Optimal Control of a Delay Partial Differential Equation

John T. Betts, SLC, KT

May 24, 2012

1 Introduction

Differential equations are one of the most widespread mathematical tools used to describe physical phenomena. Ordinary differential equations (ODE's) involve a single independent variable, and partial differential equations (PDE's) are used when more than one independent variable is required. It is not surprising that models used to describe a physical process often involve time as an independent variable. However, for a physical system, the current state usually depends on what happened at an earlier time. Predator-prey models describe the behavior of predators "now" feeding on populations of prey born "earlier." "In some situations, for example in real-time simulation, where time delays can be introduced by the computer time needed to compute an output after the input has been sampled, and where additional delays can be introduced by the operator-in-the-loop, differential equations with delays must be included in the model." [2]

The study of *delay-differential equations* (DDE's) as well has *delay-differential-algebraic equations* (DDAE's) has provided challenging analytic and numerical problems for over thirty years, and space precludes an exhaustive discussion of the many issues. Most widely used numerical methods for solving ordinary differential equations are based on a "stepwise" approach, that is, given values $\mathbf{y}_k$ at time t_k they construct new values $\mathbf{y}_{k+1}$ by stepping to the new time $t_k + h_k$ for a positive step h_k. Algorithmically, these methods can be implemented using information that is *local*. Single step methods, such as the classical Runge-Kutta scheme require information about the differential equations entirely within the step, i.e. for values $t_k \leq t \leq t_k + h_k$. Even multi-step methods only use information over a few nearby steps. In contrast, solution of a delay differential equation requires some type of "interpolation" for evaluating the delayed arguments at points well-removed from the local step. At the very least, a stepwise algorithm must "save" enough information about early solution steps just to construct the current step. Thus one computational approach is to combine a method for solving an ODE with a technique for interpolation of previous steps. In essence, when solving a DDE, of necessity the method is "less local" or "more global" than single step techniques widely used for solving ODE's. C.A.H. Paul [13] discusses a number of the issues that must be addressed when constructing DDE software and Shampine and Gahinet [14] present a MATLAB implementation. An example originally published in Russian by G. I. Marchuk is also cited by Hairer, Norsett, and Wanner [10, pp. 349–351]. Landry, Campbell, Morris, and Aguilar [11] analyze the dynamics of a classical "inverted pendulum" problem when the control incorporates a time delay.

While many authors have considered stability and numerical simulation of differential equations with delays, research into methods for *optimization* of such systems is relatively new. Historically the first techniques developed for optimal control of systems without delay, combined a numerical stepwise integrator with an optimization algorithm. Indeed these so-called indirect shooting methods have found wide applicability. The technique requires numerically solving the necessary conditions for optimality, i.e. the adjoint equations. Unfortunately, this approach becomes problematic, because the adjoint equations for systems with delay involve "advances," thus precluding a simple stepwise integration process. In fact, just deriving the necessary conditions for delay equations is often very difficult. In spite of these computational challenges, results have been published for some applications. Deshmukh, Ma, and Butcher [7] present a finite horizon optimal control model. The optimal control of a two-stage continuous stirred chemical tank reactor problem is

posed in Büskens, Göllmann, and Maurer [6] and revisited in the thesis of Gretschko [9]. An optimal control model that describes the immune response of a pathogenic disease process is developed in Stengel, Ghigliazza, Kulkarni, and Laplace [15]. Maurer presents an example in [12] that incorporates a non-convex model with state delay. Göllmann, Kern, and Maurer present an example in [8] that models optimal fishing, as well as another continuous stirred tank reactor system (CSTR) model. Batzel, Timischl-Teschl, and Kappel [3] describe a biomedical application of delay systems, that considers a model of the human cardiovascular-respiratory control system with one and two transport delays in the state equations describing the respiratory system.

In contrast to stepwise methods, the direct transcription method [4] treats the problem in a global fashion. The entire problem is first discretized, and then a large scale optimization algorithm is used to compute the optimal solution of the discrete approximation. Accuracy of the discrete approximation is improved by a mesh refinement algorithm. Since this "discretize then optimize" approach is global it permits looking forward and/or backwards, and does not require explicit derivation of the adjoint equations. The method has demonstrated success for problems with ordinary differential-algebraic systems. After defining the delay problem with ODE's in Section 2, we extend the transcription method to include delays in Section 3. Section 4 introduces an example that has both delays and a partial differential equation. In Section 5 we describe how the PDE can be converted to a system of ODE's using the *method of lines*. Sections 6-8 then present a series of examples that demonstrate the utility of the new method.

2 Delay Differential Equations

Many dynamic systems can be represented by a differential-algebraic equation (DAE) model of the form

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t] \tag{2.1a}$$

$$\mathbf{0} = \mathbf{g}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t] \tag{2.1b}$$

for $t_I \leq t \leq t_F$. The differential or state variables are denoted by $\mathbf{y}$, the algebraic or control variables by $\mathbf{u}$, and the (non-dynamic) parameters $\mathbf{p}$. For simplicity we consider equality path constraints (2.1b) although extending the results to inequalities is straightforward. The behavior of the system is quantified by the value of an *objective function* such as

$$F = \phi[\mathbf{y}(t_F), \mathbf{u}(t_F), \mathbf{p}, t_F] + \int_{t_I}^{t_F} w[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t]dt \tag{2.2}$$

which must be minimized by the optimal choice for the control functions $\mathbf{u}(t)$ and parameters $\mathbf{p}$.

Let us define a set of ν *delay functions*, ω_k that are prescribed functions of time for $k = 1, \ldots, \nu$. Although we generically refer to the functions as "delay" functions, as in

$$\omega_1(t) = t - \tau$$

for a delay time $\tau > 0$, the approach can also be used to model "advances" such as

$$\omega_2(t) = t + 5\tau.$$

Although the function $\omega_k(t)$ is not necessarily linear, the behavior of $\omega_k(t)$ over the entire phase is restricted. To characterize delay we require

$$\omega_k(t) \leq t \qquad\qquad t_I \leq t \leq t_F. \tag{2.3a}$$

To represent advance we must have

$$\omega_k(t) \geq t \qquad\qquad t_I \leq t \leq t_F. \tag{2.3b}$$

These restrictions permit a zero delay, e.g. $\omega_k(t) = t$ but preclude a reverse in direction from delay to advance, or vice versa.

Let us denote the *delayed state vector* by $\mathbf{z}(\omega_k)$. Thus the vector $\mathbf{z}(\omega_k)$ of length n_z ($n_z \leq n_y$) denotes the subset of the n_y state variables that are functions of the delay argument ω_k. The goal of this paper is to extend the DAE model (2.1a)-(2.1b) to systems of the form

$$\dot{\mathbf{y}} = \mathbf{f}\left[\mathbf{y}(t), \mathbf{u}(t), \mathbf{z}(\omega_1), \ldots, \mathbf{z}(\omega_\nu), \mathbf{p}, t\right] \tag{2.4a}$$

$$\mathbf{0} = \mathbf{g}\left[\mathbf{y}(t), \mathbf{u}(t), \mathbf{z}(\omega_1), \ldots, \mathbf{z}(\omega_\nu), \mathbf{p}, t\right]. \tag{2.4b}$$

Delay equations introduce an issue not encountered in a standard DAE problem. For a system defined by the DAE model (2.1a)-(2.1b) a complete description usually requires boundary conditions at the initial time t_I and/or the final time t_F. The boundary conditions can involve specified values for the states or nonlinear conditions $\psi(\mathbf{y}, \mathbf{u}, t) = 0$ and are enforced at specific points, namely at t_I and/or t_F. In contrast, for a delay equation such as (2.4a)-(2.4b), there must be a start up region. Specifically for delay problems as in (2.3a), we must evaluate $\mathbf{z}[\omega_k(t)]$. Thus we require

$$\mathbf{z}[\omega_k(t)] = \boldsymbol{\alpha}(\mathbf{p}, t) \qquad \text{for} \qquad \omega_k(t) \leq t_I \tag{2.5}$$

where the *prehistory* $\boldsymbol{\alpha}(\mathbf{p}, t)$ must be a function defined on the startup region $\omega_k(t) \leq t_I$. A similar function may be required at the final time for problems with advances.

3 Direct Transcription Method

When solving an optimal control problem without delays as given by (2.1a)-(2.2), the *direct transcription method* has been very effective (cf Ref [4]). Our goal is to extend the method to a new class of problems that include delay dynamics as given by (2.4a)-(2.4b). In general, the method has three basic steps:

1. **Direct Transcription** Transcribe the optimal control problem into a nonlinear programming (NLP) problem by discretization;

2. **Sparse Nonlinear Program** Solve the sparse NLP

3. **Mesh Refinement** Assess the accuracy of the approximation (i.e. the finite dimensional problem), and if necessary refine the discretization, and then repeat the optimization steps.

As such the approach can be considered a *sequential nonlinear programming* or (SNLP) method. In our software implementation the sparse NLP can be solved using either a sequential quadratic programming (SQP) or primal-dual barrier (interior point) method.

Let us address a DDAE system of the form (2.4a)-(2.4b), and to simplify the presentation temporarily assume there are no parameters and a single delay function. Define new algebraic variables $\mathbf{v}(t)$, and require they be consistent with the delayed variables. The original DDAE system can now be written as

$$\dot{\mathbf{y}} = \mathbf{f}\left[\mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), t\right] \tag{3.1}$$

$$\mathbf{0} = \mathbf{g}\left[\mathbf{y}(t), \mathbf{u}(t), \mathbf{v}(t), t\right] \tag{3.2}$$

$$\mathbf{0} = \mathbf{v}(t) - \mathbf{z}[\omega(t)] \tag{3.3}$$

The basic idea is to first discretize the problem thereby creating a finite dimensional approximation. Large scale optimization methods can then be used to adjust the variables that define the discretization. The direct transcription approach introduces a discretization of the problem by subdividing the time domain into n_s segments or intervals

$$t_I = t_1 < t_2 < \cdots < t_M = t_F, \tag{3.4}$$

where the points are referred to as node, mesh, or grid points. We use $\mathbf{y}_k \equiv \mathbf{y}(t_k)$ to indicate the value of the state variable at a grid point, with a similar notation for the algebraic variables. Thus one treats

$$\mathbf{x}^\mathsf{T} = (\mathbf{y}_1, \mathbf{u}_1, \mathbf{v}_1, \ldots, \mathbf{y}_M, \mathbf{u}_M, \mathbf{v}_M) \tag{3.5}$$

as optimization variables in a nonlinear programming problem. We then approximate the differential equation using a nonlinear algebraic constraint. When the discretization is based on an implicit Runge-Kutta (IRK) scheme the control problem is *transcribed* into a finite dimensional nonlinear program.

Thus for the simplest IRK scheme, the trapezoidal method, we approximate the differential equations (3.1) and algebraic constraints (3.2) by a set of algebraic constraints given by

$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \frac{h_k}{2}\left(\mathbf{f}_k + \mathbf{f}_{k+1}\right) = \boldsymbol{\zeta}_k \qquad k = 1, \ldots, M-1 \tag{3.6}$$

$$\mathbf{0} = \mathbf{g}_k \qquad k = 1, \ldots, M \tag{3.7}$$

where (3.6) are referred to as the *defect* constraints $\boldsymbol{\zeta}_k = \mathbf{0}$. This discretization is implicit because the optimization variables $(\mathbf{y}_k, \mathbf{u}_k, \mathbf{v}_k)$ appear as arguments in the nonlinear functions $\mathbf{f}_k \equiv \mathbf{f}(\mathbf{y}_k, \mathbf{u}_k, \mathbf{v}_k)$ and $\mathbf{g}_k \equiv \mathbf{g}(\mathbf{y}_k, \mathbf{u}_k, \mathbf{v}_k)$.

Now in order to evaluate the right hand side functions in (3.1) and (3.2) we must express the consistency relationship (3.3) in terms of the NLP optimization variables. When the delay argument is exterior to the phase, that is when $\omega_k < t_1$ (or $\omega_k > t_M$) the value of the delayed state is given by the user defined function $\alpha(\mathbf{p}, \omega_k)$. When the delay argument is interior to the phase $t_I \le \omega_k \le t_F$ let us define the interval J such that

$$t_J \le \omega(t_k) \le t_{J+1} \tag{3.8}$$

as illustrated in Figure 3.1. Then for $\delta_k = (\omega_k - t_J)/h_J = (\omega_k - t_J)/(t_{J+1} - t_J)$ the interpolated value for the delayed state is just

$$z[\omega(t_k)] \doteq z(\omega_k) = c_1 z_J + c_2 z_{J+1} + c_3 \dot{z}_J + c_4 \dot{z}_{J+1} \tag{3.9}$$

where the Hermite interpolation coefficients are

$$c_1 = (1-\delta_k)^2(1+2\delta_k) \quad c_2 = \delta_k^2(3-2\delta_k) \quad c_3 = \delta_k(1-\delta_k)^2 h_J \quad c_4 = -\delta_k^2(1-\delta_k)h_J. \tag{3.10}$$

Thus to enforce consistency at the grid points we can impose the NLP constraints

$$v(t_k) = z[\omega(t_k)] = \begin{cases} z(\omega_k) & \text{if } t_1 \le \omega_k \le t_M, \\ \alpha(\mathbf{p}, \omega_k) & \text{otherwise.} \end{cases} \tag{3.11}$$

Of course these consistency constraints are also implicit, since they involve the derivatives $\dot{z}_J$ and $\dot{z}_{J+1}$ which are given by the right hand sides of (3.1).

When using a Hermite-Simpson discretization two things change. First the NLP variables are

$$\mathbf{x}^\mathsf{T} = (\mathbf{y}_1, \mathbf{u}_1, \mathbf{v}_1, \mathbf{u}_{3/2}, \mathbf{v}_{3/2}, \ldots, \mathbf{y}_M, \mathbf{u}_M, \mathbf{v}_M) \tag{3.12}$$

where $\mathbf{u}_{k+\frac{1}{2}} = \mathbf{u}[\frac{1}{2}(t_j + t_{j+1})]$ is an additional NLP variable representing the control at the midpoint. Second the discretization constraints (3.6)-(3.7) are replaced by

$$\mathbf{0} = \mathbf{y}_{k+1} - \mathbf{y}_k - \frac{h_k}{6}\left(\mathbf{f}_k + 4\mathbf{f}_{k+\frac{1}{2}} + \mathbf{f}_{k+1}\right) = \boldsymbol{\zeta}_k \qquad k = 1, \ldots, M-1 \tag{3.13}$$

$$\mathbf{0} = \mathbf{g}_k \qquad k = 1, \ldots, M \tag{3.14}$$

$$\mathbf{0} = \mathbf{g}_{k+\frac{1}{2}} \qquad k = 1, \ldots, M-1 \tag{3.15}$$

To summarize, for the Hermite-Simpson discretization the NLP variables (3.12) must be chosen to satisfy the NLP constraints (3.13)-(3.15). Furthermore the NLP constraints (3.11) are imposed at both gridpoints and midpoints to ensure satisfying the consistency relationships (3.3).

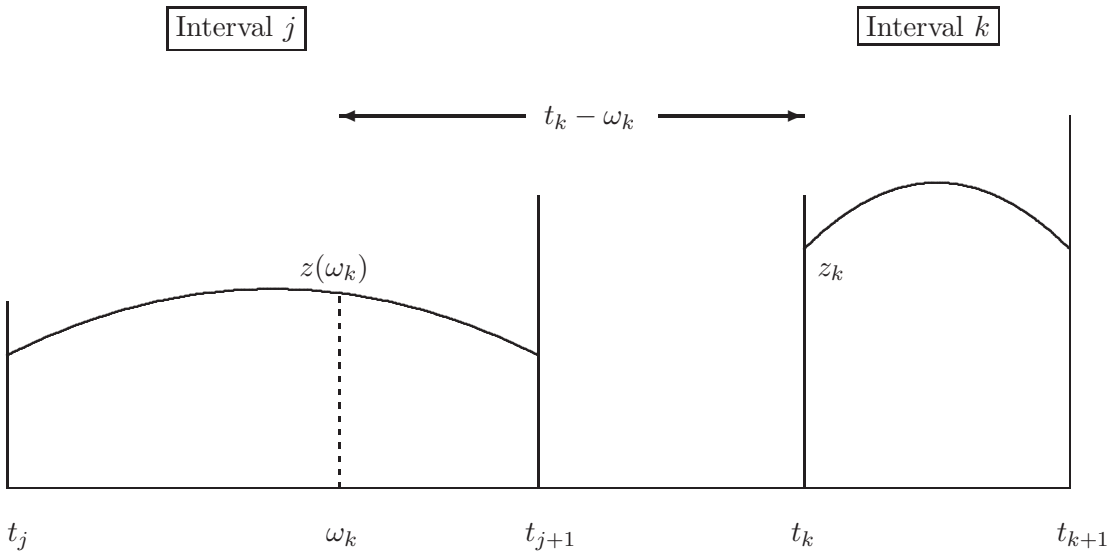

Figure 3.1: Consistency Construction

In order to efficiently solve the resulting large nonlinear programming problem it is imperative to exploit sparsity in the Jacobian and Hessian matrix. In particular the Jacobian will involve derivatives of the constraints with respect to the variables $\mathbf{x}$ given by (3.5) or (3.12). These matrices will be sparse, because the subset

$$(\mathbf{y}_k, \mathbf{y}_j, \mathbf{y}_{j+1}, \mathbf{u}_k, \mathbf{u}_j, \mathbf{u}_{j+1}, \mathbf{v}_k, \mathbf{v}_j, \mathbf{v}_{j+1}, t_k)$$

are the only variables that appear. In order to fully exploit sparsity it is important to exploit separability. This can be achieved by writing the constraint functions such that the linear terms are isolated from the nonlinear ones. Specifically using Eq. 4.114 in Reference [4], we write

$$\begin{bmatrix} \mathbf{c}(\mathbf{x}) \\ F(\mathbf{x}) \end{bmatrix} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{q}(\mathbf{x}), \tag{3.16}$$

where the vector $\mathbf{q}$ consists of the nonlinear right hand side functions $\mathbf{f}_k$ and $\mathbf{g}_k$ evaluated at the grid points. Sparse finite difference estimates can be used to compute the Jacobian and Hessian matrices as described in Ref [4]. Furthermore because the functions $\mathbf{f}$ and $\mathbf{g}$ appear linearly in the transcribed formulation, sparsity in these right hand side functions can be exploited (cf Eq. 4.115 in Ref [4]).

There are two key ideas needed to extend the direct transcription method to accommodate models with delays. First, we introduce a new algebraic variable, i.e. the *pseudo-state* $\mathbf{v}$ to represent the delayed state variable. We then introduce an algebraic (path) constraint, to ensure that $\mathbf{v}$ is consistent with the Hermite interpolant for the value of the real state at the delayed time. This leads to the NLP consistency equations (3.11), after transcribing the continuous problem into its finite dimensional counterpart.

Although most of the computational effort is the same for problems with and without delay, there is one change required in the mesh refinement procedure. In particular the refinement algorithm described in Ref. [4], which is based on work in [5], proceeds by systematically subdividing the interval containing the largest discretization error. In contrast, for delay problems, when the largest error occurs in a particular interval k where $t_k \leq t \leq t_{k+1}$ we must also subdivide the delay interval J where $t_J \leq \omega(t_k) \leq t_{J+1}$. Thus the "look back" affect illustrated in Figure 3.1, also is implemented during the mesh refinement process.

4 Delay Partial Differential Equation Example

The preceding section described how to extend the direct transcription method to ordinary differential-algebraic equations with delay. We now go one step farther and demonstrate the performance of the algorithm on a partial differential equation (PDE) with delay. First we describe the PDE problem, and then we discuss how it can be approximated by a system of ordinary differential equations.

Consider a modification to a reaction diffusion model of a species. Define $Y(x,t)$ as the population density at time t and position x where $0 \le t \le T$ and $0 \le x \le \pi$.

$$\frac{\partial Y(x,t)}{\partial t} = c_1 \frac{\partial^2 Y(x,t)}{\partial x^2} - c_2 Y(x,\omega(x,t))\left[1 + Y(x,t)\right] + U(x,t) \tag{4.1}$$

with boundary conditions

$$\frac{\partial Y(0,t)}{\partial x} = \frac{\partial Y(\pi,t)}{\partial x} = 0 \tag{4.2}$$

$$Y(x,0) = \alpha(x) = 1 + \sin(2x - \frac{\pi}{2}) \tag{4.3}$$

The (positive) diffusion parameter $c_1 = 1$, and the coefficient $c_2 = .5$ defines what the reaction consumes. The duration of the process is $T = 5$ and the goal is to minimize

$$F = \int_0^\pi \left[\int_0^T c_3 U^2(x,t)dt\right] dx + \int_0^\pi \left[Y(x,T) - h(x)\right]^2 dx \tag{4.4}$$

where $h(x)$ is the desired final population profile and $c_3 = 0.1$ is a weight.

For the sake of illustration we will consider problems with various delay functions $\omega(x,t)$. For examples with no spatial dependence we will consider the constant delay function

$$\omega(t) = t - r \tag{4.5a}$$

defined with delay time $r = .5$ as well as the variable delay functions

$$\omega(t) = t - r\cos\left(\frac{t\pi}{2T}\right) \tag{4.5b}$$

$$\omega(t) = t - r\left(1 - \frac{t}{T}\right) \tag{4.5c}$$

$$\omega(t) = t - \frac{rt}{T} \tag{4.5d}$$

$$\omega(t) = t - 1 - \frac{1}{t+1} \tag{4.5e}$$

$$\omega(t) = t - 1 - \log(t+1) \tag{4.5f}$$

$$\omega(t) = \frac{t}{2}. \tag{4.5g}$$

For both (4.5b) and (4.5c) the delay starts at r and decreases to 0 as time goes from 0 to T. In contrast for (4.5d) the delay begins at zero and then increases to r at the final time T. The variable delay functions (4.5e)-(4.5g) are suggested in Reference [16]. For (4.5g) the delay increases from zero to $T/2$. No prehistory is required for delay functions (4.5d) and (4.5g). For examples that require a prehistory when $\omega(x,t) < 0$ we set

$$Y[x,\omega(x,t)] = \alpha(x) \qquad\qquad 0 \le x \le \pi. \tag{4.6}$$

5 Method of Lines Formulation

Let us first transform the partial differential equation into a system of ordinary delay-differential equations using the method of lines. To do so introduce a spatial discretization, i.e.

$$x_k = k\delta = k\frac{\pi}{n} \qquad\qquad k = 0, \ldots, n. \tag{5.1}$$

If we denote $Y(x_k, t) = y_k(t)$, $U(x_k, t) = u_k(t)$, and $\omega(x_k, t) = \omega_k(t)$ then we can introduce a second order difference approximation for $\frac{\partial^2 Y}{\partial x^2}$. Using this approximation the PDE system Eq. (4.1) is replaced by the system of delay-differential equations

$$\dot{y}_0 = \frac{c_1}{\delta^2}\left(y_1 - 2y_0 + y_{-1}\right) - c_2 y_0(\omega_0)\left(1 + y_0\right) + u_0 \tag{5.2}$$

$$\dot{y}_k = \frac{c_1}{\delta^2}\left(y_{k+1} - 2y_k + y_{k-1}\right) - c_2 y_k(\omega_k)\left(1 + y_k\right) + u_k \qquad k = 1, \ldots, n-1 \tag{5.3}$$

$$\dot{y}_n = \frac{c_1}{\delta^2}\left(y_{n+1} - 2y_n + y_{n-1}\right) - c_2 y_n(\omega_n)\left(1 + y_n\right) + u_n. \tag{5.4}$$

Using a second order approximation to the first derivative the boundary conditions Eq. (4.2) lead to the the algebraic equations

$$\frac{\partial Y(0,t)}{\partial x} \approx \frac{1}{2\delta}(y_1 - y_{-1}) = 0 \tag{5.5}$$

$$\frac{\partial Y(\pi,t)}{\partial x} \approx \frac{1}{2\delta}(y_{n+1} - y_{n-1}) = 0. \tag{5.6}$$

From these equations it follows that

$$y_{-1} = y_1 \tag{5.7}$$

$$y_{n+1} = y_{n-1} \tag{5.8}$$

When the expressions for y_{-1} from (5.7) and y_{n+1} from (5.8) are substituted into the system (5.2)-(5.4) we obtain

$$\dot{y}_0 = \frac{2c_1}{\delta^2}\left(y_1 - y_0\right) - c_2 y_0(\omega_0)\left(1 + y_0\right) + u_0 \tag{5.9}$$

$$\dot{y}_k = \frac{c_1}{\delta^2}\left(y_{k+1} - 2y_k + y_{k-1}\right) - c_2 y_k(\omega_k)\left(1 + y_k\right) + u_k \qquad k = 1, \ldots, n-1 \tag{5.10}$$

$$\dot{y}_n = \frac{2c_1}{\delta^2}\left(y_{n-1} - y_n\right) - c_2 y_n(\omega_n)\left(1 + y_n\right) + u_n. \tag{5.11}$$

After introducing the method of lines approximation and denoting $\alpha(x_k) = \alpha_k$ the initial condition Eq. (4.3) leads to the initial conditions

$$y_k(0) = \alpha_k \qquad k = 0, \ldots, n \tag{5.12}$$

and the prehistory defined when $\omega(x,t) < 0$

$$y_k(t) = \alpha_k \qquad k = 0, \ldots, n. \tag{5.13}$$

The spatial integration in the objective can be approximated using a trapezoidal quadrature rule so

$$F = \frac{1}{2}\delta f_0 + \delta \sum_{k=1}^{n-1} f_k + \frac{1}{2}\delta f_n \tag{5.14a}$$

where

$$f_k = \int_0^T c_3 u_k^2(t)dt + \left[y_k(T) - h(x_k)\right]^2 \tag{5.14b}$$

6 Delay PDE with Single Control

To simplify the development here as well as in Appendix A, temporarily let us assume that the control depends only on time, that is $U(x,t) = u(t)$ and the target profile is just $h(x) = 5$. For this example the dynamics that determine the states $(y_0, \ldots, y_n)$ and control u are just

$$\dot{y}_0 = \frac{2c_1}{\delta^2}(y_1 - y_0) - c_2 y_0(\omega)[1 + y_0] + u \tag{6.1}$$

$$\dot{y}_k = \frac{c_1}{\delta^2}(y_{k+1} - 2y_k + y_{k-1}) - c_2 y_k(\omega)[1 + y_k] + u \qquad k = 1, \ldots, n-1 \tag{6.2}$$

$$\dot{y}_n = \frac{2c_1}{\delta^2}(y_{n-1} - y_n) - c_2 y_n(\omega)[1 + y_n] + u. \tag{6.3}$$

In this case the objective function given by (5.14a) and (5.14b) can be simplified to

$$F = \int_0^T c_3 u^2(t)dt + \frac{1}{2}\delta f_0 + \delta \sum_{k=1}^{n-1} f_k + \frac{1}{2}\delta f_n \tag{6.4a}$$

where

$$f_k = [y_k(T) - h(x_k)]^2. \tag{6.4b}$$

For problems having a constant delay function (4.5a) it is possible to construct an equivalent system of ordinary differential equations with *no delay*. This technique, called the *method of steps* [1, 10], is described in Appendix A, and can be used to verify results computed by the new delay algorithm.

6.1 Numerical Results

The partial differential delay problem defined by (6.1)-(6.3) has been solved for all of the cases discussed using a spatial discretization with $n = 15$. The mesh in the time domain was refined until the discretization error $\epsilon \leq 10^{-7}$ to give a solution with approximately eight significant figures accuracy. The method of lines formulation results in a system with 16 state variables and 17 algebraic variables. The results are summarized in Table 6.1. Notice that the optimal objective function value obtained using the method of steps compares closely with the constant delay case in the second line of Table 6.1 as it should. In contrast, there is a significant difference in both the number of grid points and the solution time. Since the time domain is broken into 10 regions when using the method of steps, the "real" number of mesh points is $10 \times 42 = 420$, which is considerably more than the 168 points needed using the new delay approach. The method of steps also leads to an ODE system with 160 states and 10 control variables which also explains why the solution time for the new approach is nearly twice as fast as the method of steps.

7 Delay PDE with Spatial Control

Let us now consider the original problem treating the control as a function of both space and time. The target profile is $h(x) = 6 - 3\cos(3x)$, and as before we choose $c_3 = 0.1$. In this case the system (5.9)-(5.11) is replaced by the system

$$\dot{y}_0 = \frac{2c_1}{\delta^2}(y_1 - y_0) - c_2 y_0(\omega)(1 + y_0) + u_0 \tag{7.1}$$

$$\dot{y}_k = \frac{c_1}{\delta^2}(y_{k+1} - 2y_k + y_{k-1}) - c_2 y_k(\omega)(1 + y_k) + u_k \qquad k = 1, \ldots, n-1 \tag{7.2}$$

$$\dot{y}_n = \frac{2c_1}{\delta^2}(y_{n-1} - y_n) - c_2 y_n(\omega)(1 + y_n) + u_n. \tag{7.3}$$

Thus to summarize the optimal control problem;

Description	M	Time	F^*
Method of Steps	38	19.1291	3.64669763
Constant Delay, cf. (4.5a)	170	11.0973	3.64661936
Variable Delay, cf. (4.5b)	183	16.9664	8.37257003
Variable Delay, cf. (4.5c)	178	11.4313	8.47992369
Variable Delay, cf. (4.5d)	149	9.40557	3.76411998
Variable Delay, cf. (4.5e)	194	12.0362	1.12890127
Variable Delay, cf. (4.5f)	158	4.89126	.492930043
Variable Delay, cf. (4.5g)	125	7.62984	.415342751

M Grid Points Time CPU seconds F^* Optimal Objective

Table 6.1: Performance Summary; Time Dependent Control

- determine the control variables $[u_0(t), \ldots, u_n(t)]$
- and state variables $[y_0(t), \ldots, y_n(t)]$
- to satisfy the differential equations (7.1)-(7.3)
- the initial conditions (5.12)
- the prehistory conditions (5.13)
- to minimize the objective function (5.14a).

7.1 Numerical Results

This problem has been solved for all of the delay functions (4.5a)-(4.5f) using a spatial discretization with $n = 15$, and Table 7.1 summarizes the computational performance. This problem formulation leads to a delay system with 16 states, and 32 algebraic variables (16 "real" controls, and 16 "pseudo-states").

Description	M	Ref	n	NDOF	Time	F^*
Constant Delay, cf. (4.5a)	191	8	15248	6096	20.9858	23.9264815
Variable Delay, cf. (4.5b)	207	8	16528	6608	16.3575	40.3828187
Variable Delay, cf. (4.5c)	191	8	15248	6096	11.8072	40.6791184
Variable Delay, cf. (4.5d)	199	9	15888	6352	22.4736	24.3653961
Variable Delay, cf. (4.5e)	269	9	21488	8592	36.3875	13.3087632
Variable Delay, cf. (4.5f)	254	11	20288	8112	19.9790	10.4068832
Variable Delay, cf. (4.5g)	233	8	18608	7440	27.2819	10.0826024

M	Grid Points	Ref	Number of Refinements
n	NLP variables	NDOF	Number of Degrees of Freedom
Time	CPU seconds	F^*	Optimal Objective

Table 7.1: Performance Summary; Time and Space Dependent Control

7.2 Bounded Control

For the sake of illustration let us revisit the case with $\omega(t) = \frac{t}{2}$ and add lower bounds on the control variable, that is,

$$U(x,t) \geq 0. \tag{7.4}$$

As expected when the control bounds are imposed the optimal objective function value increases from $F^* = 10.0826024$ to $F^* = 14.6730062$. The solution is illustrated in Figures 7.1 and 7.2. It is clear from Figure 7.2 that the control bound prevents any negative values for $U(x,t)$. In particular, at the optimal solution some of the discretized control variables lie on the lower bound, that is the *active set* is

$$\mathcal{A} = \{(k,j) \quad \text{such that} \quad |U^*(x_k,t_j)| \leq \epsilon\}$$

for some tolerance ϵ. These values are considered strictly active by the nonlinear programming algorithm. Figure 7.3 illustrates the control variable active set.

The active set is determined by the nonlinear program, and the computational cost is reflected in the overall problem solution times. Table 7.2 compares the performance of the sparse SQP and barrier algorithms for this example. Observe that the SQP algorithm which is an *active set* method is nearly four times as expensive as the primal-dual barrier algorithm. It is also interesting to note that the barrier algorithm required an additional mesh refinement iteration in comparison to the SQP algorithm. The mesh refinement algorithm corrects the small differences between the intermediate NLP solutions using a slightly different grid distribution.

	SQP Method			Barrier Method		
Ref	M	ϵ	Time	M	ϵ	Time
1	11	1.1497×10^{-1}	.21797	11	1.1498×10^{-1}	.14298
2	21	3.1822×10^{-2}	.41694	21	3.1823×10^{-2}	.35395
3	21	9.6717×10^{-3}	2.9076	21	9.6716×10^{-3}	2.2317
4	41	6.8865×10^{-4}	9.3836	41	6.8819×10^{-4}	5.1452
5	77	3.9014×10^{-5}	39.045	76	3.8952×10^{-5}	13.165
6	95	7.2941×10^{-6}	58.623	94	7.2938×10^{-6}	20.785
7	189	1.5648×10^{-6}	284.59	187	1.5493×10^{-6}	54.485
8	206	3.1929×10^{-7}	338.99	204	3.1939×10^{-7}	61.543
9	243	1.3817×10^{-7}	440.23	243	1.3481×10^{-7}	82.291
10	248	9.8565×10^{-8}	469.76	248	1.7388×10^{-7}	70.759
11				253	9.8263×10^{-8}	75.775
Total			1644.2			386.68

Table 7.2: Performance Comparison; SQP versus Barrier NLP

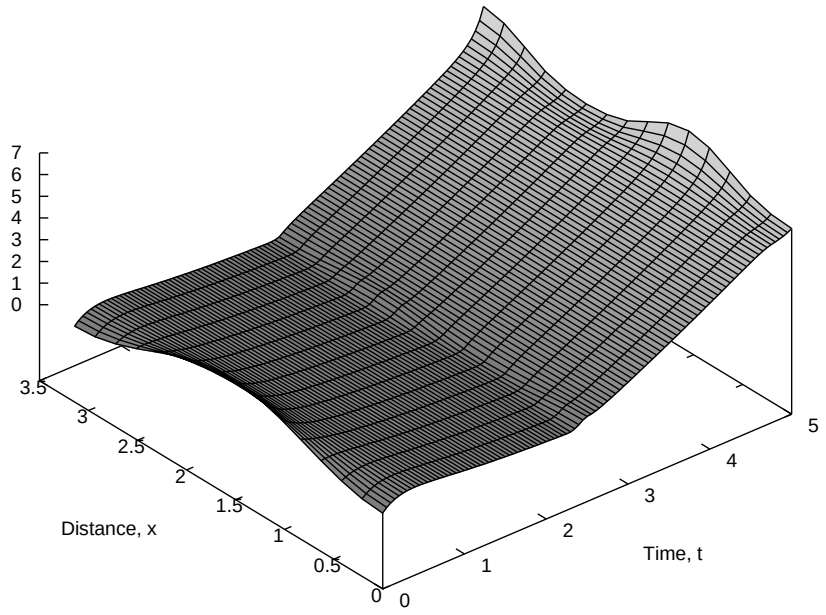

Figure 7.1: Optimal State $Y^*(x,t)$, $\omega(t) = \frac{t}{2}$

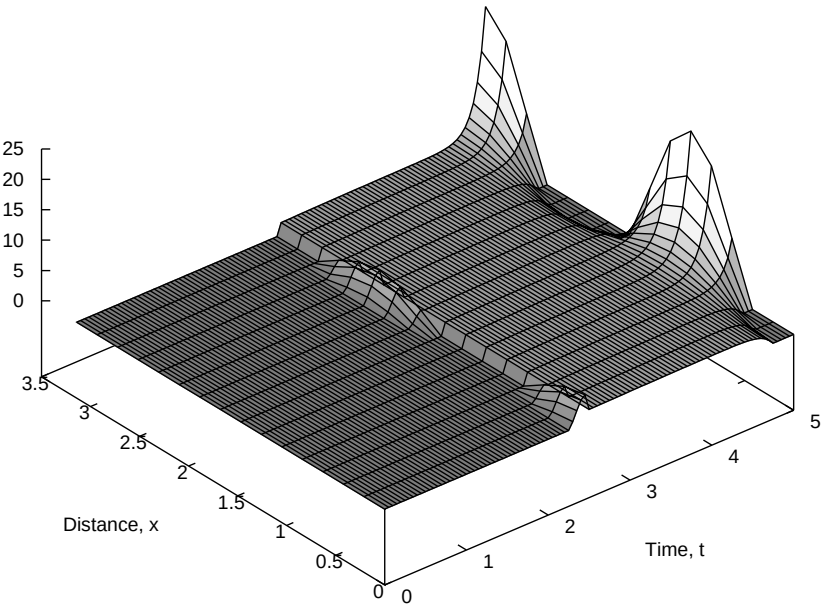

Figure 7.2: Optimal Control $U^*(x,t)$, $\omega(t) = \frac{t}{2}$

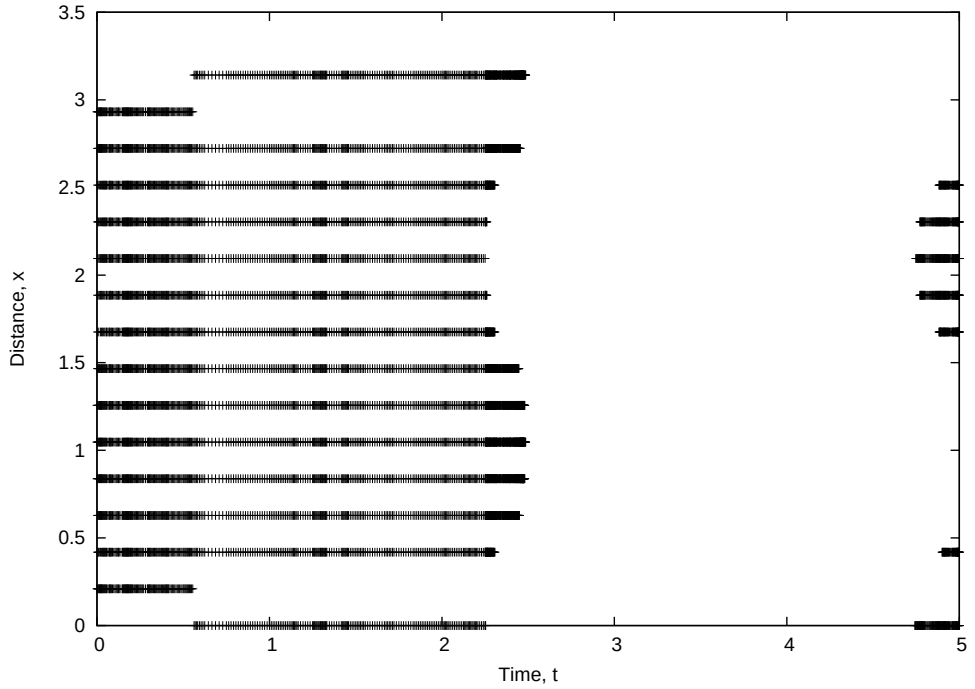

Figure 7.3: Optimal Control Active Set, $U^*(x_k, t_j) = 0$, $\omega(t) = \frac{t}{2}$

8 Delay PDE with Spatial Control and Spatial Delay

Let us now consider an example in which the delay function is also spatially dependent. Instead of delay functions given by (4.5a) - (4.5g) let us consider a spatially dependent delay function similar to (4.5d)

$$\omega(x,t) = t - \frac{rt}{T}\left[\frac{4x(\pi - x)}{\pi^2}\right]. \tag{8.1}$$

Again using the method of lines formulation we must solve the system (5.9) - (5.11) with $n+1$ spatially dependent delay functions

$$\omega_j(t) = t - \frac{rt}{T}\left[\frac{4x_j(\pi - x_j)}{\pi^2}\right] \qquad x_j = j\delta \qquad j = 0, \ldots, n. \tag{8.2}$$

However, we can also exploit the spatial properties of this particular delay function to further simplify the formulation. First, observe that for the boundary lines at $x_0 = 0$ and $x_n = \pi$, there is *no* delay since $\omega_0(t) = \omega_n(t) = t$. Consequently, for this example $y_0(\omega_0) = y_0$ and $y_n(\omega_n) = y_n$. Furthermore, the spatial dependence is symmetric about the point $x = \pi/2$. Thus we have

$$\omega_j(t) = \omega_{n-j}(t) \qquad j < n/2 \tag{8.3}$$

When this spatial symmetry is reflected in the system (5.9)-(5.11) we obtain

$$\dot{y}_0 = \frac{2c_1}{\delta^2}(y_1 - y_0) - c_2 y_0 (1 + y_0) + u_0 \tag{8.4}$$

$$\dot{y}_k = \frac{c_1}{\delta^2}(y_{k+1} - 2y_k + y_{k-1}) - c_2 y_k(\omega_k)(1 + y_k) + u_k \qquad k = 1, \ldots, \nu \tag{8.5}$$

$$\dot{y}_k = \frac{c_1}{\delta^2}(y_{k+1} - 2y_k + y_{k-1}) - c_2 y_k(\omega_{n-k})(1 + y_k) + u_k \qquad k = \nu + 1, \ldots, n-1 \tag{8.6}$$

$$\dot{y}_n = \frac{2c_1}{\delta^2}(y_{n-1} - y_n) - c_2 y_n (1 + y_n) + u_n. \tag{8.7}$$

where the quantity $n/2$ is treated as an integer, and $\nu \leq n/2$.

8.1 Numerical Results

This example is characterized by delay and control functions that depend on both time and space. Figure 8.1 illustrates the behavior of the delay surface, and the optimal solution is illustrated in Figures 8.2 and 8.3. Notice that when the method of lines is applied to this problem, there is no state delay for the boundary lines at $x_0 = 0$ and $x_n = \pi$, leading to the ordinary differential equations (8.4) and (8.7). Consequently the final problem has 16 states, and 16 real controls, but only 14 pseudo-states v_k needed to enforce consistency for the $\nu = 7$ distinct delay functions. A discretization error tolerance of 10^{-5} was used to refine the mesh and achieve approximately six significant figures of accuracy in the solution. The optimal objective function was $F^* = 30.0691624$.

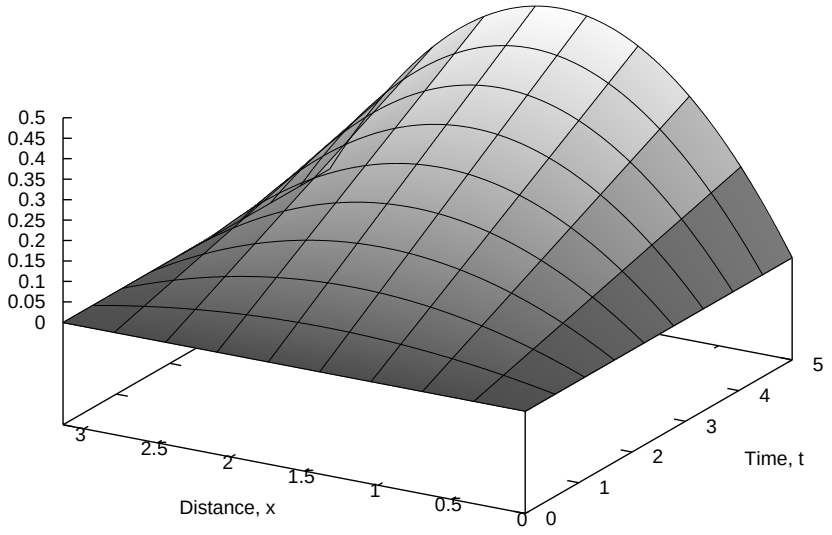

Figure 8.1: Delay Function, $t - \omega(x,t) = \frac{rt}{T}\left[\frac{4x(\pi-x)}{\pi^2}\right]$

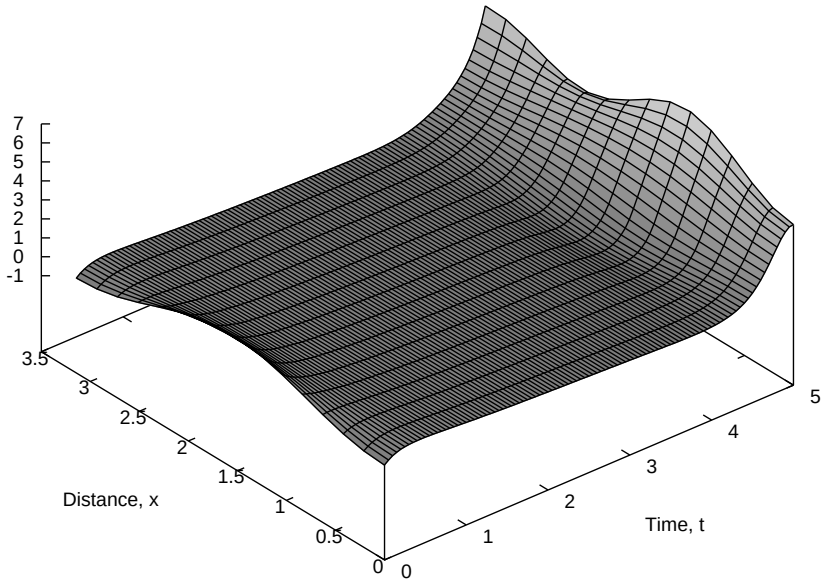

Figure 8.2: Optimal State $Y^*(x,t)$, $\omega(x,t) = t - \frac{rt}{T}\left[\frac{4x(\pi-x)}{\pi^2}\right]$

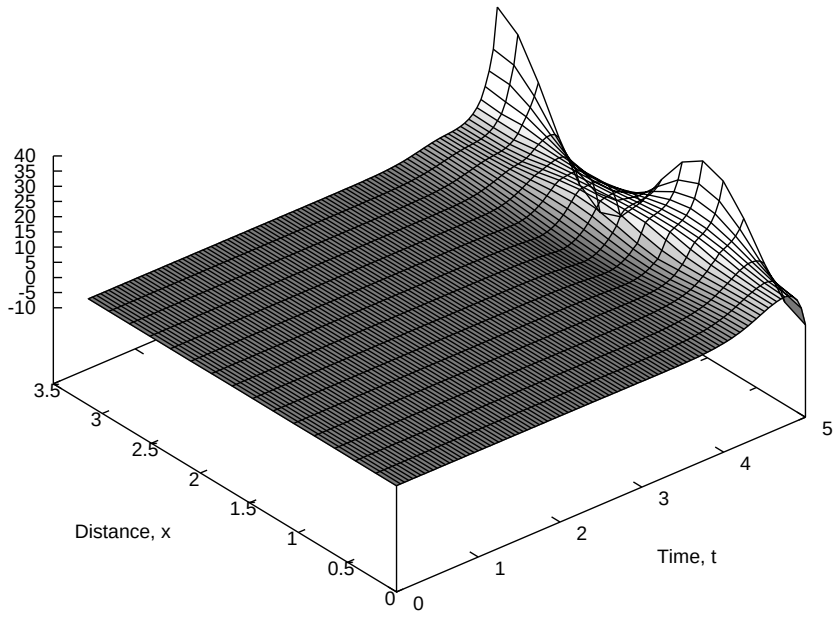

Figure 8.3: Optimal Control $U^*(x,t)$, $\omega(x,t) = t - \frac{rt}{T}\left[\frac{4x(\pi-x)}{\pi^2}\right]$

9 Summary and Conclusions

A new technique for computing the optimal control of delay-differential-algebraic dynamic systems has been introduced. The approach is illustrated by converting a problem described by a partial differential equation with delays, into a system or ordinary differential equations with delays. Results are presented for problems with constant delay, time varying delay, and delay that has both

time and spatial dependence. Although we have focused on problems with delayed state variables, we expect many of the ideas can be extended to control delays.

A Method of Steps Formulation

When the delay is constant as in (4.5a) the delay system Eq. (6.1)-Eq. (6.3) can be transformed into a system of ordinary differential equations with no delay using the method of steps. Specifically we break up the time domain into steps of length r and then define

$$\mathbf{s}_j(\rho) = \mathbf{y}(t) \qquad\qquad (j-1)r \le t \le jr \tag{A.1}$$

for $j = 0, \ldots, N$, where the number of steps $N = T/r = 10$ and $0 \le \rho \le r$. Denote the vector $\mathbf{s}_j(\rho)$ of length $(n+1)$ representing the state on delay step j by

$$\mathbf{s}_j(\rho) = \begin{bmatrix} \mathbf{y}_0[\rho + (j-1)r] \\ \vdots \\ \mathbf{y}_n[\rho + (j-1)r] \end{bmatrix} \tag{A.2}$$

for $j = 0, \ldots, N$, and $0 \le \rho \le r$. For notational clarity it is convenient to introduce a matrix with $(n+1)$ rows and $(N+1)$ columns:

$$\mathbf{S}(\rho) = \begin{bmatrix} \mathbf{s}_0(\rho) & \mathbf{s}_1(\rho) & \cdots & \mathbf{s}_N(\rho) \end{bmatrix} = \begin{bmatrix} \mathbf{y}_0(\rho - r) & \mathbf{y}_0(\rho - r + r) & & \mathbf{y}_0(\rho - r + Nr) \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{y}_n(\rho - r) & \mathbf{y}_n(\rho - r + r) & & \mathbf{y}_n(\rho - r + Nr) \end{bmatrix} \tag{A.3}$$

Note that each column of $\mathbf{S}$ corresponds the state vector $\mathbf{y}$ on a particular step. Thus element $S_{k,j}$ denotes state vector k at delay step j.

A similar notational convention must be introduced for the control, namely for $j = 1, \ldots, N$

$$u_j(\rho) = u[\rho + (j-1)r]. \tag{A.4}$$

For steps $j = 1, \ldots, N$ the dynamics in the region $0 \le \rho \le r$ become

$$\dot{S}_{0,j} = \frac{2c_1}{\delta^2}\left(S_{1,j} - S_{0,j}\right) - c_2 S_{0,j-1}\left[1 + S_{0,j}\right] + u_j \tag{A.5}$$

$$\dot{S}_{k,j} = \frac{c_1}{\delta^2}\left(S_{k+1,j} - 2S_{k,j} + S_{k-1,j}\right) - c_2 S_{k,j-1}\left[1 + S_{k,j}\right] + u_j \qquad k = 1, \ldots, n-1 \tag{A.6}$$

$$\dot{S}_{n,j} = \frac{2c_1}{\delta^2}\left(S_{n-1,j} - S_{n,j}\right) - c_2 S_{n,j-1}\left[1 + S_{n,j}\right] + u_j \tag{A.7}$$

To ensure continuity across the step boundaries, we must impose the boundary conditions

$$S_{k,j}(0) = S_{k,j-1}(r) \tag{A.8}$$

$$u_j(0) = u_{j-1}(r) \tag{A.9}$$

for the interior steps $j = 2, \ldots, N$ and all states $k = 0, \ldots, n$. The initial condition (5.12)

$$S_{k,1}(0) = \alpha_k \qquad k = 0, \ldots, n \tag{A.10}$$

and the prehistory requires

$$S_{k,0}(\rho) = \alpha_k \qquad k = 0, \ldots, n. \tag{A.11}$$

Now the objective function is given by

$$F = \sum_{j=1}^{N} \int_0^r c_3 u_j^2(\rho) d\rho + \frac{1}{2}\delta f_0 + \delta \sum_{k=1}^{n-1} f_k + \frac{1}{2}\delta f_n \tag{A.12}$$

with

$$f_k = \left[S_{k,N}(r) - h(x_k)\right]^2.$$

References

[1] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[2] U. M. Ascher and L. R. Petzold, *The Numerical Solution of Delay-Differential-Algebraic Equations of Retarded and Neutral Type*, SIAM Journal on Numerical Analysis, 32 (1995), pp. 1635–1657.

[3] J. J. Batzel, S. Timischl-Teschl, and F. Kappel, *A cardiovascular-respiratory control system model including state delay with application to congestive heart failure in humans*, Journal of Mathematical Biology, 82 (2004), pp. 519–541.

[4] J. T. Betts, *Practical Methods for Optimal Control and Estimation using Nonlinear Programming*, Second Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA., 2010.

[5] J. T. Betts, N. Biehn, S. L. Campbell, and W. P. Huffman, *Compensating for Order Variation in Mesh Refinement for Direct Transcription Methods*, Journal of Computational and Applied Mathematics, 125 (2000), pp. 147–158.

[6] C. Büskens, L. Göllmann, and H. Maurer, *Optimal control of a stirred tank reactor with time delay*, in European Consortium of Mathematics in Industry, Sept. 1994.

[7] V. Deshmukh, H. Ma, and E. Butcher, *Optimal Control of Parametrically Excited Linear Delay Differential Systems via Chebyshev Polynomials*, in Proceedings of the American Control Conference, Denver, Colorado, June 2003.

[8] L. Göllmann, D. Kern, and H. Maurer, *Optimal control problems with delays in state and control variables subject to mixed control-state constraints*, Optimal Control Applications and Methods, (2008).

[9] V. Gretschko, *Theorie und Numerik optimaler Steuerprozesse mit Retardierung und Steuer- und Zustandsbeshränkung*, PhD thesis, Universität Münster, Nov. 2007.

[10] E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations I Nonstiff Problems*, Springer-Verlag, New York, New York, 1993.

[11] M. Landry, S. A. Campbell, K. Morris, and C. O. Aguilar, *Dynamics of an Inverted Pendulum with Delayed Feedback Control*, SIAM Journal on Applied Dynamical Systems, 4 (2005), pp. 333–351.

[12] H. Maurer, *Theory and applications of optimal control problems with control and state delays*, Adelaide, Sept.–Oct. 2009, 53rd Australian Mathematical Conference.

[13] C. A. Paul, *Designing Efficient Software for Solving Delay Differential Equations*, Tech. Rep. Numerical Analysis Report No. 368, Manchester Centre for Computational Mathematics, Department of Mathematics, University of Manchester, Manchester, M13 9PL, England, Oct. 2000.

[14] L. F. Shampine and P. Gahinet, *Delay-Differential-Algebraic Equations in Control Theory*, Applied Numerical Analysis, 56 (2005), pp. 574–588.

[15] R. F. Stengel, R. Ghigliazza, N. Kulkarni, and O. Laplace, *Optimal control of innate immune response*, Optimal Control Applications and Methods, 23 (2002), pp. 91–104.

[16] Z.-Q. Wang and L.-L. Wang, *A Legendre-Gauss Collocation Method for Nonlinear Delay-Differential Equations*, Discrete and Continuous Dynamical Systems Series B, 13 (2010), pp. 685–708.