

# Parallel Solution of Optimal-Control Problems by Time-Domain Decomposition<sup>1</sup>

Martin Berggren and Matthias Heinkenschloss

*FFA, the Aeronautical Research Institute of Sweden, Computational Aerodynamics Branch,  
P.O. Box 11021, S-161 11 Bromma, Sweden; bnm@ffa.se*

*Department of Computational and Applied Mathematics, Rice University, 6100 S. Main  
Street, Houston, TX 77005-1892; heinken@caam.rice.edu*

## 1. INTRODUCTION

There is at present a growing interest for the interdisciplinary and challenging computational problem of solving optimal-control problems involving partial differential equations (PDEs). The optimal-control methodology has been used, for instance, to optimize the design of airfoils or wings with respect to quantities such as the drag coefficients or the lift/drag ratio [Jam95]. For unsteady fluid-dynamics problems modeled by the Navier-Stokes equations, the same technique has also been used to compute the optimal control of disturbances in a spatially growing boundary layer [JGN<sup>+</sup>95], and the optimal control of jets in a cavity flow [Ber95, Ber97].

Since the computational cost for optimal-control problems is dominated by the repeated solution of the equations which models the systems of interest, say the Navier-Stokes equations in three space dimensions, it makes sense to use as efficient optimization algorithms as possible and, in particular, algorithms that may be implemented for computers with a parallel architecture. We concentrate in this article on parallel algorithms for optimal control of unsteady problems.

If no state constraints are imposed, the *adjoint-equation* approach offers an efficient way of computing the gradients required by most optimization algorithms.

---

<sup>1</sup> Appeared in: M-O. Bristeau, G. Etgen, W. Fitzgibbon, J. L. Lions, J. Periaux, and M. F. Wheeler (eds.), *Computational Science for the 21st Century*, J. Wiley, Chichester, 1997, pp. 102–112.

Unfortunately, this is an inherently serial process. We introduce in this article, a *time-domain decomposition approach* which uses adjoint equations for gradient computations but nevertheless is parallelizable. In short, we divide the temporal domain in a multiple shooting fashion into nonoverlapping subintervals and solve repeatedly (but only approximately) modified control problems on each subdivision, successively altering the initial data for the state and adjoint equations. When stating the time-domain decomposition algorithm below, we do not specify how the control problems on each subdivision is solved; any suitable algorithm can be used. The methodology is discussed for a linear, parabolic PDEs. In our numerical example the optimal control subproblems are solved using a conjugate gradient method. However, formally, the approach is quite general and may potentially be applied to various kinds of optimal-control problems for linear or nonlinear unsteady models.

Previous work regarding parallel methods for optimal-control of unsteady models has been concentrated on systems directly formulated in finite dimensions, the *discrete-time optimal-control problem*. A parallel factorization of the optimality system has been introduced in [Wri90]. Several authors have used, similarly as here, a decomposition in time to introduce parallelism [CCL89, Ral96, Wri91, PP92]. The subproblems are not the same as proposed here, and these methods heavily rely on numerical linear algebra techniques and are not practicable for the problems we consider here.

Another parallel solution approach to general constraint optimization problems is *parallel constraint distribution*, a method related to the so-called *partial proximal minimization* [FM91, Fer94, BT94]. In the context of our applications, at each iteration and at each processor, the state equation is explicitly enforced at one time step only, whereas the state-equation constraint at the other time steps are enforced implicitly by adding augmented Lagrangian terms to the objective function. Each of these problems are solved and the solutions are combined in an appropriate way to form the next iteration. This introduces parallelism, but the size of the problems to be solved in parallel is equal to the size of the original problem, only the number of constraints is reduced. We mention this approach here because our method has features related to the parallel constraint distribution, but avoids the above mentioned deficiencies of this approach.

Many of the parallel methods reviewed above heavily use techniques such as matrix factorizations of optimality systems, dynamic programming techniques, or assume exact controllability of the subproblems. This makes the methods unsuitable for most problems arising from PDEs. The approach introduced here is designed with PDEs in mind. It is suitable for large scale problems and for implementation on parallel distributed-memory machines, and it may be combined with various minimization algorithms and with storage-reduction schemes ([Gri92], [BGL96]). In the following, we introduce the main idea and present some numerical results. Although our method performed successfully in our numerical tests, currently, the theoretical justification of this methods is only given for special cases. We sketch our convergence results and address possible extensions of our method.

## 2. THE MODEL PROBLEM

We consider a control problem applied to a general, linear parabolic partial differential equation,

$$\begin{aligned} \frac{\partial y}{\partial t} + Ay &= Bv && \text{in } Q, \\ y &= 0 && \text{on } \Sigma, \\ y(0) &= \bar{y}_0, \end{aligned} \tag{1}$$

where  $y$  is a scalar function of space and time. We denote the spatial domain and its boundary  $\Omega$  and  $\partial\Omega$ , respectively. We consider a bounded temporal domain  $(0, T)$ ,  $T > 0$  and define  $Q = \Omega \times (0, T)$  and  $\Sigma = \partial\Omega \times (0, T)$ .

In equation (1),  $A$  is a second-order elliptic partial differential operator. Typical examples to have in mind for the operator  $A$  are the Laplacian  $-\Delta$ , or advection-diffusion operators  $-\Delta + \mathbf{V} \cdot \nabla$  with  $\nabla \cdot \mathbf{V} = 0$ . Initial data is supplied by the function  $\bar{y}_0$ . (Other boundary conditions than the homogeneous Dirichlet conditions of equation (1) may as well be used.)

The precise way in which the control acts on the system is not important for what follows. For simplicity of notation we let the function  $v$  control the system (1) through the ‘‘actuator’’  $B$  whose domain is a Hilbert space  $X$ . We consider here  $\mathcal{U} = L^2((0, T); X)$  as our space of *admissible controls*.

Given functions  $z_1 \in L^2(Q)$  and  $z_2 \in L^2(\Omega)$ , we introduce the objective function

$$J(v) = \frac{1}{2} \int_0^T \|v\|^2 dt + \frac{k_1}{2} \int_Q (y - z_1)^2 dx dt + \frac{k_2}{2} \int_\Omega (y(T) - z_2)^2 dx$$

where  $k_1, k_2 \geq 0$  such that  $k_1 + k_2 > 0$ . The control problem is then

$$\begin{aligned} \text{Find } u \in \mathcal{U} \text{ such that} \\ J(u) \leq J(v), \forall v \in \mathcal{U}. \end{aligned} \tag{2}$$

Problem (2) has a unique solution  $u$  (under suitable assumptions), which satisfies

$$u + B^* p = 0, \tag{3}$$

where  $B^*$  is the adjoint operator to  $B$ , and where  $p$  satisfies

$$\begin{aligned} -\frac{\partial p}{\partial t} + A^* p &= k_1(y - z_1) && \text{in } Q, \\ p &= 0 && \text{on } \Sigma, \\ p(T) &= k_2(y(T) - z_2). \end{aligned} \tag{4}$$

The *state equation* (1), the *adjoint equation* (4) and the *gradient condition* (3) are a coupled system of equations, commonly referred to as the *optimality system* associated to problem (2).

We have not given the precise assumptions and the functional framework (see [Lio71, GL94], e. g., for this) that are needed to obtain a well-defined control problem since this is of no importance for the purpose of *stating* our parallel solution approach. What assumptions are made, on the operator  $A$  for instance, are of course important in the *analysis* of the algorithm, which is not provided here (cf. Section 5.).

### 3. A PARALLEL SOLUTION APPROACH

Let us partition the time interval  $(0, T)$  into  $N$  subintervals such that  $[0, T] = \cup_{n=0}^{N-1} [T_n, T_{n+1}]$ , where  $0 = T_0 < T_1 < \dots < T_n = T$ , and let us define

$$Q_n = \Omega \times (T_n, T_{n+1}), \quad \Sigma_n = \partial\Omega \times (T_n, T_{n+1}), \quad n = 0, \dots, N-1.$$

Introducing the auxiliary variables  $\{\eta_n\}_{n=1}^{N-1}$  and  $\{\lambda_n\}_{n=0}^{N-2}$ , the optimality system (1), (3), and (4) may now be recast in the form:  
for  $n = 0, \dots, N-1$ ,

$$\begin{aligned} \frac{\partial y_n}{\partial t} + Ay_n &= Bu_n && \text{in } Q_n, \\ y_n &= 0 && \text{on } \Sigma_n, \\ y_n(T_n) &= \begin{cases} \bar{y}_0 & \text{for } n = 0, \\ \eta_n, & \text{otherwise,} \end{cases} \\ -\frac{\partial p_n}{\partial t} + A^* p_n &= k_1(y_n - z_{1,n}) && \text{in } Q_n, \\ p_n &= 0 && \text{on } \Sigma_n, \\ p_n(T_{n+1}) &= \begin{cases} k_2(y_{N-1}(T) - z_2) & \text{if } n = N-1, \\ \lambda_n & \text{otherwise,} \end{cases} \\ u_n + B^* p_n &= 0, \end{aligned} \tag{5}$$

together with the conditions

$$\begin{aligned} \eta_n &= y_{n-1}(T_n) && \text{for } n = 1, \dots, N-1, \\ \lambda_n &= p_{n+1}(T_{n+1}) && \text{for } n = 0, \dots, N-2, \end{aligned}$$

that is, continuity of the states  $y_n$  and the co-states  $p_n$  at the subinterval boundaries.

The systems (5) are the optimality conditions of the following problems: for  $n = 0, \dots, N-2$ ,

$$\left\{ \begin{array}{l} \text{Solve inf } J_n(v_n), \text{ where} \\ J_n(v_n) = \frac{1}{2} \int_{T_n}^{T_{n+1}} \|v_n\|^2 dt + \frac{k_1}{2} \int_{Q_n} (y_n - z_{1,n})^2 dx dt \\ \quad + \int_{\Omega} \lambda_n (y_n(T_{n+1}) - \eta_{n+1}) dx, \\ \text{and where} \\ \frac{\partial y_n}{\partial t} + Ay_n = Bv_n && \text{in } Q_n, \\ y_n = 0 && \text{on } \Sigma_n, \\ y_n(T_n) = \begin{cases} \bar{y}_0 & \text{for } n = 0, \\ \eta_n & \text{otherwise,} \end{cases} \end{array} \right. \tag{6a}$$

and for  $n = N - 1$

$$\left\{ \begin{array}{l} \text{Solve inf } J_{N-1}(v_{N-1}), \text{ where} \\ J_{N-1}(v_{N-1}) = \frac{1}{2} \int_{T_{N-1}}^{T_N} \|v_{N-1}\|^2 dt \\ \quad + \frac{k_1}{2} \int_{Q_{N-1}} (y_{N-1} - z_{1,N-1})^2 dx dt + \frac{k_2}{2} \int_{\Omega} (y_{N-1}(T) - z_2)^2 dx, \\ \text{and where} \\ \frac{\partial y_{N-1}}{\partial t} + Ay_{N-1} = Bv_{N-1} \quad \text{in } Q_{N-1}, \\ \quad \quad \quad y_{N-1} = 0 \quad \quad \quad \text{on } \Sigma_{N-1}, \\ y_{N-1}(T_{N-1}) = \eta_{N-1}. \end{array} \right. \quad (6b)$$

Thus, the problem (2) is equivalent to the following problem:

$$\begin{array}{l} \text{For } n = 0, \dots, N - 1, \text{ solve inf } J_n(v_n) \text{ subject to} \\ \eta_n = y_{n-1}(T_n) \quad \text{for } n = 1, \dots, N - 1, \\ \lambda_n = p_{n+1}(T_{n+1}) \quad \text{for } n = 0, \dots, N - 2. \end{array} \quad (7)$$

Note that the minimization problems in (7) are coupled only through the equality constraints. We propose a relaxation approach to solve the coupled problems, and introduce a ‘red–black’ ordering to allow for parallelism in solving the subproblems. The ‘red’ indices are  $\{0, 2, 4, \dots\}$  and the ‘black’ indices are  $\{1, 3, 5, \dots\}$ . Given an current set  $\{\eta_n^{(k)}\}_{n=1}^{N-1}$  and  $\{\lambda_n^{(k)}\}_{n=0}^{N-2}$ , the next iterates  $\{\eta_n^{(k+1)}\}_{n=1}^{N-1}$  and  $\{\lambda_n^{(k+1)}\}_{n=0}^{N-2}$  in a relaxation approach of the Gauss–Seidel type are computed as follows:

**Algorithm 1**

- (1) Solve  $\text{inf } J_{2n}(v_{2n})$  with  $\eta_{2n} = \eta_{2n}^{(k)}$ ,  $\eta_{2n+1} = \eta_{2n+1}^{(k)}$ , and  $\lambda_{2n} = \lambda_{2n}^{(k)}$ .
- (2) Set  $\eta_{2n+1}^{(k+1)} = y_{2n}(T_{2n+1})$  and  $\lambda_{2n-1}^{(k+1)} = p_{2n}(T_{2n})$ .
- (3) Solve  $\text{inf } J_{2n+1}(v_{2n+1})$  with  $\eta_{2n+1} = \eta_{2n+1}^{(k+1)}$ ,  $\eta_{2n+2} = \eta_{2n+2}^{(k)}$ , and  $\lambda_{2n+1} = \lambda_{2n+1}^{(k+1)}$ .
- (4) Set  $\eta_{2n+2}^{(k+1)} = y_{2n+1}(T_{2n+2})$  and  $\lambda_{2n}^{(k+1)} = p_{2n+1}(T_{2n+1})$ .

The optimality conditions of step (1) and (3) above are still given by equation (5). All ‘red’ problems (step (1)) are uncoupled and may be solved in parallel, and similarly for the ‘black’ problems (step (3)).

A serious complication in solving the original, serial problem (2) is the large amount of computer memory that is needed in storing the states  $y$  simultaneously at all points in space and time, as needed in the forcing term of the adjoint equation. Algorithm 1 requires the storage of the controls  $u$ , and of the states  $\eta_n$  and the adjoints  $\lambda_n$  at the interfaces. In addition, for the solution of the subproblems one needs to store the states  $y_n$  throughout  $Q_n$ . These latter variables have to be stored temporarily and locally to the processor used for solving this subproblem. Note that the large amount of memory typically available in distributed-memory machines will be efficiently utilized in this case because of the parallel structure. Yet, it may be impossible, even with today's

most powerful computers, to store all of this data simultaneously. In that case, one may use, on each of the subproblems of Algorithm 1, the storage-reduction scheme introduced<sup>2</sup> in [Gri92] or a similar one discussed for optimal-control problems in § 6.6.2 of [Ber95] and in [BGL96].

We see from the definition of the objective functions  $J_n$  that the continuity of the states are enforced through a Lagrangian term. This observation makes it natural to attempt improving the robustness of the algorithm by adding a penalty term to the functions  $J_n$  obtaining instead an *augmented Lagrangian* ([Glo84], [Ber82]).

Letting  $\gamma \geq 0$ , we may replace the cost functions in (6) with the following:

$$\begin{aligned}
 J_n^\gamma(v_n) &= \frac{1}{2} \int_{T_n}^{T_{n+1}} \|v_n\|^2 dt + \frac{k_1}{2} \int_{Q_n} (y_n - z_{1,n})^2 dx dt + \int_{\Omega} \lambda_n (y_n(T_{n+1}) - \eta_{n+1}) dx \\
 &\quad + \frac{\gamma}{2} \int_{\Omega} (y_n(T_{n+1}) - \eta_{n+1})^2 dx \quad \text{for } n = 0, \dots, N-2, \\
 J_{N-1}^\gamma &= J_{N-1}.
 \end{aligned} \tag{8}$$

Instead of equations (5), the optimality systems for the minimization of the modified functionals becomes

$$\begin{aligned}
 \frac{\partial y_n}{\partial t} + Ay_n &= Bu_n && \text{in } Q_n, \\
 y_n &= 0 && \text{on } \Sigma_n, \\
 y_n(T_n) &= \begin{cases} \bar{y}_0 & \text{for } n = 0, \\ \eta_n, & \text{otherwise,} \end{cases} \\
 -\frac{\partial p_n}{\partial t} + A^*p_n &= k_1(y_n - z_{1,n}) && \text{in } Q_n, \\
 p_n &= 0 && \text{on } \Sigma_n, \\
 p_n(T_{n+1}) &= \begin{cases} k_2(y_{N-1}(T) - z_2) & \text{if } n = N-1, \\ \lambda_n + \gamma(y_n(T_{n+1}) - \eta_{n+1}) & \text{otherwise,} \end{cases} \\
 u_n + B^*p_n &= 0,
 \end{aligned} \tag{9}$$

for  $n = 0, \dots, N-1$ . This system of equation may now be solved with the relaxation algorithm 1 replacing  $J_n$  with  $J_n^\gamma$ .

#### 4. NUMERICAL EXPERIMENTS

We have performed some preliminary tests of Algorithm 1 in a small-scale serial implementation. Since this appears to be a new approach to solving unsteady optimal-control problems, we believe that such tests are a necessary first step before attempting larger, more realistic problems on machines with a parallel architecture.

For these tests, we choose a one-dimensional problem with  $\Omega = (0, 1)$ , and with the

---

<sup>2</sup> in the context of reverse-mode automatic differentiation

state equation

$$\begin{aligned} \frac{\partial y}{\partial t} - \frac{\partial^2 y}{\partial x^2} &= \sum_{m=1}^M v_m(t) \delta(x - a_m) && \text{in } Q, \\ \frac{\partial y}{\partial x}(0, t) &= 0 \\ y(1, t) &= 0 \\ y(0) &= \bar{y}_0, \end{aligned}$$

where  $\delta(x - a_m)$  denotes the Dirac measure at the point  $a_m$ . Thus, we consider the heat equation controlled through the functions  $v_m \in L^2(0, T)$  acting on the system through pointwise forcing at locations  $a_m \in (0, 1)$ ,  $m = 1, \dots, M$ . In fact, we also allow  $a_m = 0$ ; this is equivalent to a *Neumann control* at the left boundary.

Continuous, piecewise-linear functions are used for the (finite-element) spatial discretization and the Backward-Euler scheme is used for temporal discretization. Some care has to be taken when defining the discrete optimality system corresponding to (9). Not any discretization of this system will produce a system consistent with the original optimality system on the full time interval  $(0, T)$ . The details of this will be presented in a forthcoming report.

#### 4.1. RESULTS

We consider a case with 9 pointwise control actuators ( $M = 9$ ) located at

$$a_m = \frac{m}{10}, \quad m = 0, \dots, 8,$$

with the targets

$$\begin{aligned} z_1 &= \begin{cases} -\frac{1}{2} & \text{if } x < \frac{1}{2}, \\ 0 & \text{if } x > \frac{1}{2}; \end{cases} \\ z_2 &= 1 - x^3, \end{aligned}$$

the initial condition  $\bar{y}_0 = z_1$ , and using the following parameter values:

$$T = 0.2, \quad k_1 = k_2 = 10^2, \quad \Delta x = \frac{1}{40}, \quad \Delta t = \frac{T}{32}.$$

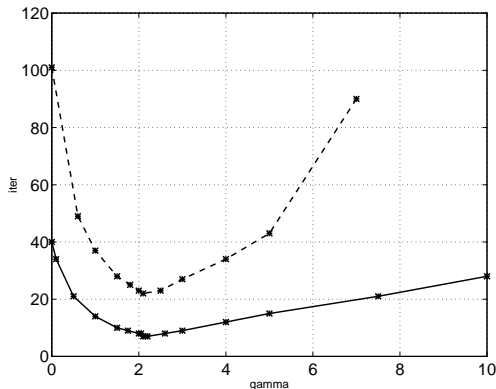
Solving the discrete version of control problem (2) for the problem above with the conjugate gradient (CG) algorithm requires 17 iterations for a relative gradient tolerance of  $10^{-8}$  in 8 byte precision.

In the discrete version of relaxation algorithm 1, we solve the red and black subproblems by the conjugate-gradient algorithm and study in this section the behavior of the algorithm for different parameter choices. As a convergence criterion for the relaxation iterations we use

$$\frac{\|\mathbf{u}^R - \mathbf{u}^{CG}\|}{\|\mathbf{u}^{CG}\|} \leq 10^{-6}, \quad (10)$$

where  $\mathbf{u}^R$  is the solution of the partitioned control problem computed by Algorithm<sup>3</sup> 1, and  $\mathbf{u}^{CG}$  is the solution computed by the CG method, using a relative gradient

<sup>3</sup> Or, rather, its discrete version. From now on, we will drop the qualifier “discrete version” for simplicity.



**Figure 1** The number of relaxation iteration versus the penalty parameter  $\gamma$  for the case  $N = 4$  (solid) and for the case  $N = 16$  (dashed).

tolerance of  $10^{-8}$ , applied directly to problem (2). The convergence criterion (10) is used to verify that we obtain the correct solution; in a practical application something else has to be used of course.

Letting  $\Delta x = 1/10$ , we apply Algorithm 1 with the augmented objective functions  $J_n^\gamma(u_n)$  for various choices of  $\gamma$ . Figure 1 shows the number of relaxation iterations versus the penalty parameter  $\gamma$  for a case of few subintervals ( $N = 4$ ) and the case of many subintervals ( $N = 16$ ). Thus, the use of the penalization term in the relaxation algorithm significantly improves the convergence, and the optimal value of  $\gamma$  does not appear to depend on the size of the subproblems.

In the runs of Figure 1, the subproblems were solved quite accurately, up to a relative gradient tolerance of  $10^{-8}$ . The next question is whether this is necessary. To investigate this, we put a limit on the maximum number of iterations allowed in the red and black subproblems. For the same problem as above, using  $\gamma = 2.1$ , Table 1 shows the number of relaxation iterations versus the maximum number of iterations in the subproblems. For the sake of comparison, let us remark that the number of subproblem iterations needed to achieve full convergence varied between 6 and 16. From the results of Table 1 we conclude that the subproblems need to be solved only approximately, and also that that larger subproblems ( $N = 4$ ) needs to be solved somewhat more accurately than smaller subproblems ( $N = 16$ ).

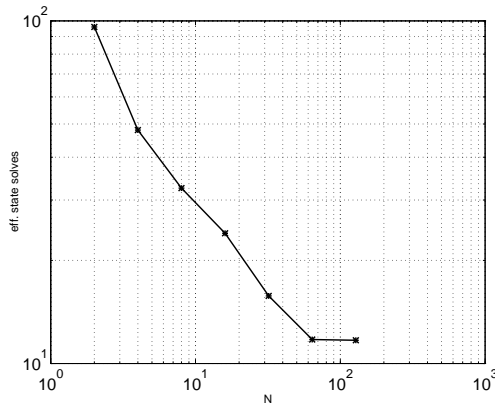
Finally, we investigate the efficiency of the method. Usually, the main computational work for optimal-control problems involving a time-dependent PDE is associated with the solutions of the state and adjoint equations. Therefore, as a measure of the computational cost we use the number of *effective state solves*: the cost of solving the state or adjoint equation once on the full interval  $(0, T)$ . Solving once the state (or adjoint) equations on the ‘black’ (or ‘red’) processors requires  $1/N$  effective state solves, assuming that  $N$  processors are available. If  $k$  iterations are made in the conjugate-gradient algorithm of each subproblem, the state equations has to be solved  $k+1$  times and the adjoint equation also  $k+1$  times. Thus, the solution of either the red or black subproblem requires  $2(k+1)/N$  effective state solves for  $k$  conjugate-gradient

**Table 1** The number of relaxation iteration for different limits on the iterations of the subproblems. No convergence is indicated by n. c. The subiteration limit of 50 means in this case that full convergence will be reached.

subit. max	# of relax. iter. $N = 4$	iter. $N = 16$
1	318	n. c.
2	45	28
3	20	22
4	10	22
5	7	22
50	7	22

**Table 2** The number of relaxation iteration and effective state solves for different sizes  $N$  of the subproblems. Also given are the (optimal) values on the limit of the subiterations.

$N$	# of relax. iter.	# of eff. state solves	subit. max.
2	8	96	5
4	8	48	5
8	13	32.5	4
16	24	24	3
32	63	15.8	1
64	94	11.8	1
128	187	11.7	1



**Figure 2** The number of effective state solves versus the size of the subproblems  $N$ .

iterations of the subproblems.

To investigate the parallel efficiency, we use a finer temporal discretization,  $\Delta t = T/128$ , fix the penalty parameter to  $\gamma = 2.1$  and solve the problem for partitionings  $N = 2, 4, \dots, 128$ , and count the number of effective state solves. In these calculation, we do not solve the subproblems exactly, but limit the subiterations to a fixed number, which may be different for different  $N$ . This iteration limit is chosen, by trial and error, to minimize the number of effective state solves for each particular value of  $N$ . The results of this investigations are shown in Table 2. From Figure 2, we see that for  $4 \leq N \leq 64$ , the number of effective state solves decreases approximately<sup>4</sup> as  $N^{-1/2}$ .

<sup>4</sup> A least-squares fit yields the value  $-0.51$  for the exponent in the interval  $4 \leq N \leq 64$ .

## 5. OUTLOOK AND CONCLUSION

Algorithm 1 is to our knowledge not analyzed in the literature. For special cases we have established convergence of this algorithm in the infinite-dimensional framework. This analysis assumes that the operator  $A$  is generated by a bilinear, continuous, but not necessarily symmetric, form  $a$  over  $H^1(\Omega) \times H^1(\Omega)$  which satisfies

$$\nu \int_{\Omega} |y|^2 dx \leq a(y, y), \quad \forall y \in H_0^1(\Omega), \quad (11)$$

for some  $\nu > 0$ . A complete discussion, however, even for these cases is lengthy and beyond the scope of the present article. Moreover, numerical evidence (not presented here) indeed indicate that the algorithm need not to converge for non-diffusive equations when  $\gamma = 0$ , but that for  $\gamma > 0$ , convergence appears to hold even for cases when the state equations is not diffusive. A comprehensive convergence analysis of this algorithm is under investigation.

## REFERENCES

- [Ber82] Bertsekas D. P. (1982) *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York.
- [Ber95] Berggren M. (1995) *Optimal Control of Time Evolution Systems: Controllability Investigations and Numerical Algorithms*. Ph. D. thesis, Department of Computational and Applied Mathematics, Rice University, Houston, Tx 77251-1892.
- [Ber97] Berggren M. (1997) Numerical solution of a flow-control problem: Vorticity reduction by dynamic boundary action. *SIAM Journal on Scientific Computing*, 1997 (to appear).
- [BGL96] Berggren M., Glowinski R., and Lions J. L. (1996) A computational approach to controllability issues for flow-related models. (I): Pointwise control of the viscous Burgers equation. *International Journal of Computational Fluid Dynamics* 7: 237–252.
- [BT94] Bertsekas D. P. and Tseng P. (1994) Partial proximal minimization algorithms for convex programming. *SIAM Journal on Optimization* 4: 551–572.
- [CCL89] Chang S.-C., Chang T.-S., and Luh P. B. (1989) A hierarchical decomposition for large-scale optimal control problems with parallel processing structure. *Automatica* 25(1): 77–86.
- [Fer94] Ferris M. C. (1994) Parallel constraint distribution in convex quadratic programming. *Mathematics of Operations Research* 19: 645–658.
- [FM91] Ferris M. C. and Mangasarian O. L. (1991) Parallel constraint distribution. *SIAM Journal on Optimization* 1: 487–500.
- [GL94] Glowinski R. and Lions J. L. (1994) Exact and approximate controllability for distributed parameter systems (Part I). *Acta Numerica* pages 269–378.
- [Glo84] Glowinski R. (1984) *Numerical Methods for Nonlinear Variational Problems*. Springer-Verlag, New York.
- [Gri92] Griewank A. (1992) Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software* 1: 35–54.
- [Jam95] Jameson A. (John Wiley & Sons, 1995) Optimum aerodynamic design using control theory. *Computational Fluid Dynamics Review*.
- [JGN<sup>+</sup>95] Joslin R. D., Gunzburger M. D., Nicolaidis R. A., Erlebacher G., and Hussaini M. Y. (1995) A self-contained, automated methodology for optimal flow control validated for transition delay. Submitted to *Journal of Fluid Mechanics*.
- [Lio71] Lions J. L. (1971) *Optimal Control of Systems Governed by Partial Differential Equations*. Springer.
- [PP92] Psiak M. L. and Park K. H. (1992) Parallel solver for trajectory optimization search directions. *Journal of Optimization Theory and Applications* 73: 519–546.
- [Ral96] Ralph D. (1996) A parallel method for unconstrained discrete-time optimal control problems. *SIAM Journal on Optimization* 6: 488–512.
- [Wri90] Wright S. J. (1990) Solution of discrete-time optimal control problems on parallel computers. *Parallel Computing* 16: 53–71.
- [Wri91] Wright S. J. (1991) Partitioned dynamic programming for optimal control. *SIAM Journal on Optimization* 1: 620–642.