A REDUCED SQP ALGORITHM FOR THE OPTIMAL CONTROL OF SEMILINEAR PARABOLIC EQUATIONS

Roland Griesse
Lehrstuhl für Ingenieurmathematik
Universität Bayreuth, Germany
roland.griesse@uni-bayreuth.de

Abstract This paper deals with optimal control problems for semilinear time-dependent partial differential equations. Apart from the PDE, no additional constraints are present. Solving the necessary conditions for such problems via the Newton-Lagrange method is discussed. Motivated by issues of computational complexity and convergence behavior, the Reduced Hessian SQP algorithm is introduced. Application to a system of reaction-diffusion equations is outlined, and numerical results are given to illustrate the performance of the reduced Hessian algorithm.

Keywords: optimal control, parabolic equation, semilinear equation, reduced SQP method, reaction-diffusion equation

Introduction

There exist two basic classes of algorithms for the solution of optimal control problems governed by partial differential equations (PDEs). They both are of an iterative fashion and are different in that *Newton-type* methods require the repeated solution of the (non-linear) PDE while the algorithms of *SQP-type* deal with the linearized PDE only. Newton-type methods have been successfully applied, e.g., to control problems for the Navier-Stokes equations in [4] and will not be discussed here.

The main focus of this paper is on SQP-type methods which basically use Newton's algorithm in order to solve the first order necessary conditions. This scheme leads to a linear boundary value problem for the state and adjoint variables. It is the size of the *discretized* linear boundary value problem that motivates a variant of this approach in the first place: The reduced SQP method, which has been the subject of the following papers: [5] introduces reduced Hessian methods in Hilbert

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35699-0_19
E. W. Sachs et al. (eds.), *System Modeling and Optimization XX*

spaces. [4] studies various second-order methods for optimal control of the time-dependent Navier-Stokes equations. [2] and [3] discuss algorithms based on inexact factorization of the full Hessian step (11) which involve the reduced Hessian (or approximations thereof) in the factors. [1] examines preconditioners for the KKT matrices arising in interior point methods, also using reduced Hessian techniques.

This paper is organized as follows: In Section 1, the class of semilinear second order parabolic partial differential equations is introduced with control provided in distributed fashion. Section 2 covers optimal control problems for these PDEs and establishes the first order necessary conditions. Section 3 describes the basic SQP method in function spaces (also called the *Newton-Lagrange method* in this context), that can be used to solve these conditions. The reduced Hessian method is derived as a variant thereof. It will be seen that this method is applicable only if the *linearized* PDE is uniquely solvable with continuous dependence on the right hand side data. The purpose of the reduced Hessian method is to significantly decrease the size of the discretized SQP steps. The associated algorithm which requires the repeated solution of the linearized state equation and of the corresponding adjoint is presented in detail. In Section 4, this procedure is applied to a system of reaction-diffusion PDEs. Finally, numerical results are given in Section 5.

While the ideas and algorithm are worked out for distributed control problems throughout this paper, boundary and mixed control problems can be treated in the very same manner with only minor modification of notation.

1. Semilinear Parabolic Equations

Let Ω be a bounded domain in $\mathbb{R}^2$ with sufficiently smooth boundary Γ and $Q = \Omega \times (0,T)$, $\Sigma = \Gamma \times (0,T)$ with given final time $T > 0$. We consider semilinear parabolic initial-boundary value problems of the following type:

$$\begin{aligned} y_t(x,t) + A(x)y(x,t) + n(x,t,y(x,t),u(x,t)) &= 0 \quad \text{in} \quad Q \\ \partial_n y(x,t) + b(x,t,y(x,t)) &= 0 \quad \text{on} \quad \Sigma \\ y(x,0) - y_0(x) &= 0 \quad \text{on} \quad \Omega. \end{aligned} \tag{1}$$

The elliptic differential operator $A(x)y = -\sum_{i,j=1}^{2} D_j(a_{ij}(x)D_i y)$ is represented by the matrix $\bar{A}(x) = (a_{ij}(x)) \in \mathbb{R}^{2\times 2}$ which is assumed to be symmetric, and $\partial_n y(x,t) = n(x)^T \bar{A}(x) \nabla y(x,t) = \sum_{i,j=1}^{2} a_{ij} n_i(x) D_j y(x,t)$ is the so-called *co-normal derivative* along the boundary Γ. When A is the negative Laplace operator $-\Delta$, $\bar{A}$ gives the identity matrix and $\partial_n y(x,t)$ is simply the normal derivative or Neumann trace of $y(x,t)$.

Questions of solvability, uniqueness and regularity for non-linear PDEs shall not be answered here. Please refer to [7] and the references cited therein. We assume that there exist Banach spaces Y for the state, U for the control and Z for the adjoint variable such that the semilinear parabolic problem (1) is well-posed in the abstract form

$$e(y,u) = 0 \quad \text{with} \quad e : Y \times U \to Z' \tag{2}$$

where Z' is the dual space of Z. The operator e may represent a *strong* or *weak* form of the state equation (1). Casting the PDE in this convenient form will allow us later to view the control problem as a PDE-constrained optimization problem and hence support a solution approach based on the Lagrange functional. However, in the detailed presentation of the algorithms, we will return to interpreting the operator e and its linearization e_y as time-dependent PDEs.

2. Optimal Control Problems

In the state equation (1), the function u defined on Q is called the *distributed* control function. A *Neumann boundary control* problem arises when, instead of u, a control function v is present in the boundary nonlinearity $b(x,t,y(x,t),v(x,t))$. Other possibilities include *Dirichlet boundary control* or even combinations of all of the above. Examples of boundary control problems can be found, e.g., in [3] and [1]. Everything presented in this paper can be and in fact has been applied to boundary control problems with only minor modifications.

The core of optimal control problems is to choose the control function $u \in U$ in order to minimize a given objective function. In practical terms, the objective can, e.g., aim at energy minimization or tracking a given desired state.

We shall use the *objective* for the distributed control case from [7]:

$$f(y,u) = \int_\Omega \varphi(x, y(x,T))\, dx + \int_Q g(x,t,y,u)\, dx\, dt \tag{3}$$

where φ asseses the terminal state and g evaluates the distributed control effort and the state trajectory in $(0,T)$.

The abstract optimal control problem considered throughout the rest of this paper can now be stated:

$$\begin{array}{lll} \text{Minimize} & f(y,u) & \text{over} \quad (y,u) \in Y \times U \\ \text{s.t.} & e(y,u) = 0 & \text{holds.} \end{array} \tag{4}$$

A particularly simple situation arises when the state equation (1) is in fact linear in (y,u) and the objective (3) is convex or even quadratic

positive definite. However, in the general case, our given problem (4) to find an optimal control u and a corresponding optimal state y minimizing (3) while satisfying the state equation $e(y,u) = 0 \in Z'$ is a non-convex problem. We will not address the difficult question of global optimal solutions but rather assume that a local optimizer $(\hat{y}, \hat{u})$ exists. The following *first order necessary conditions* involving the adjoint variable λ are well-known, see, e.g., [7] (with $-\lambda$ instead of λ):

$$\begin{aligned} -\lambda_t + A(x)^*\lambda + n_y(x,t,y,u)\lambda + g_y(x,t,y,u) &= 0 \quad \text{in} \quad Q \\ \partial_n\lambda + b_y(x,t,y)\lambda &= 0 \quad \text{on} \quad \Sigma \\ \lambda(T) + \varphi_y(x,y(T)) &= 0 \quad \text{in} \quad \Omega \\ g_u(x,t,y,u) + n_u(x,t,y,u)\lambda &= 0 \quad \text{in} \quad Q \\ y_t + A(x)y + n(x,t,y,u) &= 0 \quad \text{in} \quad Q \\ \partial_n y + b(x,t,y) &= 0 \quad \text{on} \quad \Sigma \\ y(0) - y_0(x) &= 0 \quad \text{on} \quad \Omega. \end{aligned} \tag{5}$$

These can be derived by constructing the *Lagrangian*

$$L(y,u,\lambda) = f(y,u) + \langle e(y,u), \lambda \rangle_{Z',Z} \tag{6}$$

and evaluating the conditions

$$L_y(y,u,\lambda) = 0 \quad \text{in} \quad Y' \quad \textit{(adjoint equation)} \tag{7}$$

$$L_u(y,u,\lambda) = 0 \quad \text{in} \quad U' \quad \textit{(optimality condition)} \tag{8}$$

$$L_\lambda(y,u,\lambda) = e(y,u) = 0 \quad \text{in} \quad Z' \quad \textit{(state equation)} \tag{9}$$

in their strong form.

Triplets $(\hat{y}, \hat{u}, \hat{\lambda})$ that satisfy the first order necessary conditions are called *stationary points*. Obviously, the conditions (5) or (7)–(9) constitute a non-linear two-point boundary value problem involving the non-linear *forward equation* (initial values given) for the state y and the linear *backward equation* (terminal conditions given) for the adjoint λ. In the next section we introduce an algorithm to solve this problem.

3. SQP Algorithms

As we have seen in the previous section, finding stationary points $(\hat{y}, \hat{u}, \hat{\lambda})$ and thus candidates for the optimal control problem requires the solution of the non-linear operator equation system (7)–(9). This task can be attacked by Newton's method that is commonly used to find zeros of non-linear differentiable functions.

Suppose that we are given a triplet (y^k, u^k, λ^k), the current iterate. The Newton step to compute updates $(\delta y, \delta u, \delta\lambda)$ reads

$$\begin{bmatrix} L_{yy}(y^k,u^k,\lambda^k) & L_{yu}(y^k,u^k,\lambda^k) & e_y(y^k,u^k)^* \\ L_{uy}(y^k,u^k,\lambda^k) & L_{uu}(y^k,u^k,\lambda^k) & e_u(y^k,u^k)^* \\ e_y(y^k,u^k) & e_u(y^k,u^k) & 0 \end{bmatrix} \begin{bmatrix} \delta y \\ \delta u \\ \delta\lambda \end{bmatrix} = - \begin{bmatrix} L_y(y^k,u^k,\lambda^k) \\ L_u(y^k,u^k,\lambda^k) \\ e(y^k,u^k) \end{bmatrix}$$

$$= - \begin{bmatrix} f_y(y^k,u^k) + e_y(y^k,u^k)^*\lambda^k \\ f_u(y^k,u^k) + e_u(y^k,u^k)^*\lambda^k \\ e(y^k,u^k) \end{bmatrix} \text{ in } \begin{bmatrix} Y' \\ U' \\ Z' \end{bmatrix}. \tag{10}$$

This method is referred to as the *Newton-Lagrange* algorithm. It falls under the category of SQP solvers since (10) are also the necessary conditions of an auxiliary QP problem, see, e.g., [6]. Note that in contrast to the so-called *Newton approach* (cf. [4]), the iterates (y^k, u^k) of the SQP method are *infeasible* w.r.t. the non-linear state equation, i.e. the method generates control/state pairs that satisfy the PDE (1) only in the limit.

The operators appearing in the matrix on the left hand side (the *Hessian of the Lagrangian*) deserve some further explanation. First it is worth recalling that the first partial Fréchet derivative of a mapping $g : X_1 \times X_2 \to Y$ between normed linear spaces $X = X_1 \times X_2$ and Y at a given point $x = (x_1, x_2) \in X$ is a bounded linear operator, e.g., $g_{x_1}(x) \in \mathcal{L}(X_1, Y)$. Consequently, the second partial Fréchet derivatives at x are $g_{x_1x_1}(x) \in \mathcal{L}(X_1, \mathcal{L}(X_1, Y))$, $g_{x_1x_2}(x) \in \mathcal{L}(X_2, \mathcal{L}(X_1, Y))$, etc. They can equivalently be viewed as bi-linear bounded operators, e.g., the latter taking its first argument from X_2 and its second from X_1 and mapping this pair to an element of Y.

The *adjoint operators* (or, precisely speaking, conjugate operators) appearing in the equation (10) can most easily be explained by their property of switching the arguments' order in bilinear maps:

$$\begin{aligned} e_y(y^k,u^k) &\in \mathcal{L}(Y, Z') \\ e_y(y^k,u^k)^* &\in \mathcal{L}(Z'', Y') \hookrightarrow \mathcal{L}(Z, Y') \quad \text{since} \quad Z \hookrightarrow Z'' \\ e_y(y^k,u^k)^*(z,y) &= e_y(y^k,u^k)(y,z) \qquad\qquad \text{for all} \quad y \in Y, z \in Z. \end{aligned}$$

Exploiting the fact that the adjoint variable λ appears linearly in the Lagrangian L, the Newton step (10) can be rewritten in terms of the new iterate λ^{k+1} rather than the update $\delta\lambda$. For brevity, the arguments (y^k, u^k, λ^k) will be omitted from now on:

$$\begin{bmatrix} L_{yy} & L_{yu} & e_y^* \\ L_{uy} & L_{uu} & e_u^* \\ e_y & e_u & 0 \end{bmatrix} \begin{bmatrix} \delta y \\ \delta u \\ \lambda^{k+1} \end{bmatrix} = - \begin{bmatrix} f_y \\ f_u \\ e \end{bmatrix}. \tag{11}$$

As can be expected, this system (obtained by linearization of (7)–(9)) represents a *linear* two-point boundary value problem whose solution is now the main focus.

To render problem (11) amenable for computer treatment, some discretization has to be carried out. Inevitably, its discretized version will be a large system of linear equations since it ultimately contains the values of the state, control and adjoint at all discrete time steps and all nodes of the underlying spatial grid. Thus, one seeks to minimize the dimension of the system by decomposing it into smaller parts. The *reduced Hessian algorithm* is designed just for this purpose:

Roughly speaking, it solves the linear operator equation (11) for δu first, using Gaussian elimination on the symbols in the matrix. A prerequisite to this procedure is the bounded invertibility of $e_y(y,u)$ for all (y,u) which are taken as iterates in the course of the algorithm. In other words, the linearized state equation $e_y(y,u)h = f$ has to be uniquely solvable for h (with continuous dependence on the right hand side $f \in Z'$) at these points (y,u). One obtains the reduced Hessian step

$$\begin{aligned} \left(e_u^* e_y^{-*} L_{yy} e_y^{-1} e_u + L_{uu} - L_{uy} e_y^{-1} e_u - e_u^* e_y^{-*} L_{yu}\right) \delta u & \\ = e_u^* e_y^{-*} \left[f_y - L_{yy} e_y^{-1} e\right] - f_u + L_{uy} e_y^{-1} e & \qquad (12) \\ e_y\, \delta y = -e - e_u \delta u & \qquad (13) \\ e_y^*\, \lambda^{k+1} = -f_y - L_{yy}\delta y - L_{yu}\delta u. & \qquad (14) \end{aligned}$$

The operator preceding δu is called the *reduced Hessian* $H_{\delta u}$ in contrast to the *full Hessian* matrix H appearing in (11). Note that both the full and the reduced Hessian are self-adjoint operators. After discretization, the reduced Hessian will be small and dense, whereas the full Hessian will be large and sparse. Aiming at solving a discretized version of (12) using an iterative solver, the action of the reduced Hessian on given elements $\delta u \in U$ has to be computed, plus the right hand side of (12). It can be shown that once an approximate solution δu to (12) is found, the remaining unknowns δy and λ^{k+1} obeying (13) and (14) can be expressed in terms of quantities already calculated. The overall procedure to solve (7)–(9) applying the reduced Hessian method on the inner loop decomposes nicely into the steps described in figure 1 using the auxiliary variables $h_1, h_3 \in Y$ and $h_2, h_4 \in Z$.

In many practical cases, the objective and the PDE separate as

$$f(y,u) = f_1(y) + f_2(u) \quad \text{and} \quad e(y,u) = e_1(y) + e_2(u) \qquad (15)$$

which entails $L_{uy} = L_{yu} = 0$.

We observe that for the computation of the right hand side b as well as for every evaluation of $H_{\delta u}\Box$, it is required to solve one equation in-

Reduced SQP Algorithm

1 Set $k = 0$ and initialize (y^0, u^0, λ^0).

2 Solve

(a) $e_y h_1 = e$

(b) $e_y^* h_2 = f_y - L_{yy} h_1$

and set $b := e_u^* h_2 - f_u + L_{uy} h_1$.

3 For every evaluation of $H_{\delta u}\square$ inside some iterative solver of $H_{\delta u}\delta u = b$, solve

(a) $e_y h_3 = e_u \square$

(b) $e_y^* h_4 = L_{yy} h_3 - L_{yu}\square$

and set $H_{\delta u}\square := e_u^* h_4 - L_{uy} h_3 + L_{uu}\square$.

4 Set $u^{k+1} := u^k + \delta u$.

5 Set $\delta y := -h_1 - h_3$ and $y^{k+1} := y^k + \delta y$.

6 Set $\lambda^{k+1} := -h_2 + h_4$.

7 Set $k := k + 1$ and go back to step 2.

Figure 1. Reduced SQP Algorithm

volving e_y and another involving e_y^*. It will be seen in the sequel that in our case of e representing a time-dependent PDE these are in fact solutions of the linearized forward (state) equation and the corresponding backward (adjoint) equation, see figure 2 in the following section.

Note that the linear system involving the reduced Hessian $H_{\delta u}$ is significantly reduced in size as compared to the full Hessian of the Lagrangian, the more so as in practical applications, there are many more state than control variables.

4. Example

As an example, distributed control of a semilinear parabolic system of reaction-diffusion equations will be discussed. The PDE system describes a chemical reaction $C_1 + C_2 \rightarrow C_3$ where the three substances are subject to diffusion and a simple non-linear reaction law.

While in the discussion so far only *one* (scalar) PDE appears, the generalization to *systems* of PDEs is straightforward. In the example, the state $y = (c_1, c_2, c_3)^T$ as well as the adjoint $\lambda = (\lambda_1, \lambda_2, \lambda_3)^T$ now have three scalar components while the control is still one-dimensional. The linearized systems occuring in the computation of the auxiliary variables $h_1, \ldots, h_4$ feature a coupling between their components which is generated by the non-linearity in the state equation (16). Also note that this example satisfies the separation condition (15).

The reaction-diffusion system under consideration is given by

$$\begin{aligned} c_{1_t} &= D_1 \Delta c_1 - k_1 c_1 c_2 & \partial_n c_1 &= 0 & c_1(0) &= c_{10} \\ c_{2_t} &= D_2 \Delta c_2 - k_2 c_1 c_2 + u & \partial_n c_2 &= 0 & c_2(0) &= c_{20} \\ c_{3_t} &= D_3 \Delta c_3 + k_3 c_1 c_2 & \partial_n c_3 &= 0 & c_3(0) &= c_{30} \end{aligned} \tag{16}$$

where the control acts only through component two. The boundary conditions simply mean that the boundary of the reaction vessel is impermeable. The constants D_i and k_i are all non-negative and denote diffusion and reaction coefficients, respectively.

The objective in this case is a standard least-squares-type functional

$$f(y,u) = \int_\Omega [c_1(x,T) - c_{1d}]^2 \, dx + \gamma \int_Q u(x,t)^2 \, dx \, dt$$

in order to minimize the distance of component one's terminal state $c_1(x,T)$ to a given desired state c_{1d} while taking control cost into account, weighted by a factor $\gamma > 0$. In case one is interested in maximum product yield, the term $-\int_\Omega c_3(x,T) \, dx$ can be inserted into the objective.

The individual steps in the reduced Hessian algorithm for this particular example are given in figure 2. There the vector $(c_1^k, c_2^k, c_3^k, u^k, \lambda_1^k, \lambda_2^k, \lambda_3^k)^T$ denotes the current iterate. It stands out that the linear systems for $h_1, \ldots, h_4$ can equivalently be written as

$$h_{i_t} + \hat{K} h_i = f_i \quad \text{for} \quad i \in \{1,3\} \tag{17}$$

$$-h_{j_t} + \hat{K}^T h_j = g_j \quad \text{for} \quad j \in \{2,4\} \tag{18}$$

where the operator matrix

$$\hat{K} = \begin{bmatrix} -D_1\Delta - k_1 c_2^k & -k_1 c_1^k & 0 \\ -k_2 c_2^k & -D_2\Delta - k_2 c_1^k & 0 \\ k_3 c_2^k & k_3 c_1^k & -D_3\Delta \end{bmatrix} \tag{19}$$

is *non-symmetric*. Please notice that this phenomenon does *not* occur in *scalar* PDE control problems.

Reduced Hessian steps for the reaction-diffusion example

Solve for $h_1 = (h_{11}, h_{12}, h_{13})^T$:

$$
\begin{aligned}
&h_{11_t} - D_1\Delta h_{11} - k_1 c_2^k h_{11} - k_1 c_1^k h_{12} = c_{1_t}^k - D_1\Delta c_1^k - k_1 c_1^k c_2^k \\
&h_{12_t} - D_2\Delta h_{12} - k_2 c_1^k h_{12} - k_2 c_2^k h_{11} = c_{2_t}^k - D_2\Delta c_2^k - k_2 c_1^k c_2^k + u^k \\
&h_{13_t} - D_3\Delta h_{13} + k_3 c_2^k h_{11} + k_3 c_1^k h_{12} = c_{3_t}^k - D_3\Delta c_3^k + k_3 c_1^k c_2^k \\
&\partial_n h_{11} = 0 \qquad \partial_n h_{12} = 0 \qquad \partial_n h_{13} = 0 \\
&h_{11}(0) = c_1^k(0) - c_{10} \qquad h_{12}(0) = c_2^k(0) - c_{20} \qquad h_{13}(0) = c_3^k(0) - c_{30}
\end{aligned}
$$

Solve for $h_2 = (h_{21}, h_{22}, h_{23})^T$:

$$
\begin{aligned}
&-h_{21_t} - D_1\Delta h_{21} - k_1 c_2^k h_{21} - k_2 c_2^k h_{22} + k_3 c_2^k h_{23} = g_{21} \\
&-h_{22_t} - D_2\Delta h_{22} - k_2 c_1^k h_{22} - k_1 c_1^k h_{21} + k_3 c_1^k h_{23} = g_{22} \\
&-h_{23_t} - D_3\Delta h_{23} = 0 \\
&\qquad g_{21} = -k_1 h_{12}\lambda_1^k - k_2 h_{12}\lambda_2^k - k_3 h_{12}\lambda_3^k \\
&\qquad g_{22} = -k_1 h_{11}\lambda_1^k - k_2 h_{11}\lambda_2^k - k_3 h_{11}\lambda_3^k \\
&\partial_n h_{21} = 0 \qquad \partial_n h_{22} = 0 \qquad \partial_n h_{23} = 0 \\
&h_{21}(T) = 2[c_1^k(T) - c_{1d} - h_{11}(T)] \qquad h_{22}(T) = 0 \qquad h_{23}(T) = 0
\end{aligned}
$$

Set $b = -h_{22} - 2\gamma u^k$.
Solve for $h_3 = (h_{31}, h_{32}, h_{33})^T$:

$$
\begin{aligned}
&h_{31_t} - D_1\Delta h_{31} - k_1 c_2^k h_{31} - k_1 c_1^k h_{32} = 0 \\
&h_{32_t} - D_2\Delta h_{32} - k_2 c_1^k h_{32} - k_2 c_2^k h_{31} = -\square \\
&h_{33_t} - D_3\Delta h_{33} - k_3 c_2^k h_{31} - k_3 c_1^k h_{32} = 0 \\
&\partial_n h_{31} = 0 \qquad \partial_n h_{32} = 0 \qquad \partial_n h_{33} = 0 \\
&h_{31}(0) = 0 \qquad h_{32}(0) = 0 \qquad h_{33}(0) = 0
\end{aligned}
$$

Solve for $h_4 = (h_{41}, h_{42}, h_{43})^T$:

$$
\begin{aligned}
&-h_{41_t} - D_1\Delta h_{41} - k_1 c_2^k h_{41} - k_2 c_2^k h_{42} - k_3 c_2^k h_{43} = g_{41} \\
&-h_{42_t} - D_2\Delta h_{42} - k_2 c_1^k h_{42} - k_1 c_1^k h_{41} - k_3 c_1^k h_{43} = g_{42} \\
&-h_{43_t} - D_3\Delta h_{43} = 0 \\
&\qquad g_{41} = k_1 h_{32}\lambda_1^k + k_2 h_{32}\lambda_2^k + k_3 h_{32}\lambda_3^k \\
&\qquad g_{42} = k_1 h_{31}\lambda_1^k + k_2 h_{31}\lambda_3^k + k_3 h_{31}\lambda_3^k \\
&\partial_n h_{41} = 0 \qquad \partial_n h_{42} = 0 \qquad \partial_n h_{43} = 0 \\
&h_{41}(T) = 2h_{31}(T) \qquad h_{42}(T) = 0 \qquad h_{43}(T) = 0
\end{aligned}
$$

Set $H_{\delta u}\square := -h_{42} + 2\gamma\square$.

Figure 2. Reduced SQP Algorithm for the Reaction-Diffusion Example

5. Numerical Results

In this section, results obtained from an implementation of the reduced Hessian algorithm will be presented. All coding has been done in Matlab 6.0 using the PDE toolbox to generate the spatial mesh and the finite element matrices. The performance of the reduced Hessian algorithm will be demonstrated in comparison to an iterative algorithm working on the *full* Hessian of the Lagrangian H given in (11).

To this end the convergence behavior over iteration count of *one* particular SQP step (corresponding to steps 2 and 3 in the algorithm) will be shown. For the tests we chose

$$
\begin{array}{lll}
c_1^k(x,t) = 0.5 & c_{10}(x) = 0.1 + \chi_{\{x_1 > 0.3\}}(x) & \lambda_1^k(x,t) = 0 \\
c_2^k(x,t) = 0.5 & c_{20}(x) = 0.1 + \chi_{\{x_2 > 0.3\}}(x) & \lambda_2^k(x,t) = 0 \\
c_3^k(x,t) = 0.5 & c_{30}(x) = 0 & \lambda_3^k(x,t) = 0 \\
c_{1d}(x) = 0 & D_1 = 0.01 & k_1 = 0.5 \\
u^k(x,t) = 0 & D_2 = 0.05 & k_2 = 1.5 \\
\gamma = 1 & D_3 = 0.15 & k_3 = 2.5
\end{array}
$$

on some finite element discretization of the unit circle $\Omega \subset \mathbb{R}^2$, where χ_A denotes the indicator function of the set $A \cap \Omega$. The final time was $T = 10$.

As was seen earlier in equation (11), there are three block rows in H, corresponding to the linearizations of the adjoint equation, the optimality condition and the state equation, respectively. For our tests, these have been semi-discretized using piecewise linear triangular finite elements in space. The ODE systems obtained by the method of lines are of the following form:

$$M\dot{y} + Ky = f \quad \text{(forward equations)} \tag{20}$$

$$-M\dot{\lambda} + K^T\lambda = g \quad \text{(backward equations)} \tag{21}$$

They were treated by means of the implicit Euler scheme with constant step size. Of course, suitable higher order integrators can be used as well. Using this straightforward approach yields one drawback that becomes apparent in figure 3: The discretized full Hessian matrix H is no longer symmetric, although the continuous operator H is self-adjoint. The same holds for the discretized reduced Hessian $H_{\delta u}$.

This is due to the treatment of initial and terminal conditions in the linearized state and the adjoint equation. Nevertheless, there are methods that reestablish symmetry, but these will not be pursued in the course of this paper since qualitatively, the convergence results remain unchanged. For that reason, the non-symmetry will be approved,

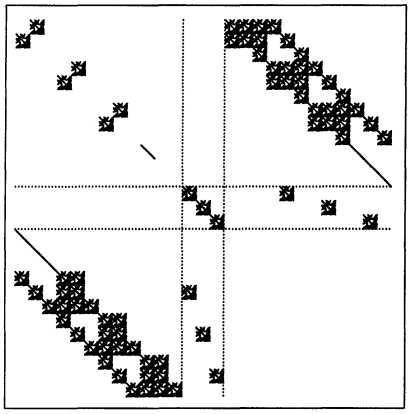

Figure 3. Non-symmetry of discretized full Hessian, $nt = 4$ time steps, implicit Euler, dotted lines indicate blocks corresponding to (11)

thereby waiving the possibility to use, e.g., a conjugate gradient method to solve the reduced problem but relying on iterative solvers capable of non-symmetric problems. In the tests, `GMRES` has proved quite efficient on the full Hessian problem while `CGS` and `BICGSTAB` failed to generate reasonably better iterates than the initial all-zero guess. For the reduced Hessian, all three algorithms found the solution to high accuracy, and `CGS` needed the fewest iterations to do so. As a common basis, `GMRES` with no restarts was used for both the full and the reduced Hessian problem.

Note that while the discretized state and adjoint allocate nt (equal to 4 in figure 3) discrete time steps, the discretized control needs only $nt - 1$. This is attributed to the use of the Euler method where, after discretization, $u(t = 0)$ does not appear in any of the equations.

In order to illustrate the convergence properties, it is convenient to have the *exact* discretized solution $(\delta y, \delta u, \lambda^{k+1})$ of the full SQP step (11) at hand. To that end, the full Hessian matrix was set up explicitly for a set of relatively coarse discretizations, and the exact solution was computed using a direct solver based on Gaussian elimination (Matlab's backslash operator). The exact solution δu of (12) was obtained in the same way after setting up the reduced Hessian matrix, where the corresponding δy and λ^{k+1} were calculated performing the forward/backward integration given by (13) and (14). These two reference solution triplets differ only by entries of order 1E-15 and will be considered equal.

It has to be mentioned that for these low-dimensional examples (cf. table 1), a direct solver is a lot faster than any iterative algorithm. However, setting up the exact reduced Hessian matrix of course is not an option for fine discretizations.

Figures 4–6 illustrate the convergence behavior of `GMRES` working on the reduced versus the full Hessian matrix: For δu_{ref} denoting the exact

discretized solution, the graphs show the relative error history

$$e^j(t) = \frac{||\delta u^j(t) - \delta u_{\text{ref}}(t)||}{||\delta u_{\text{ref}}(t)||} \tag{22}$$

in the L^2 norm, where $\delta u^j(t)$ denotes the approximate solution generated by the iterative solver after j iterations, taken at the time grid point $t \in [0, T]$. The same relative errors can be defined for δu substituted by $\delta c_1, \ldots, \delta c_3$ or $\lambda_1^{k+1}, \ldots, \lambda_3^{k+1}$ which are the components of the state update δy and the new adjoint estimate λ^{k+1}.

Each figure shows the relative error history $e_j(t)$ of either δu or δc_1 obtained using **GMRES** with no restarts after $j = 4, 8, \ldots, 28$ iterations on the reduced problem and after $j = 100, 200, \ldots, 600$ iterations on the full problem. The figures for δc_2, δc_3 and $\lambda_1^{k+1}, \ldots, \lambda_3^{k+1}$ look very much the same and are not shown here. The discretization level is characterized by the number of discrete time steps *nt* and the number of grid points in the finite element mesh *poi*. Table 1 lists the number of optimization variables in the full and reduced case for the individual discretizations used.

nt	*poi*	*# of vars (reduced)*	*# of vars (full)*
9	25	200	1550
9	81	648	5022
19	81	1458	10692

Table 1. Number of optimization variables for different discretizations

It can clearly be seen that the iterative solver works very well on the reduced system while it needs many iterations on the full matrix. This was to be expected since it is a well-known fact (see, e.g., [1] and [2]) that iterative solvers working on the full Hessian require preconditioning. Although the evaluation of $H_{\delta u}$ times a vector is computationally more expensive than H times a vector, the reduced Hessian algorithm is by far the better choice over the unpreconditioned full algorithm. To give some idea why the reduced Hessian algorithm outperforms the full Hessian version, let us define

$$P = \begin{bmatrix} -e_u^* e_y^{-*} & I & e_u^* e_y^{-*} L_{yy} e_y^{-1} \\ 0 & 0 & I \\ I & 0 & 0 \end{bmatrix} \tag{23}$$

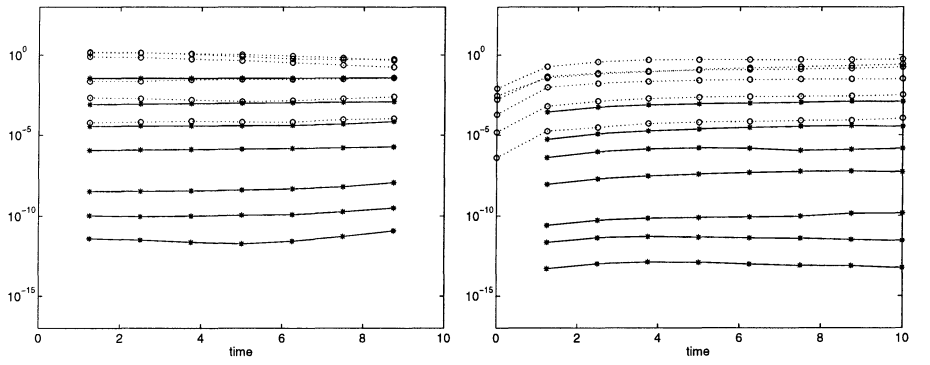

Figure 4. Relative error history for δu (left) and δc_1 (right) on the reduced (solid lines) problem for $j = 4, 8, \ldots, 28$ iterations and on the full (dotted lines) problem for $j = 100, 200, \ldots, 600$ iterations at discretization level $nt = 9$, $poi = 25$

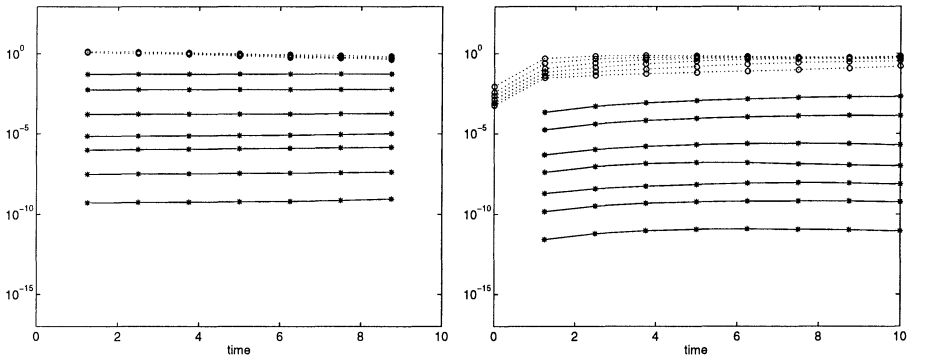

Figure 5. Relative error history for δu (left) and δc_1 (right) on the reduced (solid lines) problem for $j = 4, 8, \ldots, 28$ iterations and on the full (dotted lines) problem for $j = 100, 200, \ldots, 600$ iterations at discretization level $nt = 9$, $poi = 81$

as the *left preconditioner* for the full Hessian problem (11) with the first two columns permuted (for simplicity, the separation condition (15) is assumed to hold): From (11), we get

$$P \begin{bmatrix} & L_{yy} & e_y^* \\ L_{uu} & & e_u^* \\ e_u & e_y & \end{bmatrix} \begin{bmatrix} \delta u \\ \delta y \\ \lambda^{k+1} \end{bmatrix} = -P \begin{bmatrix} f_y \\ f_u \\ e \end{bmatrix}, \tag{24}$$

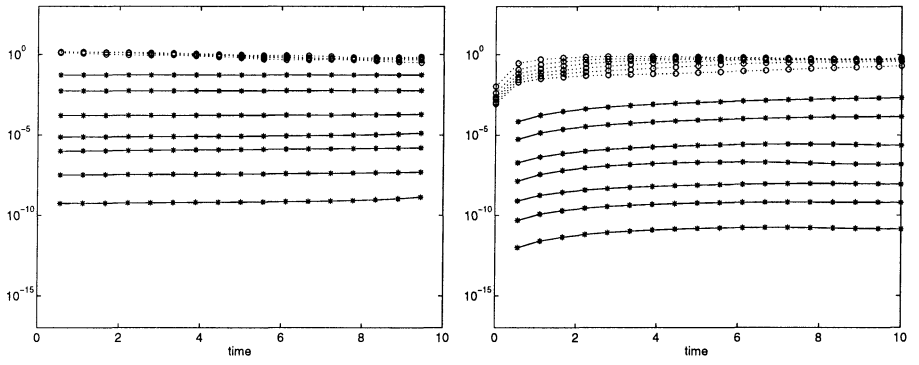

Figure 6. Relative error history for δu (left) and δc_1 (right) on the reduced (solid lines) problem for $j = 4, 8, \ldots, 28$ iterations and on the full (dotted lines) problem for $j = 100, 200, \ldots, 600$ iterations at discretization level $nt = 19$, $poi = 81$

which is equivalent to the block-triangular system

$$\begin{bmatrix} H_{\delta u} & & \\ e_u & e_y & \\ & L_{yy} & e_y^* \end{bmatrix} \begin{bmatrix} \delta u \\ \delta y \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} e_u^* e_y^{-*} \left[f_y - L_{yy} e_y^{-1} e \right] - f_u \\ -e \\ -f_y \end{bmatrix} \tag{25}$$

whose rows are just the equations (12)–(14). Hence the reduced Hessian problem is nothing else than the full problem after preconditioning with P. Comparing (11) to (25), it turns out that the preconditioning actually provides the iterative solver with some insight into the interdependence of the unknown variables. While in the full Hessian system, the solver takes *all* variables as degrees of freedom, in the reduced system only the *true* free variables (i.e. the controls) appear and the state and the adjoint are calculated consistently. From this point of view, the reduced Hessian method resembles what is usually called a *direct single shooting* approach, applied to a linear-quadratic model.

The necessity to have the full and reduced Hessian matrix explicitly available for the numerical tests limits the discretization levels to very coarse ones throughout this paper. In practice, however, control problems for time-dependent PDEs with about 275 000 unknowns (including 40 000 control variables) have been successfully solved on a desktop PC within 2 hours using the reduced Hessian SQP algorithm.

References

[1] Battermann, A. & Heinkenschloss, M.: *Preconditioners for Karush-Kuhn-Tucker Matrices Arising in the Optimal Control of Distributed Systems*, in: W. Desch,

F. Kappel, K. Kunisch (eds.), Optimal Control of Partial Differential Equations, Vorau 1997, Birkhäuser Verlag, Basel, Boston, Berlin, 1998, pp. 15-32.

[2] Biros, G. & Ghattas, O.: *Parallel Lagrange-Newton-Krylov-Schur Methods for PDE-Constrained Optimization. Part I: The Krylov-Schur Solver*, Technical Report, Laboratory for Mechanics, Algorithms, and Computing, Carnegie Mellon University, 2000.

[3] Biros, G. & Ghattas, O.: *Parallel Lagrange-Newton-Krylov-Schur Methods for PDE-Constrained Optimization. Part II: The Lagrange-Newton Solver, and its Application to Optimal Control of Steady Viscous Flows*, Technical Report, Laboratory for Mechanics, Algorithms, and Computing, Carnegie Mellon University, 2000.

[4] Hinze, M. & Kunisch, K.: *Second Order Methods for Optimal Control of Time-dependent Fluid Flow*, Bericht Nr. 165 des Spezialforschungsbereichs F003 Optimierung und Kontrolle, Karl-Franzens-Universität Graz (1999), to appear in SIAM J. Control Optim.

[5] Kupfer, F.-S.: *An infinite-dimensional convergence theory for reduced SQP methods in Hilbert space*, SIAM J. Optimization 6, 1996.

[6] Nocedal, J. & Wright, S.: *Numerical Optimization*, Springer, 1999.

[7] Tröltzsch, F.: *On the Lagrange–Newton–SQP Method for the Optimal Control of Semilinear Parabolic Equations*, SIAM J. Control Optim. 38, No. 1, pp. 294–312, 1999.