

Adaptive Algorithms for Optimal Control of Time-Dependent Partial Differential-Algebraic Equation Systems

Radu Serban*
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
radu@llnl.gov

Shengtai Li[†] and Linda R. Petzold[‡]
Department of Computer Science
University of California Santa Barbara
shengtai@engineering.ucsb.edu
petzold@engineering.ucsb.edu

October 8, 2001

Abstract

This paper describes an adaptive algorithm for optimal control of time-dependent partial differential algebraic equation (PDAE) systems. A direct method based on a modified multiple shooting type technique and sequential quadratic programming (SQP) is used for solving the optimal control problem, while an adaptive mesh refinement (AMR) algorithm is employed to dynamically adapt the spatial integration mesh. Issues of coupling the AMR solver to the optimization algorithm are addressed. For time-dependent PDAEs which can benefit from the use of an adaptive mesh, the resulting method is shown to be highly efficient.

1 Introduction

In this paper we introduce COOPTAM, a software package with adaptive time and space integration for optimal control of time-dependent partial differential-algebraic equation (PDAE) systems. COOPTAM combines our previous work on optimal control of differential-algebraic equation (DAE) systems [6, 16, 17] and on adaptive mesh refinement (AMR) methods for 2-D PDAEs [13, 12], resulting in a powerful new tool for dynamic optimization. The optimal control problem is solved by a modified multiple shooting technique and sequential quadratic programming (SQP) [7]. Spatial adaptivity is achieved by the AMR algorithm, while time adaptivity is provided by the underlying DAE solver (DASPK3.0) [3, 11]. The overall structure of COOPTAM and the main interdependencies are shown in Fig. 1.

There is a large body of work on PDE-constrained optimization and optimal control, but most studies were done in the context of optimization of elliptic PDEs. Such problems were tackled in many different engineering areas, such as optimal control of steady Navier-Stokes flows [8, 4] and structural optimization [9, 5] to mention only a few. A few studies have considered optimal control of time-dependent PDEs [10, 17] but we are not aware of any work on optimal control of time-dependent PDEs with adaptivity in both space and time.

The target of the COOPTAM software is the optimal control of time-dependent PDAEs whose solutions exhibit steep fronts that move in time over the spatial domain. To resolve details around the moving front, the spatial discretization grid must be very fine around the front but can be significantly coarser far from the front. Since the computational complexity depends directly on the dimension of the spatial grid, using an adaptive mesh strategy during the time integration of such PDAEs greatly improves the overall performance of the optimal control software.

*This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

[†]This research was supported in part by NSF KDI ATM-9873133, NSF-ARPA PC 239415, NSF CCR 98-96198, and NSF ITR ACI 00-86061.

[‡]Correspondence to: Linda R. Petzold, Department of Computer Science, University of California, Santa Barbara, CA 93106

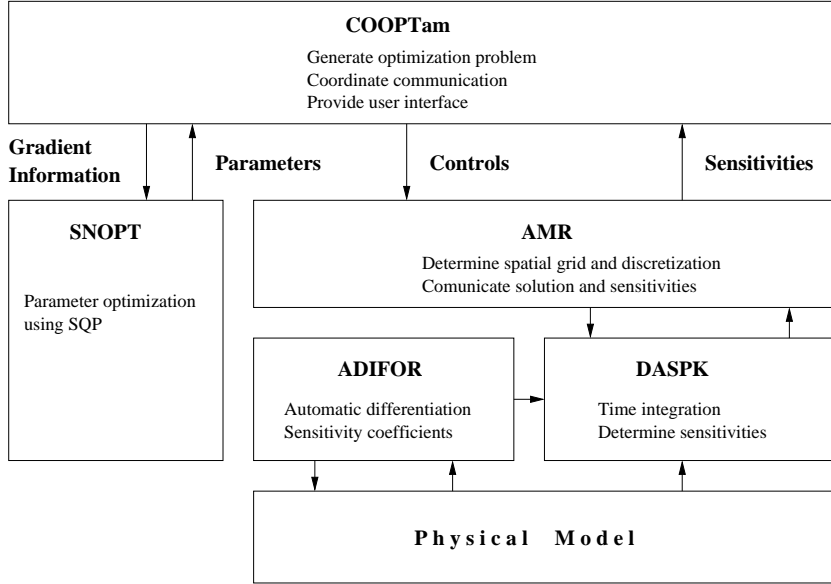


Figure 1: Structure of the COOPTAM code.

We begin in Section 1.1 by presenting the general class of problems under consideration. In Section 2 we give more details on the two basic components of COOPTAM: the modified multiple shooting method and SQP for dynamic optimization, and the adaptive mesh refinement algorithm for solution of the PDAEs. In Section 3 we focus on the technical issues of coupling these two techniques. We conclude in Section 4 by presenting numerical results which highlight the capabilities of our software as well as the efficiency improvements due to the use of the AMR method in conjunction with modified multiple shooting and SQP.

1.1 The optimal control problem

Consider $\Omega \subset \mathbf{R}^2$ and let $\mathbf{w}(t, \mathbf{x}) : [t_0, t_1] \times \Omega \rightarrow \mathbf{R}^{N_w}$ satisfy the time varying, spatial 2-D PDAE system

$$\mathbf{F}(t, \mathbf{x}, \mathbf{w}, \partial_t, \partial_{\mathbf{x}}, \mathbf{p}, \mathbf{u}(t)) = 0 \quad (1)$$

where $t \in [t_0, t_1]$ and $\mathbf{x} = [x_1, x_2]^T \in \Omega$, with initial conditions

$$\mathbf{w}(t, \mathbf{x}) = \mathbf{w}_0(\mathbf{x}), \quad t = t_0 \text{ and } \mathbf{x} \in \Omega, \quad (2)$$

and boundary conditions

$$\mathbf{w}(t, \mathbf{x}) = \mathbf{w}_{bc}(t), \quad t \in [t_0, t_1] \text{ and } \mathbf{x} \in \partial\Omega. \quad (3)$$

The control parameters \mathbf{p} and the vector-valued control function $\mathbf{u}(t)$ must be determined such that the scalar objective function

$$\int_{t_0}^{t_1} \psi(t, \mathbf{x}, \mathbf{w}(t, \mathbf{x}), \mathbf{p}, \mathbf{u}(t)) dt + \gamma(t_1, \mathbf{x}, \mathbf{w}(t_1, \mathbf{x}), \mathbf{p}, \mathbf{u}(t_1)) \quad (4)$$

is minimized and some additional equality and/or inequality constraints

$$\mathbf{g}(t, \mathbf{x}, \mathbf{w}(t, \mathbf{x}), \mathbf{p}, \mathbf{u}(t)) \gtrless 0$$

are satisfied. The optimal control function $\mathbf{u}^*(t)$ is assumed to be continuous.

There are two main approaches for solving optimal control problems of the type described above. If there are no time invariant parameters \mathbf{p} , the so-called *indirect method*, based on Pontryagin's maximum principle,

constructs the Hamiltonian and adjoint system corresponding to the optimal control problem (1)-(4) and then solves the resulting boundary value problem for the optimal control \mathbf{u}^* . In the *direct method* the controls \mathbf{u} are parameterized in a lower dimensional space and the resulting nonlinear programming problem is solved directly. Because generation of the adjoint system and solution of the resulting boundary value problem in the indirect method are difficult for general problems and because the direct method covers a larger class of problems of interest (such as dynamic optimization, where only time invariant parameters and no time varying controls appear in the model equations) we have implemented a direct approach.

As mentioned before, a direct approach for the optimal control problem employs a low order parameterization of the time varying controls, the coefficients of which are to be determined by an optimization algorithm. At each iteration of the optimizer, the model equations (1) are integrated in time and their solution is used in evaluating the current cost function (4). In COOPTAM we transform the system of PDAEs into a system of DAEs through semidiscretization in space. We do this using an adaptive mesh refinement (AMR) technique which is presented in more detail in Section 2.2. The variables in the resulting DAE are denoted by $\bar{\mathbf{w}}$. The dimension of $\bar{\mathbf{w}}$ can be very large. However, the dimensions of the parameters \mathbf{p} and of the representation of the control function $\mathbf{u}(t)$ are assumed here to be much smaller. To represent $\mathbf{u}(t)$ in a low-dimensional vector space, we use piecewise polynomials on $[t_0, t_1]$, their coefficients being determined by the optimization. For ease of presentation we can therefore assume that the vector \mathbf{p} contains both the parameters and these coefficients (we let N_p denote the combined number of these values).

Hence the problem is given by

$$\text{minimize} \quad \int_{t_0}^{t_1} \bar{\psi}(t, \bar{\mathbf{w}}(t), \mathbf{p}) dt + \bar{\gamma}(t_1, \bar{\mathbf{w}}(t_1), \mathbf{p}) \quad (5a)$$

$$\text{subject to} \quad \bar{\mathbf{F}}(t, \bar{\mathbf{w}}, \bar{\mathbf{w}}', \mathbf{p}) = 0, \quad \bar{\mathbf{w}}(t_0) = \bar{\mathbf{w}}_0, \quad (5b)$$

$$\text{and} \quad \bar{\mathbf{g}}(t, \bar{\mathbf{w}}(t), \mathbf{p}) \geq 0. \quad (5c)$$

2 Description of Methods

In this section we briefly describe the two basic components of our optimal control software, COOPTAM. First we discuss the modified multiple shooting method and SQP implemented in COOPTAM. Then we outline the adaptive mesh refinement algorithm employed in the integration of the underlying PDAE.

For more details on the optimization method, including control parameterization, additional bounds on controls, modifications of the multiple shooting method and the resulting optimization Jacobian, we encourage the reader to refer to [6, 16, 17]. Further details on the adaptive mesh refinement technique and its implementation are given in [13, 12].

2.1 Modified multiple shooting for dynamic optimization

There are a number of well-known methods for direct discretization of the optimization problem (5). The *single shooting method* solves the DAEs (5b) over the interval $[t_0, t_1]$, with the set of controls generated at each iteration by the optimization algorithm. However, it is well-known that single shooting can suffer from a lack of stability and robustness [1]. Moreover, for this method it is difficult to maintain additional constraints and to ensure that the iterates are physical or computable. The *finite-difference method* or *collocation method* discretizes the DAEs over the interval $[t_0, t_1]$. The DAE solutions at each discrete time and the set of controls are generated at each iteration by the optimization algorithm. Although this method is more robust and stable than the single shooting method, it requires the solution of an optimization problem which for a large-scale DAE system (such as those considered here) is enormous, and it does not allow for the use of adaptive DAE software.

We thus consider the *multiple-shooting method* for the discretization of (5). In this method, the time interval $[t_0, t_1]$ is divided into subintervals $[t_i, t_{i+1}]$ ($i = 1, \dots, N_{ms}$), and the differential equations (5b) are solved over each subinterval, where additional intermediate variables \mathbf{W}_i are introduced. On each subinterval we denote the solution at time t of (5b) with initial value \mathbf{W}_i at t_i by $\bar{\mathbf{w}}(t, t_i, \mathbf{W}_i, \mathbf{p})$. More details on the integration are given in Section 2.2.

Continuity between subintervals is achieved via the continuity constraints

$$\mathbf{C}_1^i(\mathbf{W}_{i+1}, \mathbf{W}_i, \mathbf{p}) \equiv \mathbf{W}_{i+1} - \bar{\mathbf{w}}(t_{i+1}, t_i, \mathbf{W}_i, \mathbf{p}) = \mathbf{0}.$$

The additional constraints (5c) are required to be satisfied at the boundaries of the shooting intervals

$$\mathbf{C}_2^i(\mathbf{W}_i, \mathbf{p}) \equiv \mathbf{g}(t_i, \mathbf{W}_i, \mathbf{p}) \succeq \mathbf{0}.$$

Following common practice, we write

$$\phi(t) = \int_{t_0}^t \bar{\psi}(\tau, \bar{\mathbf{w}}(\tau), \mathbf{p}) d\tau, \quad (6)$$

which satisfies $\phi'(t) = \psi(t, \bar{\mathbf{w}}(t), \mathbf{p})$, $\phi(t_0) = 0$. This introduces another equation and variable into the differential system (5b). The discretized optimal control problem becomes

$$\min_{\mathbf{W}_2, \dots, \mathbf{W}_{N_{ms}}, \mathbf{p}} \phi(t_1) + \bar{\gamma}(t_1) \quad (7)$$

subject to the constraints

$$\mathbf{C}_1^i(\mathbf{W}_{i+1}, \mathbf{W}_i, \mathbf{p}) = \mathbf{0}, \quad (8a)$$

$$\mathbf{C}_2^i(\mathbf{W}_i, \mathbf{p}) \geq \mathbf{0}. \quad (8b)$$

This problem can be solved by an optimization code. We use the solver SNOPT5.3 [7], which incorporates an SQP method. The SQP methods require a gradient and Jacobian matrix that are the derivatives of the objective function and constraints with respect to the optimization variables. We compute these derivatives via highly efficient DAE sensitivity software DASPK3.0 [11]. The sensitivity equations to be solved by DASPK3.0 are generated via the automatic differentiation software ADIFOR2.0 [2].

This basic multiple-shooting type of strategy can work very well for small-to-moderate size DAE systems, and has an additional advantage that it is inherently parallel. However, for large-scale DAE systems there is a problem because the computational complexity grows rapidly with the dimension of the DAE system. The difficulty lies in the computation of the derivatives of the continuity constraints (8a) with respect to the variables \mathbf{W}_i . The work to compute the derivative matrix $\partial \bar{\mathbf{w}}(t) / \partial \mathbf{W}_i$ is of order $O(N^2)$, and for the problems under consideration N can be very large (for example, for an DAE system obtained from the semi-discretization of a PDAE system, N is the product of the number of PDAEs and the number of spatial grid points). In contrast, the computational work for the single shooting method is of order $O(NN_p)$ although the method is not as stable, robust or parallelizable.

We reduce the computational complexity of the multiple shooting method for this type of problem by modifying the method to make use of the structure of the continuity constraints to reduce the number of sensitivity solutions which are needed to compute the derivatives. To do this, we recast the continuity constraints in a form where only the matrix-vector products $(\partial \bar{\mathbf{w}}(t) / \partial \mathbf{W}_i) \xi_j$ are needed, rather than the entire matrix $\partial \bar{\mathbf{w}}(t) / \partial \mathbf{W}_i$. The matrix-vector products are directional derivatives; each can be computed via a single sensitivity analysis. The number of vectors ξ_j such that the directional sensitivities are needed is small, of order $O(N_p)$. Thus the complexity of the modified multiple shooting computation is reduced to $O(NN_p)$, roughly the same as that of single shooting. Unfortunately, the reduction in computational complexity comes at a price: the stability of the modified multiple shooting algorithm suffers from the same limitations as single shooting. However, this is not an issue for many PDE systems, including the applications described here. This is due to the fact that optimal control is usually considered for problems which have already been simulated successfully and are thus stable from the left. We have found that the modified method is more robust than single shooting for nonlinear problems. Further details on the algorithm can be found in [6].

In the context of the SQP method, the use of modified multiple shooting involves a transformation of the constraint Jacobian. The affected rows are those associated with the continuity constraints and any path constraints applied within the shooting intervals. Path constraints enforced at the shooting points (and other constraints involving only discretized states) are not transformed. The transformation is cast almost entirely at the user level and requires minimal changes to the optimization software, which is important because software in this area is constantly being modified and improved. Gill et al. ([6]) have shown that the modified quadratic subproblem yields a descent direction for the ℓ_1 penalty function. DAOPT is a modification to the SNOPT5.3 optimization code that uses a merit function based on an ℓ_1 penalty function.

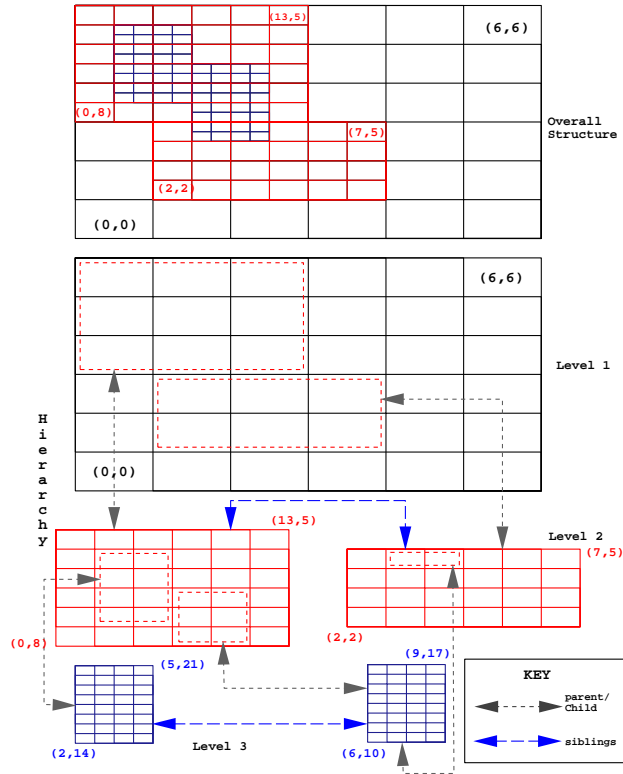


Figure 2: The hierarchical data structure for SAMR.

2.2 Adaptive mesh refinement

A method of lines (MOL) approach was employed to solve the PDAE on an adaptive mesh. First the PDAE system is semi-discretized in space to form a DAE system. Then the DAE system is integrated by a variable order variable step-size backward differentiation formula (BDF) method. The structured adaptive mesh refinement (SAMR) method was used to achieve adaptivity in space when the mesh varies with time. SAMR uses a hierarchical block data structure where each block (called a *patch*) can be solved as a single grid. A refinement can take place at each time step or after a number of time steps. During the refinement, the monitor function is evaluated on the current level and the points with significant errors are flagged. After buffer points are added around these points, they are clustered and organized into patches. The refinement is done recursively until the finest level is reached or there are no flagged points. An example of a 2-D hierarchical grid is shown in Fig. 2.

Most implementations of SAMR have used explicit time integration, and refined time as well as space by taking local smaller time steps for finer grids. Explicit time integration is not efficient when solving some parabolic type PDEs or when solving for the solutions of near steady-state equations. In [13], SAMR was combined with an implicit time integration solver `DASPK3.0` [12], which also has the sensitivity analysis capability. An efficient transformation between the `DASPK3.0` flat structure and the AMR hierarchical data structure was designed and applied in [13].

After a new grid is generated and the solution has been interpolated from the old mesh to the new one, the simplest approach would be to restart the time integrator as though solving a new problem. This is called a *full restart* and is appropriate for single-step time integration methods. For multistep methods, a full restart would cause the ODE/DAE solver to choose the lowest order single-step method and to reduce the time step size to satisfy the error tolerance of the lowest order method. Therefore, we use a *warm restart* in our implementation [13]. The history array used by the ODE/DAE solver is interpolated to the new mesh, and the integration is continued with almost the same step size and order as would have been used had the remeshing not taken place.

Even with the warm restart, the overhead of the mesh adaptation is relatively high. The most significant

cost is evaluation of the Jacobian. To further reduce the overhead of the mesh adaptation, we perform the mesh refinement after a number of time steps instead of at every time step, and replace the old adaptive mesh with the new one only when the variance is large enough. An automatic differentiation tool is used to reduce the cost and to increase the reliability of the Jacobian evaluation.

There are two possibilities for the sensitivity analysis for a PDE system. First, we can use the MOL approach and transform the PDE system into an ODE/DAE system. Then the sensitivity methods in DASPK3.0 can be used. The sensitivity equations can be evaluated by several options in DASPK3.0, such as the finite-difference or ADIFOR2.0 options. This approach does not require any modification of the PDE discretization codes. For these reasons, this is the approach implemented in COOPTAM. The other approach is to solve the sensitivity PDEs coupled with the original physical PDEs directly. They are simultaneously discretized in space and then the coupled ODE/DAE system is solved by DASPK3.0.

For the adaptive grid solver, a decision must be made on whether the selection of mesh refinement should be based only on the state PDEs or on both the state and sensitivity PDEs. We observed in our applications that the sensitivity PDEs shared the same refinement regions as the state PDEs. Therefore, only the state PDEs are used in our mesh refinement. However, both the state and sensitivity PDEs are interpolated from the old mesh to the new one if the new mesh is adopted. The details on how sensitivity variables are stored in the AMR hierarchical structure and transformed to the DASPK3.0 flat linear structure are described in [13].

3 Coupling Issues

Difficulties in coupling the optimal control software with the adaptive mesh refinement integrator arise from the discrepancies in the spatial meshes that are used by the optimizer and the integrator. On one hand, the optimizer must be provided a fixed-size problem which means that information passed to the optimizer (constraint and objective functions and their derivatives) and information provided by the optimizer (current estimates of parameters, control parameters, and states at the multiple shooting interface) relate to a fixed spatial mesh, henceforth called the *optimization mesh* (OM). On the other hand, the adaptive mesh algorithm has the freedom of changing the mesh during integration over any given shooting interval. This can involve both refinement and coarsening of the spatial mesh. We call this changing mesh the *integration mesh* (IM). Besides these two meshes, we introduce the *user mesh* (UM) which is the initial mesh provided by the user. Usually, the UM provides only the first level in the mesh hierarchy. However, the UM can enforce refinement in certain zones of the spatial domain, where either the problem is known to require it, or where more details are needed in the evaluation of the cost functional. For an example of the latter case see Section 4.

There are three major coupling issues: (1) optimizer-integrator mesh correlation, (2) recovery of sensitivities on the OM, and (3) computation of the cost functional on hierarchical mesh structures.

Optimizer-integrator mesh correlation. As mentioned before, the main problem comes from the fact that we must provide a fixed-size problem to the optimizer. Our goal is to generate the OM in such a way that it captures relevant details at the multiple shooting interfaces since these are the only points in time where communication between the optimizer and the integrator takes place. At these points, the optimizer provides initial values for the integration over the following shooting interval, while the integrator provides final solutions and sensitivities over the previous shooting interval. As a final consideration, we must realize that the mesh refinement history during integration over any given shooting interval depends on the current estimates of the parameters and controls. As a consequence, an OM generated using only the initial guess for parameters and controls may not capture the necessary details at the optimal solution. To overcome these problems we have developed a staggered optimization strategy. In *stage 1*, we obtain a first approximation of the optimal parameters and controls by performing an optimization on a fixed mesh, during which the UM provided initially by the user is used as both OM and IM. During this first stage, the adaptivity in space of the PDAE integrator is turned off. In *stage 2*, with the parameters and controls obtained at the end of stage 1, the PDAEs are integrated on an adaptive mesh. The mesh refinement history is stored, and the OM is generated as the reunion of the IMs at the end of all shooting intervals. Finally, in *stage 3* a final optimization on the adaptive mesh is performed. For reasons explained next, during this final stage the integration mesh, although allowed to change, is enforced to always contain the OM generated in stage 2.

Recovery of sensitivities on the Optimization Mesh. The second difficulty arises from the fact that sensitivities with respect to some initial conditions cannot be computed from sensitivities with respect to neighboring initial conditions. This could become an issue if the AMR algorithm decides to coarsen the grid during the integration. Since the integrator is restarted whenever a change in the IM is performed, the sensitivities with respect to initial conditions corresponding to the discarded nodes are lost. The values of these sensitivities at the end of the shooting interval cannot be computed a posteriori from sensitivities with respect to initial conditions corresponding to nodes that were computed during the integration over the shooting interval. Therefore we impose that at all times during integration over a given shooting interval, $OM \subseteq IM$. In other words, coarsening of the OM is not allowed.

Computation of the cost functional. Finally, a hierarchical mesh structure necessarily stores information for the same physical point in several grid levels. As a consequence, the same point could potentially be included more than once in the evaluation of the cost functional. To overcome this problem, we introduce a coloring of the nested levels such that only values from the most refined level are used in evaluating the cost functional.

4 Numerical Results

To illustrate the use of COOPTAM we consider the following PDE dependent on the control $u(t)$

$$w_t + \frac{1}{2}\nabla w^2 - r(x, y, t)\nabla^2 w = 0, \quad (9)$$

with $r(x, y, t) = C(x - x_m)(x - x_M)(y - y_m)(y - y_M)u(t) + r_0$. This test problem, a modification of Burger's equations, was chosen because its solution, like those in applications targeted by our software, exhibits a steep moving front and hence its efficient solution requires an adaptive method. The domain Ω is given as

$$\Omega = \{(x, y) \mid x_m \leq x \leq x_M; y_m \leq y \leq y_M\}$$

and $t \in [0, t_f]$. In the numerical computations presented below, we have used $\Omega = [0, 0.5] \times [0, 0.5]$, $t \in [0, 0.9]$, $C = 50.0$ and $r_0 = 0.005$. With the constraint $u(0) = 0$, boundary conditions and initial values are obtained from the analytical solution for $r = r_0$,

$$\bar{w}(x, y, t) = \frac{1}{1 + \exp[(x + y - t)/(2r_0)]}$$

as

$$\begin{aligned} \text{B.C.} \quad w(x, y, t) &= \bar{w}(x, y, t), \text{ for } (x, y) \in \partial\Omega \text{ and } \forall t \in [0, t_f] \\ \text{I.C.} \quad w(x, y, 0) &= \bar{w}(x, y, 0), \text{ for } (x, y) \in \Omega. \end{aligned} \quad (10)$$

Note that, if the control function $u(t) = 0$, the solution of (9) is a steep front that moves in time from the lower left corner towards the upper right corner. At any time, far from this moving front (both behind it and ahead of it) the solution is very flat. To resolve the details of the solution, the integration grid must be very fine around the front but can be significantly coarser far from it. Since the front moves in time, an adaptive mesh strategy is desirable during the integration.

The optimal control u^* and corresponding optimal solution $w^*(x, y, t)$ must be determined such that the solutions at specified points in Ω follow prescribed time trajectories. The cost function can thus be written as

$$J(u) = \sum_i^M \int_0^{t_f} [w(x_i, y_i, t) - w_i(t)]^2 dt \quad (11)$$

which must be minimized subject to the dynamical model (9) with boundary conditions and initial conditions (10) and the additional constraint $u(0) = 0$.

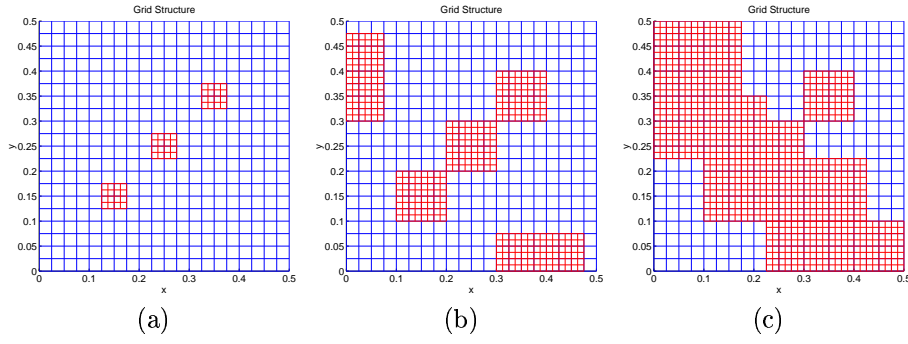


Figure 3: (a) UM grid; (b) OM grid generated after step 1; (c) OM grid generated without step 1

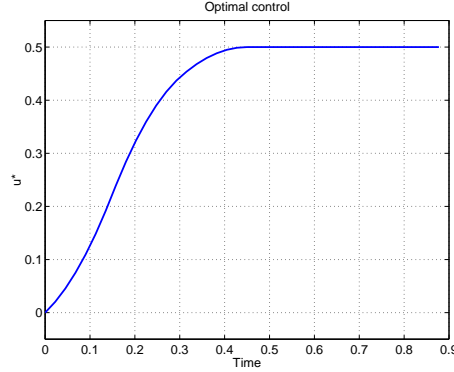


Figure 4: Optimal control

Using $N_{ms} = 2$ shooting intervals and $N_u = 3$ control subintervals per shooting interval, a minimal dimension piecewise quadratic polynomial control parameterization requires $N_p = 10$ parameters. We considered the following $M = 3$ trajectories w_i in the cost function:

$$\begin{aligned}
 w_1(0.15, 0.15, t) &= \begin{cases} 0 & \text{if } t \leq 0.1 \\ (t - 0.1)/0.4 & \text{if } 0.1 < t < 0.5 \\ 1 & \text{if } t \geq 0.5 \end{cases} \\
 w_2(0.25, 0.25, t) &= \begin{cases} 0 & \text{if } t \leq 0.2 \\ 0.9(t - 0.2)/0.5 & \text{if } 0.2 < t < 0.7 \\ 0.9 & \text{if } t \geq 0.7 \end{cases} \\
 w_3(0.35, 0.35, t) &= \begin{cases} 0 & \text{if } t \leq 0.3 \\ 0.7(t - 0.3)/0.6 & \text{if } 0.3 < t < 0.9 \end{cases}
 \end{aligned} \tag{12}$$

The initial spatial mesh (UM) contains 515 grid points structured on two levels, with the second level providing more grid nodes around the points where trajectories are imposed (see Fig. 3-(a)). Starting with an initial guess $u_0(t) = 0$, the staggered optimization procedure described in Section 3 was employed to reduce the cost function from $J(u_0) = 1.185$ to $J(u_1^*) = 0.126$ after 12 optimization iterations in stage 1 and then to $J(u^*) = 0.118$ after 5 optimization iterations in stage 3. In stage 2, the optimization mesh (OM) was generated and contained 755 grid points (see Fig. 3-(b)). The largest number of grid points in any of the integration meshes was 1168.

The optimal control u^* is shown in Fig. 4. Figure 5-(a) shows the optimal solution $w(x, y)$ at $t = 0.45$ and is presented here to explain the structure of the optimization mesh OM. In Fig. 5-(b) we present trajectories at the three points considered in (12) with solid lines representing the desired trajectories w_i ,

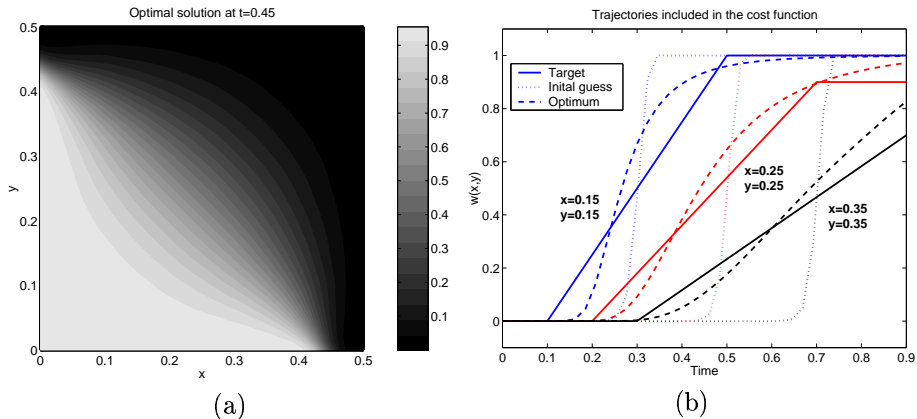


Figure 5: (a) Optimal solution at the shooting interface $t=0.45$; (b) Trajectories considered in the cost function

Table 1: Comparison of efficiency of different strategies. CPU times were obtained on a PC workstation with an Intel Pentium III 800 MHz processor running Linux 2.2.12. The code was compiled with `gcc` with first level optimization.

	Staggered optimization		Fixed mesh optimization
	Stage 1	Stage 2	
OM size	515	755	1681
Initial c.f.	1.185	0.126	1.185
Optimal c.f.	0.126	0.118	0.122
Optimization iterations	12	5	34
CPU time (s)	67.75	227.24	2461.10

dotted lines representing trajectories obtained with the initial guess control, and dashed lines corresponding to the optimal solution.

We note that the first optimization stage is really important. If the optimization mesh OM were generated using the initial guess control ($u_0(t) = 0$ in this example) it may contain far more grid points than actually necessary close to the optimal solution. Figure 3-(c) presents such an OM which contains 1152 grid points (we remind that the largest IM contains only 1168).

Finally, in order to estimate the computational savings due to the use of AMR, we have solved the optimal control problem on a mesh that had second level refinement everywhere. During this computation, the mesh was held fixed (UM=OM=IM) and contained 1681 grid points. The optimal solution was obtained after 34 iterations with a final cost function of $J(u^*) = 0.122$. Moreover, this computation was more than 8 times slower than the staggered optimization procedure used above. Results are summarized in Table 1.

5 Conclusions and Future Work

We have presented methods and software for the optimal control of large scale dynamical models described by systems of time-dependent partial differential-algebraic equations (PDAE). Our software COOPTAM combines the use of efficient numerical methods for solving the model equations and the required sensitivity equations (DASPK3.0 with AMR) with a package for large-scale optimization based on sequential quadratic programming (SNOPT5.3). COOPTAM implements a modified multiple shooting method for the resulting dynamic optimization problem which improves efficiency in the optimization Jacobian evaluation. We have described the underlying methods in COOPTAM for coupling these two powerful technologies and demon-

strated their effectiveness for a PDE problem.

Future work will focus on implementation of adjoint sensitivity analysis for optimal control, as well as on exploring the possibility of automatically generating the sensitivity PDEs, as opposed to the current implementation where we compute sensitivities of the semidiscretized PDE.

References

- [1] ASCHER, U.M., MATTHEIJ, R.M., AND RUSSELL, R.D. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. SIAM, Philadelphia, PA, 1995.
- [2] BISCHOF, C., CARLE, A., CORLISS, G., GRIEWANK, A., AND HOVLAND, P. ADIFOR - Generating Derivative Codes from Fortran Programs. *Scientific Programming 1* (1992), 11–29.
- [3] BRENNAN, K.E., CAMPBELL, S.L., AND PETZOLD, L.R. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, Philadelphia, PA, 1995.
- [4] GHATTAS, O. AND BARK, J.-H. Optimal Control of Two- and Three-Dimensional Incompressible Navier-Stokes Flows. *Journal of Computational Physics 136* (1997), 231–244.
- [5] GHATTAS, O. AND OROZCO, C. A parallel reduced Hessian SQP method for shape optimization. In *Multidisciplinary Design Optimization: State of the Art* (1997), Alexandrov, N.M. and Hussaini, M.Y., Ed., SIAM, pp. 133–152.
- [6] GILL, P.E., JAY, L.O., LEONARD, M.W., PETZOLD, L.R., AND SHARMA, V. An SQP Method for the Optimal Control of Large-Scale Dynamical Systems. *J. Comp. Appl. Math. 20* (2000), 197–213.
- [7] GILL, P.E., MURRAY, W., AND SAUNDERS, M.A. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. Tech. Rep. 97-2, Department of Mathematics, University of California, San Diego, La Jolla, CA, 1997.
- [8] GUNZBURGER, M.D., HOU, L.S., AND SVOBODNY, T.P. Optimal Control and Optimization of Viscous, Incompressible Flows. In *Incompressible Computational Fluid Dynamics* (1993), Gunzburger, M.D. and Nicolaides, R.A., Ed., pp. 109–150.
- [9] HAUG, E.J. AND ARORA, J.S. *Applied Optimal Design*. John Wiley and Sons, New York, NY, 1979.
- [10] HE, B., GHATTAS, O., AND ANTAKI, J.F. Computational Strategies for Shape Optimization of Navier-Stokes Flows. Tech. Rep. CMU-CML-97-102, Computational Mechanics Lab, Department of Civil and Environmental Engineering, Carnegie Mellon University, 1997.
- [11] LI, S. AND PETZOLD, L.R. Design of New DASPK for Sensitivity Analysis. Tech. rep., Department of Computer Science, University of California, Santa Barbara, CA, 1999.
- [12] LI, S. AND PETZOLD, L.R. Software and Algorithms for Sensitivity Analysis of Large-Scale Differential-Algebraic Systems. *J. Comp. Appl. Math. 13* (2000), 131–145.
- [13] LI, S., PETZOLD, L.R., AND HYMAN, J.M. Solution Adapted Mesh Refinement and Sensitivity Analysis for Parabolic Partial Differential Equation Systems. Submitted to Special volume of Lecture Notes in Computational Science and Engineering, Springer, 2001.
- [14] MALY, T. AND PETZOLD, L.R. Numerical Methods and Software for Sensitivity Analysis of Differential-Algebraic Systems. *Applied Numerical Mathematics 20* (1997), 57–79.
- [15] PETZOLD, L.R., ROSEN, J.B., GILL, P.E., JAY, L.O., AND PARK, K. Numerical Optimal Control of Parabolic PDEs using DASOPT. In *Large Scale Optimization with Applications, Part II: Optimal Design and Control* (1997), Biegler, L., Coleman, T., Conn, A., and Santosa, F., Ed., vol. 93 of *IMA Volumes in Mathematics and its Applications*, pp. 271–300.

- [16] SERBAN, R. COOPT - Control and Optimization of Dynamic Systems - Users' Guide. Tech. Rep. UCSB-ME-99-1, Department of Mechanical and Environmental Engineering, University of California, Santa Barbara, CA, 1999.
- [17] SERBAN, R. AND PETZOLD, L.R. COOPT - A Software Package for Optimal Control of Large-Scale Differential-Algebraic Equation Systems. *J. Math. Comp. Sim.* 56(2) (2001), 187–203.