired that the transient temperature in a specified interior subdomain follow a prescribed temperature-time trajectory as closely as possible. The domain Ω is given by

$$\Omega = \{(x, y) \mid 0 \le x \le x_{\max},\ 0 \le y \le y_{\max}\},$$

and the control boundaries are given by

$$\partial\Omega_1 = \{(x, y) \mid y = 0\}, \quad \text{and} \quad \partial\Omega_2 = \{(x, y) \mid x = 0\}.$$

The temperature distribution in Ω, as a function of time, is controlled by the energy input across the boundaries $\partial\Omega_1$ and $\partial\Omega_2$, as discussed below. The other two boundaries ($x = x_{\max}$ and $y = y_{\max}$) are assumed to be insulated, so that no energy flows into or out of Ω along the normals to these boundaries. The temperature must be controlled in the subdomain

$$\Omega_c = \{(x, y) \mid x_c \le x \le x_{\max},\ y_c \le y \le y_{\max}\}.$$

This is illustrated in Fig. 5.1.

The control problem is to be solved for the time interval $t \in [0, t_{\max}]$. The temperature $T = T(x, y, t)$ is then determined by the nonlinear parabolic PDE given below, for $(x, y, t) \in \Omega \times [0, t_{\max}]$.

The temperature T is controlled by heat sources located on the boundaries $\partial\Omega_1$ and $\partial\Omega_2$. These heat sources are represented by control functions $u_1(x, t)$ on $\partial\Omega_1$, and $u_2(y, t)$ on $\partial\Omega_2$. The control functions are to be determined. The objective is to control the temperature-time trajectory on the subdomain Ω_c. A target trajectory

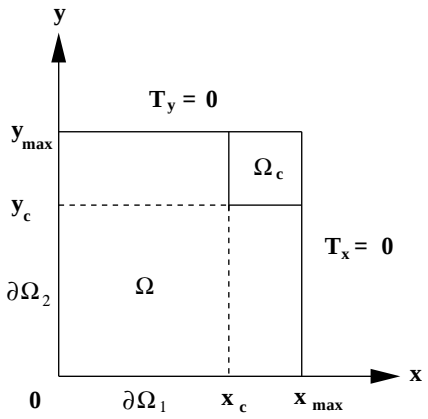

FIG. 5.1. *Two dimensional spatial domain for the parabolic control test problem.*

$\tau(t)$, $t \in [0, t_{\max}]$, is specified. The actual temperature in Ω_c should approximate $\tau(t)$ as closely as possible.

We measure the difference between $T(x, y, t)$ and $\tau(t)$ on Ω_c by the function

$$\phi(u) = \int_0^{t_{\max}} \int_{y_c}^{y_{\max}} \int_{x_c}^{x_{\max}} w(x, y, t)[T(x, y, t) - \tau(t)]^2 \, dx \, dy \, dt, \tag{5.1}$$

where $w(x, y, t) \geq 0$ is a specified weighting function. The control functions u_1 and u_2 are determined so as to

$$\underset{u}{\text{minimize}} \quad \phi(u), \tag{5.2}$$

subject to $T(x, y, t)$ satisfying the PDE and other constraints.

The temperature $T(x, y, t)$ must satisfy the following PDE, boundary conditions, and bounds

$$\begin{aligned}
\alpha(T)[T_{xx} + T_{yy}] + S(T) &= T_t, & (x, y, t) &\in \Omega \times [0, t_{\max}] \\
T(x, 0, t) - \lambda T_y &= u_1(x, t), & x &\in \partial\Omega_1 \\
T(0, y, t) - \lambda T_x &= u_2(y, t), & y &\in \partial\Omega_2 \\
T_x(x_{\max}, y, t) &= 0, \\
T_y(x, y_{\max}, t) &= 0, \\
0 \leq T(x, y, t) &\leq T_{\max}.
\end{aligned} \tag{5.3}$$

The controls u_1 and u_2 are also required to satisfy the bounds

$$0 \leq u_1, u_2 \leq u_{\max}.$$

The initial temperature distribution $T(x, y, 0)$ is a specified function. The coefficient $\alpha(T) = \lambda / c(T)$, where λ is the heat conduction coefficient and $c(T)$ is the heat capacity. The source term $S(T)$ represents internal heat generation, and is given by

$$S(T) = S_{\max} e^{-\beta_1/(\beta_2 + T)}$$

where $S_{\max}$, β_1, $\beta_2 \geq 0$ are specified nonnegative constants.

The numerical solution has been obtained by constructing finite-difference grids in space, and solving the resulting ODEs by the multiple-shooting method as described below.

A uniform rectangular grid is constructed on the domain Ω

$$\begin{aligned} x_i &= i\Delta x, \quad i = 0, 1, \ldots, m, \qquad \Delta x = x_{\max}/m \\ y_j &= j\Delta y, \quad j = 0, 1, \ldots, n, \qquad \Delta y = y_{\max}/n. \end{aligned}$$

Then let

$$\begin{aligned} T_{ij}(t) &= T(x_i, y_j, t), \qquad u_{1i}(t) = u_1(x_i, t), \qquad \alpha_{ij}(t) = \alpha(T_{ij}(t)), \\ S_{ij}(t) &= S(T_{ij}(t)), \qquad u_{2j}(t) = u_2(y_j, t). \end{aligned}$$

The PDE is then approximated in the interior of Ω by the following system of $(m-1)(n-1)$ ODEs

$$\begin{aligned} \frac{dT_{ij}}{dt} = \frac{\alpha_{ij}}{\Delta x^2}[T_{i-1,j} - 2T_{ij} + T_{i+1,j}] \\ + \frac{\alpha_{ij}}{\Delta y^2}[T_{i,j-1} - 2T_{ij} + T_{i,j+1}] + S_{ij}, \end{aligned} \tag{5.4}$$

for $i = 1, 2, \ldots, m-1$, $j = 1, 2, \ldots, n-1$. Each of the $2(m+n)$ boundary points also satisfies a differential equation similar to (5.4). These will include values outside Ω, which are eliminated by using the boundary conditions. Specifically, we use

$$\begin{aligned} T_{i,n+1} &= T_{i,n-1}, \quad i = 0, 1, \ldots, m \\ T_{m+1,j} &= T_{m-1,j} \quad j = 0, 1, \ldots, n, \end{aligned}$$

to approximate the conditions $T_y = 0$ and $T_x = 0$.

The finite-difference approximations to the boundary conditions on $\partial\Omega_1$ and $\partial\Omega_2$ are given by

$$T_{i0} - \frac{\lambda}{2\Delta y}(T_{i1} - T_{i,-1}) = u_{1i}, \quad i = 0, 1, \ldots, m \tag{5.5a}$$

$$T_{0j} - \frac{\lambda}{2\Delta x}(T_{1j} - T_{-1,j}) = u_{2j}, \quad j = 0, 1, \ldots, n \tag{5.5b}$$

These relations are used to eliminate the values $T_{i,-1}$ and $T_{-1,j}$ from the differential equations (as in (5.4)), for the functions T_{ij} on $\partial\Omega_1$ and $\partial\Omega_2$. As a result, the control functions u_{1i} and u_{2j} are explicitly included in these differential equations, giving $2(m+n)$ additional differential equations. Together with the $(m-1)(n-1)$ ODEs given by (5.4), this gives a total of $(m+1)(n+1)$ ODEs for the same number of unknown functions $T_{ij}(t)$. To simplify the notation in what follows, this system of $(m+1)(n+1)$ ODEs will be represented by

$$\frac{dv(t)}{dt} = f(v, u(t), t), \quad v(0) = v_0, \tag{5.6}$$

where v_0 represents the initial value of $v(t)$, and $u = u(t)$ the control functions. The vector function $u(t)$ has elements $u_{1i}(t)$, $i = 0, 1, \ldots, m$, and $u_{2j}(t)$, $j = 0, 1, \ldots, n$. These ODEs correspond to those given by (1.1a).

As discussed earlier the multiple shooting method is applied by dividing the total time interval $[0, t_{\max}]$ into N equal lengths Δt, with $N\Delta t = t_{\max}$. Also let $t_k = k\Delta t$,

$k = 0, 1, \ldots, N$. The system of ODEs (5.6) on $[0, t_{\max}]$ is now considered as N independent systems, each on its own time subinterval $[t_k, t_{k+1}]$. Let $v_k(t)$ represent $v(t)$ and $u_k(t)$ represent $u(t)$ on $[t_k, t_{k+1}]$, and $\bar{v}_k$ be the initial value of $v_k(t)$. Then $v_k(t)$ must satisfy

$$\frac{dv_k}{dt} = f(v_k, u_k(t), t), \quad v_k(t_k) = \bar{v}_k, \quad k = 0, 1, \ldots, N-1.$$

The value of $\bar{v}_0 = v_0$, while the remaining initial values $\bar{v}_k$, $k = 1, 2, \ldots, N-1$, are determined by continuity conditions (2.6) in the optimization problem. This is illustrated in Fig. 5.2.

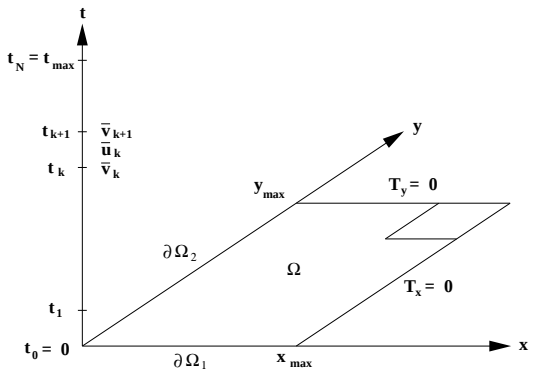

FIG. 5.2. *Space-time domain for test problem showing the shooting intervals.*

For each subinterval, the control vector $u_k(t)$ is approximated as in (2.3), with the parameters $\bar{u}_k$ being determined by the optimization. Bounds on the $u_k(t)$ at $t = t_k$ (and any additional points) give linear inequalities on the $\bar{u}_{kl}$. Since $u_k(t)$ is given in terms of the control parameters $\bar{u}_k$, it is clear that $v_k(t_{k+1})$ is a function of $\bar{v}_k$ and $\bar{u}_k$. This dependence has been explicitly given earlier in (2.5).

Equations (2.6) represent the $N(m+1)(n+1)$ individual equality constraints that must be satisfied. The optimization code SNOPT requires the Jacobian of these constraints with respect to the parameters $\bar{v}_k$ and $\bar{u}_k$. These partial derivatives can be obtained using the sensitivity capability of DASPKSO. The sensitivity of each element of $s(\bar{v}_k, \bar{u}_k)$ with respect to each element of $\bar{v}_k$ and $\bar{u}_k$ must be computed. As s, $\bar{v}_k \in \mathbb{R}^{n_v}$ and $\bar{u}_k \in \mathbb{R}^{n_{\bar{u}}}$, this requires that for each subinterval, $n_v(n_v + n_{\bar{u}})$ sensitivity calculations are required. Thus a total of $Nn_v(n_v + n_{\bar{u}})$ such calculations must be made to estimate the Jacobian. In order to reduce this computation to a reasonable size, other approaches are needed, and they are being investigated.

The objective function is computed by adding the single ODE (1.2) to the system (5.6). The gradient of the objective function is then obtained as part of the sensitivity computation.

The state bounds on the $T_{ij}(t_k)$ are imposed at each discrete time t_k by the simple bounds

$$0 \le \bar{v}_k \le T_{\max} e, \quad k = 1, 2, \ldots, N. \tag{5.7}$$

These will enforce the bounds at the points t_k, but there may be some small violation at intermediate time points.

The optimization problem to be solved can now be stated as follows: minimize the spatial discretization of (5.1) subject to the linear equality constraints (2.4), the bound constraints (5.7), and the nonlinear equality constraints (2.6).

The nonlinear parabolic PDE boundary control problem described by (5.1), (5.2) and (5.3) has been solved computationally using the discrete approximation described above. Numerical results for ten cases, including cases with the nonlinear source term and bounds on the interior temperatures, are summarized in the next section.

6. Computational results with DASOPT. The purpose of the computations summarized in this section was to test the DASOPT code on the relatively simple 2D nonlinear parabolic PDE problem described in the previous section. This test problem has the property that the size of the optimization problem can be easily increased by simply using a finer spatial grid. This readily permits the dependence of solution time on problem size to be observed.

It was also important to determine if the combination of DASPKSO and SNOPT would result in a convergent algorithm for this type of problem. As shown in the examples below, convergence to an optimal control was typically obtained in no more than 17 major iterations of SNOPT. While this parabolic PDE can be solved using single shooting, we used multiple shooting in order to test the performance of the combined system.

This type of problem also permitted testing the capability to impose inequality constraints on the state variables, in this case bounds on the interior temperatures. This ability is clearly shown by comparing the control and temperatures obtained with and without bounds on the maximum permitted interior temperatures.

The computational results obtained with DASOPT, using the CRAY C90, on the optimal control 2D nonlinear PDE will now be summarized. The rectangular domain (see Fig. 5.1) is chosen as $\Omega = \{(x, y) \mid 0 \le x \le 0.8,\ 0 \le y \le 1.6\}$. The time integration interval is $[0, 2]$ and the goal is to follow as closely as possible a specified time-temperature trajectory $\tau(t)$ (as specified in all following figures) in the subdomain $\Omega_c = \{(x, y) \mid 0.6 \le x \le 0.8,\ 1.2 \le y \le 1.6\}$. We want to determine the boundary control so as to minimize the objective (5.1) with $w(x, y, t) = 0$ for $t \in [0, 0.2]$ and $w(x, y, t) = 1$ for $t \in [0.2, 2]$. On the boundaries $\partial\Omega_1$ and $\partial\Omega_2$ the controls $u_1(x, t)$ and $u_2(y, t)$ are given by a control function $u(t)$ as follows:

$$
\begin{aligned}
u_1(x,t) &= \begin{cases} u(t) & 0 \le x \le 0.2; \\ \left(1 - \dfrac{x - 0.2}{1.2}\right) u(t) & 0.2 \le x \le 0.8. \end{cases} \\
u_2(x,t) &= \begin{cases} u(t) & 0 \le y \le 0.4; \\ \left(1 - \dfrac{y - 0.4}{2.4}\right) u(t) & 0.4 \le y \le 1.6. \end{cases}
\end{aligned}
\tag{6.1}
$$

Note that for any fixed t, u is constant on the boundary $\partial\Omega_1$ for $0 \le x \le 0.2$, and then decreases linearly to $u/2$ at $x = 0.8$. The control u_2 on $\partial\Omega_2$ is similar. We also impose the initial condition $u(0) = 0$.

For the multiple shooting, the time integration interval is divided into ten shooting intervals of equal length 0.2. We maintain the lower bound of zero on the temperature at each shooting point. Each shooting interval is actually divided into two control subintervals (explaining the presence of an additional index j) where the control function $u(t)$ is represented by a quadratic polynomial

$$u_{kj}(t) = \bar{u}_{kj0} + \bar{u}_{kj1}(t - t_{kj}) + \bar{u}_{kj2}(t - t_{kj})^2. \tag{6.2}$$

We enforce continuity in time at the extremities of each control subinterval among all $u_{kj}(t)$ and their derivative $u'_{kj}(t)$. We also impose the following bounds on the control parameters

$$|\bar{u}_{kj1}| \leq 5, \qquad |\bar{u}_{kj2}| \leq 7.$$

We maintain an upper bound on the maximal value of the control $u_{\max} = 1.1$ and, except in one case, a lower bound of zero at the extremities and in the middle of each control subinterval.

In all ten test cases presented here, the PDE parameters λ, c and α were assumed to be constant, with the values $\lambda = c = \frac{1}{2}$, and $\alpha = 1$. Therefore, the PDE is linear when $S_{\max} = 0$. The parameters in $S(T)$ were chosen as $\beta_1 = 0.2$ and $\beta_2 = 0.05$. In addition to the linear case $S_{\max} = 0$, the values of $S_{\max} = 0.5$, 1.0, were used to show the significant effect of the nonlinear heat source term. At $t = 0$, the initial temperature $T_{ij}(0) = 0$ was used for all cases.

The effect of the state variable bounds is shown by requiring that the temperatures at every space-time grid-point satisfy $T_{ij}(t_k) \leq T_{\max}$. This upper bound was imposed in three of the ten cases. A lower bound of zero was also imposed for all ten cases, but was only active in Case 9.

The computational results obtained for the ten cases are summarized in Table 6.1. The time dependent optimal solution for each of the ten cases is presented in Figs. 6.1–6.10. The figure number corresponds to the case number in Table 6.1, so that Fig. 6.x shows results for Case x.

TABLE 6.1
Summary of test problem optimal solutions.

Case	Grid Size	$S_{\max}$	$T_{\max}$ Bound	Initial Values	ϕ ($\times 10^5$)	Major Itns	Time (Secs)	Time /Itn
1	5×5	0.0	None	0	1.525	17	176	10.4
2	5×9	0.0	None	0	1.517	16	488	30.5
3	5×17	0.0	None	0	1.515	16	1584	99.0
4	9×17	0.0	None	0	1.536	11	3489	317.2
5	5×9	0.5	None	#2	1.836	16	432	27.0
6	5×9	1.0	None	#5	15.92	7	208	29.7
7	5×9	0.0	0.7	0	5.754	9	285	31.7
8	5×9	0.5	0.7	#7	2.490	7	224	32.0
9	5×9	1.0	0.7	#5	4.277	6	204	34.0
10	5×9	0.0	None	0	0.826	17	545	32.1

In Table 6.1, the grid size describes the discrete grid on the spatial domain Ω. For example, the 5×5 grid gives $\Delta x = 0.2$, $\Delta y = 0.4$, and defines T_{ij}, for $i, j = 0$, 1, 2, 3, 4. Thus for an $m \times n$ grid, there are mn spatial grid points, including boundary grid points.

The column "$S_{\max}$" shows the degree of nonlinearity of the problem, where $S_{\max} = 0$ implies that the problem is linear. The column "$T_{\max}$ Bound" shows when a state upper bound is imposed. The column "Initial Values" gives the initial estimates used for the $T_{ij}(t_k)$ and the $\bar{u}_{kj}$ control coefficients. The value zero assumes no knowledge of the optimal solution and gives the most difficult optimization problem. Much better estimates can be obtained from the optimal solution with a coarser grid, or a lower value of $S_{\max}$. A nonzero entry indicates that the optimal $T_{ij}(t_k)$ and $\bar{u}_{kj}$

from a previous case were used as initial estimates. The value of the entry gives the particular case used.

The SNOPT default parameter settings were used throughout, except for the optimality tolerance, which was set to 10^{-5}. Roughly speaking, these settings give an approximate minimizer with a reduced-gradient norm less than 10^{-5} and a maximum nonlinear constraint violation less than 10^{-6} (for further details of the termination criteria, see [25]). The default maximum number of limited memory updates stored (the number "ℓ" of §3) is 20.

The last four columns in Table 6.1 give the results of the computation. The minimum value of the objective function ϕ, scaled by 10^5, is shown for each case. The number of major iterations required by SNOPT, the CRAY C90 cpu time (in seconds), and the average time per iteration are given in the last three columns.

Considerably more information on the optimal solution to each case is presented in Figs. 6.1–6.10. These ten figures show the optimal control and selected temperatures as a function of time. The dotted line shows the control $u(t)$. The solid line (identical for all cases) shows the desired temperature-time trajectory $\tau(t)$ on the subdomain Ω_c. The dashed line shows the temperature $T_{00}(t)$ at the boundary grid point $x = y = 0$. Finally, the dash-dot lines show the temperatures at each of the grid points in the subdomain Ω_c.

We now comment briefly on these computational results. First, we observe that DASOPT determines the optimal control (to within the specified tolerances) with very few SQP major iterations. As shown in Table 6.1, no more than 17 iterations were needed for any one of the ten cases. A grand total of 132 objective and constraint evaluations and 122 major iterations were required to solve the ten cases. It follows that, on average, SNOPT required slightly more than one function evaluation per iteration. This favorable performance is due primarily to the use of the SQP method in SNOPT. The ten figures show clearly how the optimal control is able to minimize the difference between the solid line $\tau(t)$ and the temperature in Ω_c, as given by the dash-dot lines. This difference is measured by the objective function $\phi(u)$. Of course, it is not possible for any boundary temperature control to obtain an interior temperature in Ω_c that exactly follows the desired temperature $\tau(t)$. This is because of the time delay and smoothing effect of the heat equation. Therefore, the actual optimal value of ϕ is positive in all cases considered and depends primarily on the extent to which the problem is constrained, and the degree of nonlinearity. This explains why the temperature profiles in Ω_c shown in Figs. 6.1–6.10 do not match exactly. However, we have observed a good agreement in the optimal values of ϕ found by DASOPT when using different grid-sizes, see, e.g., the optimal values of ϕ obtained in Cases 1–4. The value of ϕ obtained in each of the ten cases is at least a local minimum, as determined by the termination test in SNOPT. The smallest objective (Case 10, with $\phi = 0.826 \times 10^{-5}$) corresponds to the least constrained linear problem. For comparison, the value of ϕ with $u(t) = 0$ and $T_{ij}(t) = 0$, is $\phi = 1674.7 \times 10^{-5}$, so that the objective function for Case 10 is reduced by a factor of approximately 2000 by the optimization. The largest value (Case 6), is highly nonlinear and is constrained by the requirement that $u(t) \geq 0$. This constraint is removed for Case 9, and it is seen that the value of ϕ is reduced by more than a factor of three. The smallest value of ϕ is obtained (Case 10) because the control is piecewise linear, with a derivative discontinuity permitted at the ends of each time subinterval. All other cases satisfy the continuity constraints (2.4) on both the control function and its derivative. Another illustration of the effect of additional state constraints is given by comparing Cases 2

and 7. The only difference between them is that the value $T_{\max} = 0.7$ is enforced in Case 7.

The significant effect of the nonlinear heat generation term is shown when $S_{\max}$ is 0.5 and 1.0. Matching the desired trajectory is more difficult with increased interior heat generation. This is illustrated most clearly in Fig. 6.6, where the constraint $u(t) \geq 0$ substantially reduces the ability to remove internally generated heat for $t \geq 1.2$.

Finally, we discuss the effect of the grid size on the accuracy of the solution to the PDE, and the computation time required. The accuracy of the approximate solution to the PDE increases as the spatial mesh size $(\Delta x, \Delta y)$ decreases, that is, as the number mn of spatial grid points increases. This increased accuracy is, however, obtained at the cost of a substantial increase in computing time. The primary cause of this increase is the sensitivity calculation needed to obtain the gradient of the objective function and the Jacobian of the nonlinear constraints (2.6) that enforce the matching conditions. In the implementation used for these tests, the time needed for the sensitivity calculation is proportional to $(mn)^2$. Therefore it increases by a factor of approximately 36 in going from a 5×5 grid (Case 1) to a 9×17 grid (Case 4). The actual increase in time per iteration is approximately a factor of 31, so that the sensitivity calculation requires over 90% of the total computing time.

Computing the time-temperature curves in Ω_c shown in Figs. 6.1–6.4, we observe that they are essentially unchanged as the grid size decreases. Furthermore, the optimal control $u(t)$ for each of these cases is almost identical. This indicates that the coarse grid (Fig. 6.1) gives a good approximation to the optimal control and temperatures for the more accurate finer grid (Fig. 6.4). This permits the use of a multigrid method, where the optimal solution of a coarse grid is used as the initial values for a finer grid solution. This was tested on these cases, with the result that only three or four major iterations were needed to get to the optimal solution with the finer grid. For more difficult problems there will be larger changes in the optimal solution for the finer grid, but this multigrid technique should still be very useful in reducing the total computing effort.

Closing remarks. The codes DASSLSO, DASPKSO and SENSD, as well as the driver routines for the test problems in [24], are available via anonymous FTP from ftp.cs.umn.edu, in the /users/tmaly directory.

Acknowledgments. We are grateful to two referees for their helpful and constructive comments.

REFERENCES

[1] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Classics in Applied Mathematics, 13, Society for Industrial and Applied Mathematics (SIAM) Publications, Philadelphia, PA, 1995. ISBN 0-89871-354-4.

[2] J. T. Betts, *Issues in the direct transcription of optimal control problems to sparse nonlinear programs*, in Control Applications of Optimization, R. Bulirsch and D. Kraft, eds., vol. 115 of Internat. Ser. Numer. Math., Basel, 1994, Birkhäuser, pp. 3–17.

[3] J. T. Betts and W. P. Huffman, *The application of sparse nonlinear programming to trajectory optimization*, J. of Guidance, Control, and Dynamics, 14 (1991), pp. 338–348.

[4] L. T. Biegler, J. Nocedal, and C. Schmid, *A reduced Hessian method for large-scale constrained optimization*, SIAM J. Optim., 5 (1995), pp. 314–347.

[5] C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland, *ADIFOR—generating derivative codes from Fortran programs*, Scientific Programming, 1 (1992), pp. 11–29.

[6] K. E. Brenan, *Differential-algebraic equations issues in the direct transcription of path constrained optimal control problems*, Ann. Numer. Math., 1 (1994), pp. 247–263.

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

[8] A. Buckley and A. LeNir, *QN-like variable storage conjugate gradients*, Math. Prog., 27 (1983), pp. 155–175.

[9] R. Bulirsch, E. Nerz, H. J. Pesch, and O. von Stryk, *Combining direct and indirect methods in optimal control: Range maximization of a hang glider*, in Calculus of Variations, Optimal Control Theory and Numerical Methods., R. Bulirsch, A. Miele, J. Stoer, and K. H. Well, eds., vol. 111 of Internat. Ser. Numer. Math., Basel, 1993, Birkhäuser, pp. 273–288.

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

[11] E. D. Dickmanns and K. H. Well, *Approximate solution of optimal control problems using third-order Hermite polynomial functions*, in Proc. 6th Technical Conference on Optimization Techniques, Berlin, Heidelberg, New York, London, Paris and Tokyo, 1975, Springer Verlag.

[12] S. K. Eldersveld, *Large-scale sequential quadratic programming algorithms*, PhD thesis, Department of Operations Research, Stanford University, Stanford, CA, 1991.

[13] J. C. Gilbert and C. Lemaréchal, *Some numerical experiments with variable-storage quasi-Newton algorithms*, Math. Prog., (1989), pp. 407–435.

[14] P. E. Gill and W. Murray, *Conjugate-gradient methods for large-scale nonlinear optimization*, Report SOL 79-15, Department of Operations Research, Stanford University, Stanford, CA, 1979.

[15] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, Numerical Analysis Report 96-2, Department of Mathematics, University of California, San Diego, La Jolla, CA, 1996.

[16] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *Maintaining LU factors of a general sparse matrix*, Linear Algebra and its Applications, 88/89 (1987), pp. 239–270.

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

[18] D. M. Gritsis, C. C. Pantelides, and R. W. H. Sargent, *Optimal control of systems described by index two differential-algebraic equations*, SIAM J. Sci. Comput., 16 (1995), pp. 1349–1366.

[19] C. R. Hargraves and S. W. Paris, *Direct trajectory optimization using nonlinear programming and collocation*, J. of Guidance, Control, and Dynamics, 10 (1987), pp. 338–348.

[20] D. Kraft, *On converting optimal control problems into nonlinear programming problems*, in Computational Mathematical Programming, K. Schittkowski, ed., NATO ASI Series F: Computer and Systems Sciences 15, Springer Verlag, Berlin and Heidelberg, 1985, pp. 261–280.

[21] R. Lamour, *A well posed shooting method for transferable DAEs*, Numer. Math., 59 (1991), pp. 815–829.

[22] F. Leibfritz and E. W. Sachs, *Numerical solution of parabolic state constrained control problems using SQP and interior point methods*, in Large Scale Optimization: the State of the Art, W. W. Hager, D. W. Hearn, and P. M. Pardalos, eds., Kluwer Academic Publishers, Dordrecht, 1994, pp. 245–258. ISBN 0-7923-2798-5.

[23] G. Leitmann, *The Calculus of Variations and Optimal Control*, Mathematical Concepts and Methods in Science and Engineering, 24, Plenum Press, New York and London, 1981. ISBN 0-306-40707-8.

[24] T. Maly and L. R. Petzold, *Numerical methods and software for sensitivity analysis of differential-algebraic systems.* to appear in Applied Numerical Mathematics, 1996.

[25] B. A. Murtagh and M. A. Saunders, *MINOS 5.4 User's Guide*, Report SOL 83-20R, Department of Operations Research, Stanford University, Stanford, CA, 1993.

[26] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Math. Comput., 35 (1980), pp. 773–782.

[27] C. C. Pantelides, R. W. H. Sargent, and V. S. Vassiliadis, *Optimal control of multistage systems described by high-index differential-algebraic equations*, Internat. Ser. Numer. Math., 115 (1994), pp. 177–191.

[28] H. J. Pesch, *Real-time computation of feedback controls for constrained optimal control problems*, Optimal Control Appl. Methods, 10 (1989), pp. 129–171.

[29] V. H. Schultz, *Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots*, PhD thesis, University of Heidelberg, 1996.

[30] V. H. Schulz, H.-G. Bock, and M. C. Steinbach, *Exploiting invariants in the numerical solution of multipoint boundary value problems for DAE.* submitted to SIAM J. Sci. Comput., 1996.

[31] M. C. Steinbach, *A structured interior point SQP method for nonlinear optimal control problems*, in Control Applications of Optimization, R. Bulirsch and D. Kraft, eds., vol. 115 of Internat. Ser. Numer. Math., Basel, 1994, Birkhäuser, pp. 213–222.

[32] P. Tanartkit and L. T. Biegler, *Stable decomposition for dynamic optimization*, Ind. Eng. Chem. Res., (1995), pp. 1253–1266.

[33] I.-B. Tjoa and L. T. Biegler, *A reduced SQP strategy for errors-in-variables estimation*, Comput. Chem. Eng., 16 (1992), pp. 523–533.

[34] O. von Stryk, *Numerical solution of optimal control problems by direct collocation*, in Control Applications of Optimization, R. Bulirsch, A. Miele, J. Stoer, and K. H. Well, eds., vol. 111 of Internat. Ser. Numer. Math., Basel, 1993, Birkhäuser, pp. 129–143.

[35] O. von Stryk and R. Bulirsch, *Direct and indirect methods for trajectory optimization*, Ann. Oper. Res., 37 (1992), pp. 357–373.

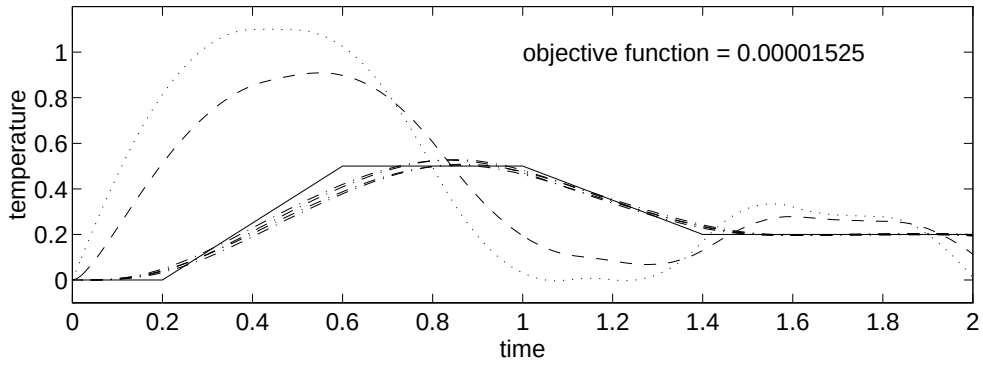

FIG. 6.1. *Optimal solution computed by DASOPT on a* 5×5 *grid. Solid line:* $\tau(t)$*. Dotted line:* $u(t)$*. Dashed line:* $T_{00}(t)$*. Dash-dot lines:* $T_{ij}(t)$ *in* Ω_c*.*

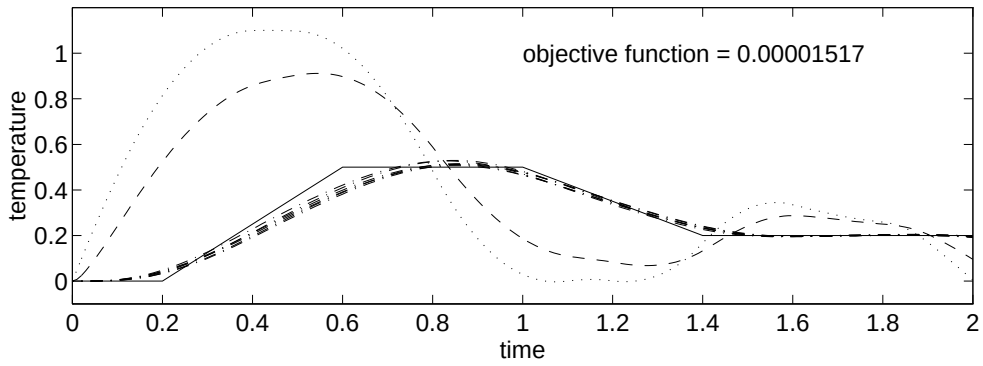

FIG. 6.2. *Optimal solution computed by DASOPT on a* 5×9 *grid. Solid line:* $\tau(t)$*. Dotted line:* $u(t)$*. Dashed line:* $T_{00}(t)$*. Dash-dot lines:* $T_{ij}(t)$ *in* Ω_c*.*

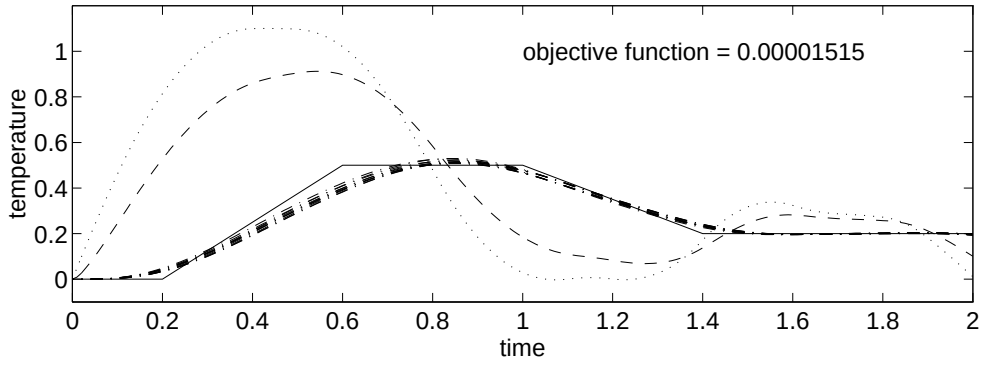

FIG. 6.3. *Optimal solution computed by DASOPT on a* 5×17 *grid. Solid line:* $\tau(t)$*. Dotted line:* $u(t)$*. Dashed line:* $T_{00}(t)$*. Dash-dot lines:* $T_{ij}(t)$ *in* Ω_c*.*

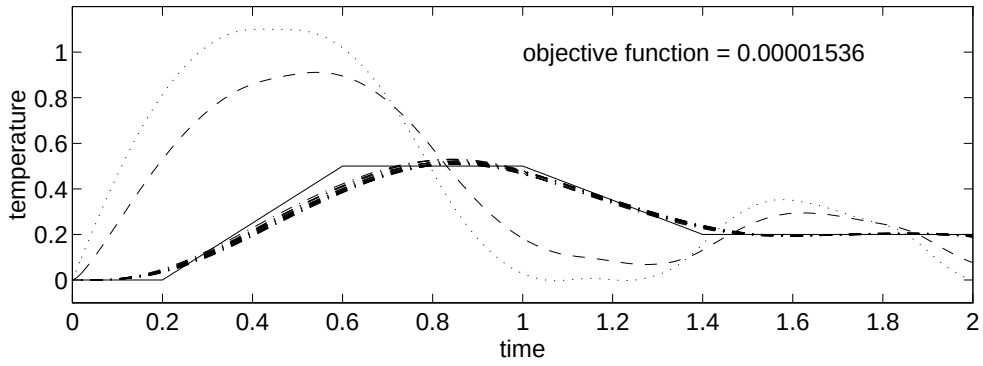

FIG. 6.4. *Optimal solution computed by* $DASOPT$ *on a* 9×17 *grid. Solid line:* $\tau(t)$. *Dotted line:* $u(t)$. *Dashed line:* $T_{00}(t)$. *Dash-dot lines:* $T_{ij}(t)$ *in* Ω_c.

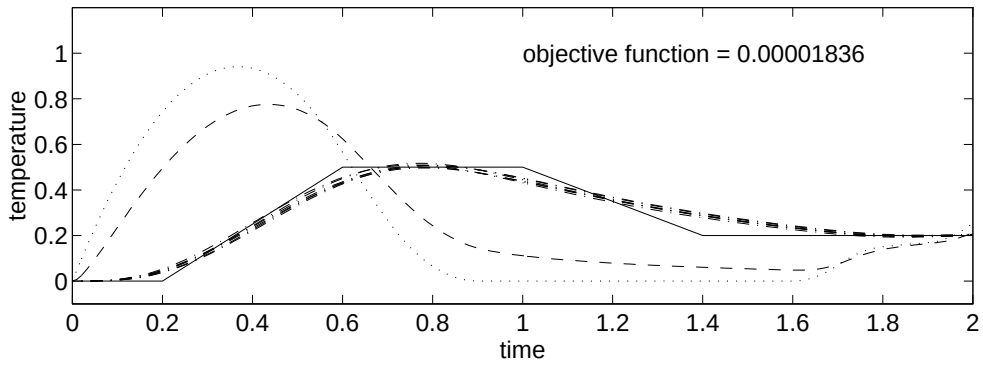

FIG. 6.5. *Optimal solution computed by* $DASOPT$ *on a* 5×9 *grid with* $S_{\max} = 0.5$. *Solid line:* $\tau(t)$. *Dotted line:* $u(t)$. *Dashed line:* $T_{00}(t)$. *Dash-dot lines:* $T_{ij}(t)$ *in* Ω_c.

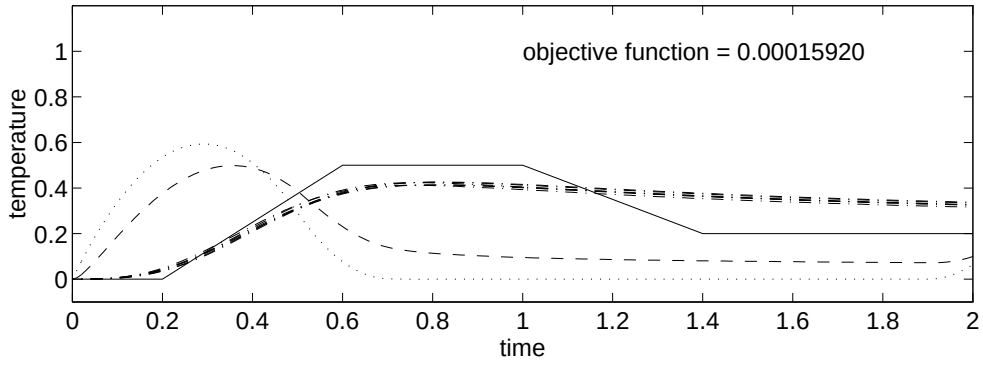

FIG. 6.6. *Optimal solution computed by* $DASOPT$ *on a* 5×9 *grid with* $S_{\max} = 1$. *Solid line:* $\tau(t)$. *Dotted line:* $u(t)$. *Dashed line:* $T_{00}(t)$. *Dash-dot lines:* $T_{ij}(t)$ *in* Ω_c.

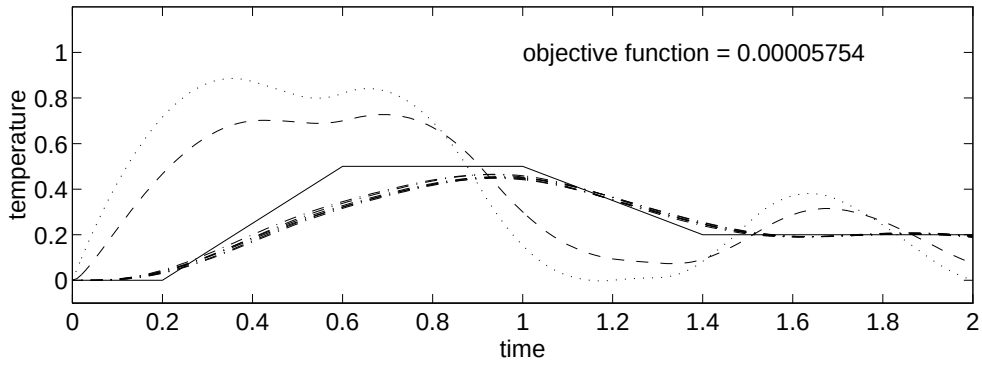

FIG. 6.7. *Optimal solution computed by DASOPT on a* 5×9 *grid with* $T_{ij}(t) \leq 0.7$. *Solid line:* $\tau(t)$. *Dotted line:* $u(t)$. *Dashed line:* $T_{00}(t)$. *Dash-dot lines:* $T_{ij}(t)$ *in* Ω_c.

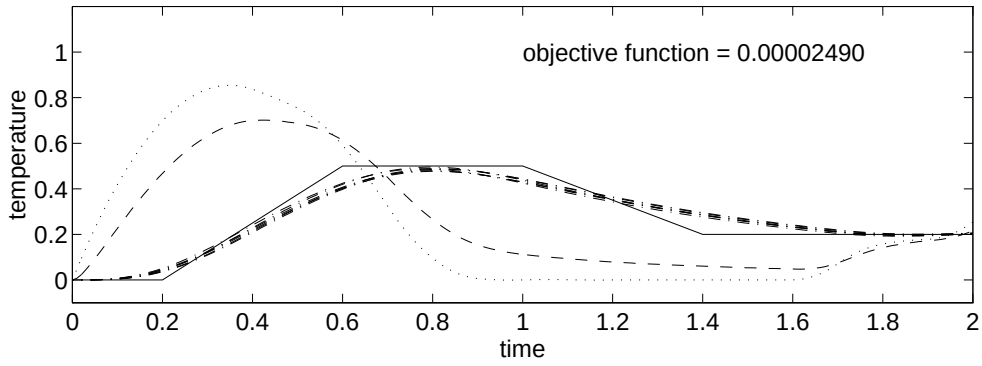

FIG. 6.8. *Optimal solution computed by DASOPT on a* 5×9 *grid with* $T_{ij}(t) \leq 0.7$ *and* $S_{\max} = 0.5$. *Solid line:* $\tau(t)$. *Dotted line:* $u(t)$. *Dashed line:* $T_{00}(t)$. *Dash-dot lines:* $T_{ij}(t)$ *in* Ω_c.

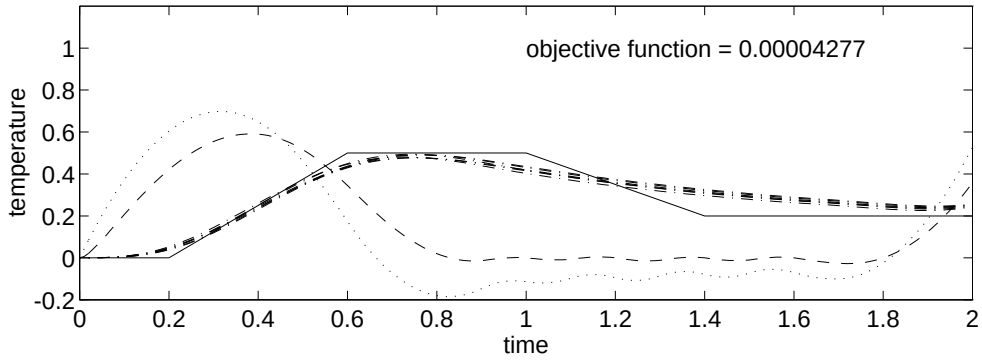

FIG. 6.9. *Optimal solution computed by DASOPT on a* 5×9 *grid with* $T_{ij}(t) \leq 0.7$, $S_{\max} = 1$, *and no lower bound on* $u(t)$. *Solid line:* $\tau(t)$. *Dotted line:* $u(t)$. *Dashed line:* $T_{00}(t)$. *Dash-dot lines:* $T_{ij}(t)$ *in* Ω_c.

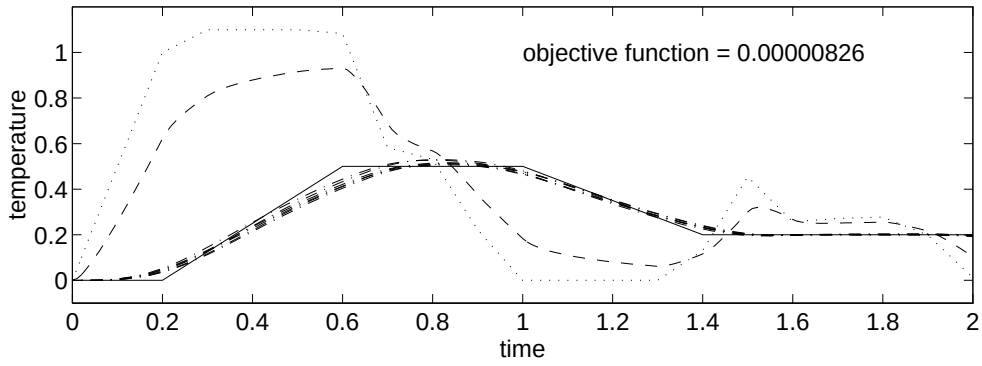

FIG. 6.10. *Optimal solution computed by DASOPT on a* 5×9 *grid with piecewise continuous linear control* $u(t)$. *Solid line:* $\tau(t)$. *Dotted line:* $u(t)$. *Dashed line:* $T_{00}(t)$. *Dash-dot lines:* $T_{ij}(t)$ *in* Ω_c.