Advances in Simultaneous Strategies for Dynamic Process Optimization

Lorenz T. Biegler[*], Arturo M. Cervantes and Andreas Wächter
Department of Chemical Engineering
Carnegie Mellon University
Pittsburgh PA 15213, USA.

February 19, 2001

Abstract

Following on the popularity of dynamic simulation for process systems, dynamic optimization has been identified as an important task for key process applications. In this study, we present an improved algorithm for simultaneous strategies for dynamic optimization. This approach addresses two important issues for dynamic optimization. First, an improved nonlinear programming strategy is developed based on interior point methods. This approach incorporates a novel filter-based line search method as well as preconditioned conjugate gradient method for computing search directions for control variables. This leads to a significant gain in algorithmic performance. On a dynamic optimization case study, we show that nonlinear programs (NLPs) with over 800,000 variables can be solved in less than 67 CPU minutes. Second, we address the problem of moving finite elements through an extension of the interior point strategy. With this approach we develop a reliable and efficient algorithm to adjust elements to track optimal control profile breakpoints and to ensure accurate state and control profiles. This approach is also demonstrated on a dynamic optimization for two distillation columns.

Key words: interior point; dynamic optimization; nonlinear programming; moving finite elements; nonlinear programming; collocation

1 Introduction

Over the past decade, applications in dynamic simulation have increased significantly in the process industries. These are driven by strong competitive markets faced by operating companies along with tighter specifications on process performance and regulatory limits. Moreover, the development of powerful commercial modeling tools for dynamic simulation, such as ASPEN Custom

[*]e-mail: lb01@andrew.cheme.cmu.edu. Tel: (412)-268-2232. Fax:(412)-268-7139.

Modeler and gProms, has led to their introduction in industry alongside their widely used steady state counterparts. Dynamic optimization is the natural extension of these dynamic simulation tools because it automates many of the decisions required for engineering studies. Applications of dynamic simulation can be classified into *off-line* and *on-line* tasks. Off-line tasks include:

- Design to avoid undesirable transients for chemical processes, including process startups and shutdowns, handling of upsets and results of severe disturbances, and transitions to different operating points [3],
- Design of distributed (e.g., packed bed) unit operations such as reactors, chromatographic and adsorption separations and detailed heat exchanger models [30],
- Systematic strategies to deal with interactions of process and equipment design, and synthesis and tuning of controllers [39, 31],
- Design and dynamic operation of batch processes, along with interactions resulting from the scheduling of these processes [9],
- Process safety studies and the evaluation of control schemes under abnormal operations [3].

On the other hand, on-line tasks include [1, 10]:

- Nonlinear Model Predictive Control (NMPC) requiring the solution of a dynamic optimization problem with a nonlinear dynamic process model. This problem may be solved off-line to determine trajectory information or on-line to update the model with each new set of inputs,
- System identification with nonlinear process models to identify the states and unmeasured inputs of the process, given measured inputs and outputs,
- Related estimation tasks to identification including data reconciliation and model parameter estimation.

1.1 Methods for dynamic optimization

Chemical processes are modeled dynamically using differential-algebraic equations (DAEs). The DAE formulation consists of differential equations that describe the dynamic behavior of the system, such as mass and energy balances, and algebraic equations that ensure physical and thermodynamic relations. To apply these to dynamic optimization a number of approaches can be taken. In particular, DAE optimization problems can be solved using a variational approach [32] or by various strategies that apply a Nonlinear Programming (NLP) solver to the DAE model.

The indirect or *variational approach* is based on the solution of the first order necessary conditions for optimality that are obtained from Pontryagin's Maximum Principle [32]. For problems without inequality constraints, the optimality conditions can be formulated as a set of differential-algebraic equations. Obtaining a solution to these equations requires careful attention to the boundary conditions. Often the state variables have specified initial conditions and the adjoint variables have final conditions; the resulting two-point boundary value problem (TPBVP) can be addressed with different approaches, including single shooting, invariant embedding, multiple shooting or some discretization method such as collocation on finite elements or finite differences. A review of these approaches can be found in [15]. On the other hand, if the problem requires the handling of active constraints, finding the correct switching structure as well as suitable initial guesses for state and adjoint variables is often very difficult.

Methods that apply NLP solvers can be separated into two groups: the sequential and the simultaneous strategies. In the *sequential methods*, only the control variables are discretized; these techniques are also known as *control variable parametrization* methods. Given initial conditions and a set of control parameters, the process model is integrated with a DAE solver at each iteration. This produces values of the objective function and constraints (at fixed points) which are used by a nonlinear programming solver to find the optimal parameters in the control parametrization. In this formulation the control variables are represented as piecewise polynomials [42, 3] and optimization is performed with respect to the polynomial coefficients. The gradients of the objective function with respect to the control coefficients and parameters are calculated either from sensitivity equations of the DAE system or by integration of the adjoint equations. The sequential approach is a feasible path method; in every iteration the DAE system is solved. However, this procedure is robust only when the system contains stable modes. Otherwise, finding a feasible solution for a given set of control parameters may be difficult. See [15, 42, 39] for a review of these methods.

Multiple shooting serves as a bridge between sequential approaches and simultaneous approaches that are based on a complete discretization of the state and control variables. Here the time domain is partitioned into smaller time elements and the DAE models are integrated separately in each element [11, 27]. Control variables are treated in the same manner as in the sequential approach. Moreover, to obtain gradient information, sensitivities are obtained for both the control variables as well as the initial conditions of the states in each element. Finally, equality constraints are added to the nonlinear program in order to link the elements and ensure that the states are continuous across each element. With this approach, inequality constraints for states and controls can be imposed directly at the grid points. For piecewise constant or linear controls this approximation is accurate enough, but path constraints for the states may not be satisfied between grid points.

1.2 Simultaneous approach to dynamic optimization

In this study we focus on simultaneous approaches. Also known as *direct transcription*, these techniques fully discretize the state and control variables, leading to large-scale NLP problems which usually require special solution strategies [7, 8, 13, 14, 16]. As a result, these methods directly couple the solution of the DAE system with the optimization problem; the DAE system is solved only once, at the optimal point, and therefore can avoid intermediate solutions that may not exist or may require excessive computational effort. Moreover, simultaneous approaches have advantages for problems with path constraints and with instabilities that occur for a range of inputs. Because they can be seen as extensions of robust boundary value solvers, they are able to "pin down" unstable modes in the forward direction by enforcing the appropriate boundary condition. In addition, these methods allow the direct enforcement of state and control variable constraints, at the same level of discretization as the state variables of the DAE system.

On the other hand, there are some disadvantages to the simultaneous approach. First, for optimal control problems where control variables are discretized at the same level as the state variables, there are a number of open questions related to convergence to the solution of the original variational problem. A number of studies have shown (e.g., [33, 18]) that the Karush Kuhn Tucker (KKT) conditions of the simultaneous NLP are consistent with the optimality conditions of the variational problem. However, stability problems remain due to presence of high index constraints and singular arcs. In particular, interesting stability questions arise regarding appropriate discretizations of control and state profiles. Empirical evidence of this instability and practical remedies have been given in [28, 16, 5] and special cases of these have been analyzed rigorously in [6] and [20]. Coupled with this question is the placement of finite elements in order to maintain accuracy of the discretized DAE model and to determine the optimal breakpoint location in the optimal control profile.

A second disadvantage arises from the need to solve large nonlinear programs; specialized methods are required to solve them efficiently. These NLPs are usually solved using variations of Successive Quadratic Programming (SQP) and both full-space and reduced space options exist for these methods. Full-space methods take advantage of the DAE optimization problem structure and the sparsity of the model. They are best suited for problems where the ratio of state variables to control variables is small [7, 8]. Here, second derivatives of the objective function and constraints are usually required, as are measures to deal with directions of negative curvature in the Hessian matrix [7, 24].

On the other hand, in most process engineering problems, the number of state variables is much larger than the number of control variables. For dynamic optimization problems, the number of discretized control variables rarely exceeds a few hundred while the number of state variables could be over a million. For these cases, a reduced space SQP approach (rSQP) can be more efficient. With this approach, either projected Hessian matrices or their quasi-Newton approximations may be used, thus avoiding the necessity of second derivatives. An efficient algorithm can be constructed by decoupling the search direction into its range and null space components.

In [14, 16], we present a simultaneous rSQP algorithm that exploits the sparsity of the DAE system, as well as the almost block diagonal structure of the DAE optimization problem. The DAE system is discretized using collocation on finite elements. The variables are then partitioned into dependent and independent variables in each element. A Newton step for the dependent variables is obtained by solving blocks of equations for each element. For this system, unstable modes are avoided by selecting a numerically stable pivot sequence for the collocation matrix in each element and consequently by selecting state variables as decisions in the partitioning step. This approach has the effect of imposing stabilizing boundary conditions on the system.

With this decomposition, solution and storage of the large collocation matrix is avoided, but the size of the reduced subproblem still contains bounds on all of the variables. Solution of this subproblem with an active-set quadratic programming method can be inefficient and interior point (IP) (or barrier) methods serve as a desirable alternative to eliminate the combinatorial problem of selecting an active set. IP techniques have been applied to the solution of large NLP problems [12, 43, 25, 38] and stem from the classical work on penalty-barrier functions [22]. In these approaches inequality constraints are replaced by logarithmic barrier terms in the objective function and the resulting equality constrained problem is solved with Newton-type methods applied to the optimality conditions. These methods were explored for simultaneous dynamic optimization [16], where the structure of the collocation equations was exploited using the elemental decomposition [14].

In this study we extend this interior point strategy in three ways. First, we improve the interior point algorithm through the use of a preconditioned conjugate gradient (PCG) method to update the control variables. This approach replaces the quasi-Newton step for these variables with a Newton step. Moreover, by supplying the PCG method with directional differenced quantities, the Newton step is calculated even without exact second derivative information for the projected Hessian. Consequently, the resulting interior point method requires far fewer iterations than the quasi-Newton approach in [16].

Second, a novel line search strategy is introduced that is based on a bicriterion minimization, with the objective function and constraint infeasibility as competing objectives. Termed the *filter* approach, this method was recently developed in a trust region context for SQP and other nonlinear programming algorithms [23]. Since then the filter has been adopted for interior point methods as well [40].

Third, we consider the important issue of moving finite elements in the simultaneous approach. Adjusting and adding finite elements is essential for the representation of accurate state profiles and also to optimal breakpoints locations in control profiles. In a previous study [36] a nested strategy was developed with elements adjusted in an outer loop while the control and state variables were optimized in an inner loop. While successful, this approach required repeated nonlinear programming solutions as well as safeguards to deal with nondifferentiable functions. In this study we show that the interior point approach can be extended to deal with moving finite elements quite easily.

In the next section we present some background on the simultaneous optimization approach. Section 3 then follows with a description of the improved interior point algorithm that includes a filter line search as well as preconditioned conjugate gradient solvers. Section 4 demonstrates this strategy through a medium scale case study based on a distillation column optimization. In particular, this approach leads to an efficient algorithm for solving dynamic optimization problems. Section 5 then presents an extension of the interior point approach to deal with moving finite elements and a demonstration of this approach will be seen in Section 6 for the case study in Section 4. Finally, the paper is summarized and conclusions are drawn in Section 7.

2 NLP problem formulation

The general DAE optimization problem can be stated as follows:

$$\min_{z(t),y(t),u(t),t_f,p} \varphi(z(t_f), y(t_f), u(t_f), t_f, p) \tag{1}$$

s.t. Semi-explicit DAE model:

$$\begin{aligned} \frac{dz(t)}{dt} &= F\left(z(t), y(t), u(t), t, p\right) && (2) \\ 0 &= G\left(z(t), y(t), u(t), t, p\right) && (3) \end{aligned}$$

Initial conditions:

$$z(0) = z^0 \tag{4}$$

Point conditions:

$$H_s\left(z(t_s), y(t_s), u(t_s), t_s, p\right) = 0 \qquad \text{for } s \in \{1, \ldots, n_S\} \tag{5}$$

Bounds:

$$\begin{array}{ccccc} z^L & \leq & z(t) & \leq & z^U \\ y^L & \leq & y(t) & \leq & y^U \\ u^L & \leq & u(t) & \leq & u^U \\ p^L & \leq & p & \leq & p^U \\ t_f^L & \leq & t_f & \leq & t_f^U \end{array} \tag{6}$$

where

φ	is a scalar objective function,
F	are the right hand sides of differential equation constraints,
G	are algebraic equation constraints, assumed to be index one,
H_s	are additional point conditions at fixed times t_s,
z	are differential state profile vectors,
z^0	are the initial values of z,
y	are algebraic state profile vectors,
u	are control profile vectors,
p	is a time-independent parameter vector,
t_f	is the final time.

The DAE optimization problem is converted into an NLP by approximating state and control profiles by a family of polynomials on finite elements ($t_0 < t_1 < \ldots < t_{ne} = t_f$). Here, we use a monomial basis representation [2] for the differential profiles, as follows:

$$z(t) = z_{i-1} + h_i \sum_{q=1}^{ncol} \Omega_q \left(\frac{t - t_{i-1}}{h_i} \right) \frac{dz}{dt}_{i,q} \tag{7}$$

where z_{i-1} is the value of the differential variable at the beginning of element i, h_i is the length of element i, $dz/dt_{i,q}$ is the value of its first derivative in element i at the collocation point q, and Ω_q is a polynomial of order $ncol$, satisfying

$$\begin{aligned} \Omega_q(0) &= 0 & \text{for } q = 1, \ldots, ncol \\ \Omega'_q(\rho_r) &= \delta_{q,r} & \text{for } q, r = 1, \ldots, ncol \end{aligned}$$

where ρ_r is the r^{th} collocation point within each element. Here, Radau collocation points are used because they allow us to set constraints easily at the end of each element and to stabilize the system more efficiently if high index DAEs are present. In addition, the control and algebraic profiles are approximated using a similar monomial basis representation which takes the form:

$$y(t) = \sum_{q=1}^{ncol} \psi_q \left(\frac{t - t_{i-1}}{h_i} \right) y_{i,q} \tag{8}$$

$$u(t) = \sum_{q=1}^{ncol} \psi_q \left(\frac{t - t_{i-1}}{h_i} \right) u_{i,q}. \tag{9}$$

Here $y_{i,q}$ and $u_{i,q}$ represent the values of the algebraic and control variables, respectively, in element i at collocation point q. ψ_q is a Lagrange polynomial of order $ncol$ satisfying

$$\psi_q(\rho_r) = \delta_{q,r} \qquad \text{for } q, r = 1, \ldots, ncol.$$

From (7), the differential variables are required to be continuous throughout the time horizon, while the control and algebraic variables are allowed to have discontinuities at the boundaries of the elements. It should be mentioned that with representation (7), the bounds on the differential variables are enforced directly at element boundaries; however, they can be enforced at all collocation points by writing appropriate point constraints (5).

For now we assume that the number of finite elements, ne, and their lengths are pre-determined. Substitution of equations (7)–(9) into (1)–(6) leads to the following NLP.

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{10}$$

$$\text{s.t.} \quad c(x) = 0 \tag{11}$$

$$x^L \leq x \leq x^U \tag{12}$$

where $x = \left(\frac{dz}{dt}_{i,q}, z_i, y_{i,q}, u_{i,q}, t, p\right)^T$, $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and $c : \mathbb{R}^n \longrightarrow \mathbb{R}^m$.

2.1 Reduced-Hessian Successive Quadratic Programming (rSQP).

The NLP problem, (10)–(12), can be solved using an rSQP method [13, 35, 14]. This method is efficient for solving DAE optimization problems, especially when the dimension of the state variables is much larger than that of the control variables ($n \gg n - m$). The efficiency of the solution procedure is also improved by performing matrix factorizations in each element. This allows us to preserve and exploit the structure of the problem and to detect ill-conditioning due to unstable modes in the DAE system. At each iteration k, a search direction d_k is obtained by solving a quadratic approximation of the original problem (10) to (12)

$$\min_{d \in \mathbb{R}^n} \quad g_k^T d_k + \frac{1}{2} d_k^T H_k d_k \tag{13}$$

$$\text{s.t.} \quad c_k + A_k^T d_k = 0 \tag{14}$$

$$x^L \leq x_k + d_k \leq x^U \tag{15}$$

where $g_k = g(x_k)$ is the gradient of f at x_k, $H_k = H(x_k)$ denotes the Hessian of the Lagrangian function at x_k, and $A_k = A(x_k)$ are the gradients of the constraints at iteration k, and $c_k = c(x_k)$.

The variables are partitioned into m dependent (R space) and $n - m$ independent (Q space) variables. The independent variable space occupies the null space of A_k^T. The complete set of variables spans the full space. Note that the control variables and parameters are not necessarily the independent variables. With this partition A takes the form

$$A_k^T = \begin{bmatrix} C_k & N_k \end{bmatrix} \tag{16}$$

where the $m \times m$ basis matrix C_k is nonsingular. Defining the matrices

$$Q_k = \begin{bmatrix} -C_k^{-1} N_k \\ I \end{bmatrix} \qquad R_k = \begin{bmatrix} I \\ 0 \end{bmatrix}, \tag{17}$$

the search direction can be written as

$$d_k = R_k d_R + Q_k d_Q. \tag{18}$$

Note that the matrix Q_k satisfies

$$A_k^T Q_k = 0$$

and is a null-space basis matrix for A_k^T.

The range space direction d_R is now determined by solving

$$d_R = -C_k^{-1} c_k, \tag{19}$$

and the null space direction d_Q is obtained from the following reduced QP subproblem

$$\min_{d_Q \in \mathbb{R}^{n-m}} \quad \left(Q_k^T g_k + Q_k^T H_k R_k d_R\right)^T d_Q + \frac{1}{2} d_Q^T \left(Q_k^T H_k Q_k\right) d_Q \tag{20}$$

$$\text{s.t.} \quad x^L - x_k - R_k d_R \;\leq\; Q_k d_Q \;\leq\; x^U - x_k - R_k d_R. \tag{21}$$

2.2 Elemental decomposition

The partitioning in (18) allows us to perform a special decomposition of the matrix A_k that we will briefly explain. In the remainder of this section we will omit the iteration index k in our notation for simplicity. First, consider the Jacobian of the discretized system of equations.

$$A^T = \begin{bmatrix}
Z^0_{init} & & & & & & & & & & & 0 & & & & 0 \\
Z^0_1 & DZ^1_1 & Y^1_1 & & & & & & & & & U^1_1 & & & & P^1_1 \\
Z^0_2 & DZ^1_2 & Y^1_2 & & & & & & & & & U^1_2 & & & & P^1_2 \\
\vdots & \vdots & \vdots & & & & & & & & & \vdots & & & & \vdots \\
Z^0_{ncol} & DZ^1_{ncol} & Y^1_{ncol} & & & & & & & & & U^1_{ncol} & & & & P^1_{ncol} \\
Z^0_a & D^1_a & Y^1_a & & & & & & & & & U^1_a & & & & P^1_a \\
I & D^1 & 0 & -I & & & & & & & & 0 & & & & 0 \\
 & & & Z^1_1 & DZ^2_1 & Y^2_1 & & & & & & 0 & U^2_1 & & & P^2_1 \\
 & & & & & \ddots & \ddots & & & & & & & \ddots & & \vdots \\
 & & & & & & \ddots & \ddots & & & & & & & \ddots & \vdots \\
 & & & & & & & Z^{ne-1}_{ncol} & DZ^{ne}_{ncol} & Y^{ne}_{ncol} & & & & & U^{ne}_{ncol} & P^{ne}_{ncol} \\
 & & & & & & & Z^{ne-1}_a & D^{ne}_a & Y^{ne}_a & & & & & U^{ne}_a & P^{ne}_a \\
 & & & & & & & I & D^{ne} & 0 & -I & & & & &
\end{bmatrix} \tag{22}$$

where I represents the identity matrix of appropriate size, and D^i is a matrix containing the coefficients of the continuity equations of the i^{th} element. Z^i_q, DZ^i_q, Y^i_q, U^i_q and P^i_q represent the Jacobian of the collocation equations with respect to z_i, $dz/dt_{i,q}$, $y_{i,q}$, $u_{i,q}$ and p, at collocation point q and element i. Z^i_a, D^i_a, Y^i_a, U^i_a and P^i_a, correspond to the Jacobian of the additional constraints. As indicated in (22), it is assumed that these constraints can be separated by elements. The factorization of this matrix is performed over smaller matrices, each one representing a finite element. To explore this decomposition, consider the rows and columns of A^T, corresponding to element i:

$$A^i = \begin{bmatrix} Z_1^{i-1} & DZ_1^i & Y_1^i & 0 & U_1^i & P_1^i \\ Z_2^{i-1} & DZ_2^i & Y_2^i & 0 & U_2^i & P_2^i \\ \vdots & \vdots & \vdots & 0 & \vdots & \vdots \\ Z_{ncol}^{i-1} & DZ_{ncol}^i & Y_{ncol}^i & 0 & U_{ncol}^i & P_{ncol}^i \\ Z_a^{i-1} & D_a^i & Y_a^i & 0 & U_a^i & P_a^i \\ I & D^i & 0 & -I & 0 & 0 \end{bmatrix}. \tag{23}$$

If no parameters p are present, the decomposition of this matrix can be performed directly, as all the variables can be eliminated locally. In the case that a parameter is present, the last column of A^i, which corresponds to the parameters, will be coupled to the entire system. In this case, we create separate dummy parameters for each finite element.

In order to apply a reduced space algorithm, we partition the search direction into the space spanned by A and its null space. This partition is performed by applying an LU factorization with partial pivoting on each rectangular system A^i. Following [19], this LU factorization will yield a dichotomous system in each element. If an unstable mode is present in the DAE, A^i is required to be partitioned so that the end conditions of any increasing mode are fixed or become decision variables. Here, if a differential variable z_j has an increasing mode, $\frac{dz}{dt}_{j,ncol}$ would be specified and would correspond to a column in the null space. Correspondingly, a column corresponding to a control variable or a parameter would be added to the range space. By considering the variables that span the columns of the null space to be fixed, the decomposition approach is equivalent to solving a discretized, linear BVP.

After the basis is selected, we can represent the overall matrix A with the following structure and partition:

$$A^T = \left[\begin{array}{cccccccc|ccc} I & & & & & & & & 0 & & \\ T^1 & C^1 & & & & & & & N^1 & & \\ I & \widehat{C}^1 & -I & & & & & & \widehat{N}^1 & & \\ & & T^2 & C^2 & & & & & & N^2 & \\ & & I & \widehat{C}^2 & -I & & & & & \widehat{N}^2 & \\ & & & & T^3 & C^3 & & & & & N^3 \\ & & & & & \ddots & \ddots & & & & \ddots \end{array}\right] = \left[\begin{array}{c|c} C & N \end{array}\right]$$

and the corresponding right hand sides are

$$c^T = \begin{bmatrix} c^0 & c^1 & \widehat{c}^1 & c^2 & \widehat{c}^2 & c^3 & \cdots \end{bmatrix}.$$

By premultiplying T^i, C^i, and N^i by the inverse of C^i in each element, we can develop a forward decomposition strategy that allows us to calculate $C^{-1}N$ and $C^{-1}c$. In our previous work, the dense matrix $C^{-1}N$ was calculated and stored explicitly. However, in our new implementation for very large problems the sparse LU factors of C^i are stored instead and reused as needed.

This decomposition strategy is very efficient, but as the number of discretized variables increases, the active set solution of the reduced QP subproblem can become a bottleneck [37]. This is the result of the combinatorial problem of choosing an active set from a large number of bound constraints. To address this issue we replace the QP subproblem (20,21) and apply an interior point algorithm directly to the NLP problem. Here the inequality constraints (bounds) are eliminated from the NLP, thus avoiding the need to choose an active set. In the next section we present a detailed description of this algorithm.

3 Interior point method applied to NLP

In order to simplify the presentation of our algorithm we assume that all variables have only lower bounds of zero and consider the following problem:

$$\begin{aligned} \min \quad & f(x) && (24) \\ \text{s.t.} \quad & c(x) = 0 && (25) \\ & x \geq 0 && (26) \end{aligned}$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and $c : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ are assumed to be sufficiently smooth. Extending the approach described below to the general problem formulation (10)–(12) is straight-forward. The algorithm follows a barrier approach, where the bound constraints (26) are replaced by a logarithmic barrier term which is added to the objective function to give

$$\begin{aligned} \min \quad & \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \ln(x^{(i)}) && (27) \\ \text{s.t.} \quad & c(x) = 0 && (28) \end{aligned}$$

with a barrier parameter $\mu > 0$. Here, $x^{(i)}$ denotes the i^{th} component of the vector x. Since the objective function of this barrier problem becomes arbitrarily large as x approaches the boundary of the nonnegative orthant $\{x \,|\, x \geq 0\}$, it is clear that a local solution $x_*(\mu)$ of this problem lies in the interior of this set, i.e., $x_*(\mu) > 0$. The degree of influence of the barrier is determined by the size of μ, and $x_*(\mu)$ converges to a local solution x_* of the original problem (24)–(26) as $\mu \to 0$. Consequently, a strategy for solving the original NLP is to solve a sequence of barrier problems (27)-(28) for decreasing barrier parameters μ_l, where l is the counter for the sequence of subproblems. Since the exact solution $x_*(\mu_l)$ is not of interest for large μ_l, the corresponding barrier problem is solved only to a relaxed accuracy ϵ_l, and the approximate solution is then used as a starting point for the next barrier problem with $\lim_{l\to\infty} \epsilon_l = 0$.

To solve the barrier problem for a fixed value of μ_l we follow a primal-dual approach (see e.g. [21]), that generates search directions for primal variables $x > 0$ as well as for *dual* variables $v > 0$ which correspond to the Lagrange multipliers to the bound constraints (26) as $\mu_l \to 0$.

After defining *dual variables* by $v = \mu X^{-1}e$, the optimality conditions of (27), (28) can be written as:

$$\begin{aligned} \nabla f(x) + A(x)\lambda - v &= 0 && (29)\\ XVe - \mu e &= 0 && (30)\\ c(x) &= 0. && (31) \end{aligned}$$

where the components of the vector λ are the Lagrange multipliers for the equality constraints (28). Throughout this section, e denotes the vector of appropriate dimension of all ones, and a capital letter of a vector name (e.g. X) denotes the diagonal matrix with the vector elements on the diagonal. Solving this system of nonlinear equations by Newton's method is equivalent to solving the following quadratic program at (x_k, λ_k, v_k):

$$\begin{aligned} \min_{d^x \in \mathbb{R}^n} \quad & \nabla\varphi_\mu(x_k)^T d^x + \frac{1}{2}(d^x)^T\left(H_k + \Sigma_k\right)d^x && (32)\\ \text{s.t.} \quad & A_k^T d^x + c_k = 0 && (33) \end{aligned}$$

if the matrix $H_k + \Sigma_k$ is positive definite in the null space of A_k^T. Here $\Sigma_k = X_k^{-1}V_k$ and d^x is the search direction for x. The similarity of this QP to (13)-(15) allows us to employ the decomposition presented in Section 2. As in (18), we partition the overall primal step into a range and null space component, $d^x = Q_k d_Q + R_k d_R$. The range step can be obtained using (19), while d_Q is the solution of the reduced QP (20). The reduced space QP is now unconstrained and its solution can directly be computed as

$$d_Q = -[Q_k^T(H_k + \Sigma_k)Q_k]^{-1}\left(Q_k^T\nabla\varphi_\mu(x_k) + w_k\right). \qquad (34)$$

with

$$w_k = Q_k^T(H_k + \Sigma_k)R_k d_R. \qquad (35)$$

3.1 Solving for the null space step

In our previous study the reduced Hessian $\tilde{B}_k = [Q_k^T(H_k + \Sigma_k)Q_k]$ in (34) was approximated by means of a quasi-Newton method. In order to allow for an efficient update that adapts to changing values of μ, the two terms in the reduced Hessian were separated, $\tilde{B}_k = Q_k^T H_k Q_k + Q_k^T\Sigma_k Q_k$, and only the first term $B_k \approx Q_k^T H_k Q_k$ was estimated by a quasi-Newton method. The explicit computation of the second term is easy since Σ_k is diagonal and readily available, and $B_k + Q_k^T\Sigma_k Q_k$ is then used as an estimate for $\tilde{B}_k$ in (34). In the previous implementation we used both BFGS and SR1 quasi-Newton updates for B_k.

On the other hand a direct solution of (34) with an exact reduced Hessian $[Q_k^T(H_k + \Sigma_k)Q_k]$ would provide a faster convergence rate and could greatly reduce the number of NLP iterations. This can be done in two ways. First, the reduced Hessian can be constructed using finite differences. A slightly more accurate approach is to calculate matrix-vector products of the Hessian of the

Lagrangian, $H_k w$ for the reduced Hessian. For this task, one may use the ADOL-C package [26] applied to the model equations (2)-(3). Second, the linear system (34) can be solved with a preconditioned conjugate gradient (PCG) method. These iterative methods rely on matrix vector products (e.g., $[Q_k^T(H_k + \Sigma_k)Q_k]w$) and many of these products can be substituted by divided differences of first derivatives. For instance the first term in this product can be approximated by:

$$Q_k^T H_k Q_k w \approx \frac{Q_k^T(\nabla L(x_k + tQ_k w) - \nabla L(x_k))}{t}$$

where t is a small scalar. Consequently second derivatives need not be supplied at all.

For efficient performance of the conjugate gradient method, it is essential to supply an effective preconditioner. Here the preconditioning matrix P_k needs to approximate $[Q_k^T(H_k + \Sigma_k)Q_k]^{-1}$. For this task we considered two preconditioners. The first is due to Morales and Nocedal [29] and applies a BFGS update to approximate $B_k \approx [Q_k^T(H_k+\Sigma_k)Q_k]^{-1}$ based on matrix-vector products, $[Q_k^T(H_k+\Sigma_k)Q_k]w$, supplied from the PCG iterations. The BFGS update to P_k is applied at *every PCG iteration*. However, the preconditioner itself is applied once to solve the linear system at x_k. The second preconditioner separates the two terms; here the BFGS update is applied only to the first term $B_k \approx [Q_k^T H_k Q_k]$ and the second $Q_k^T \Sigma_k Q_k$ is calculated directly. Once updated, we have $P_k = (B_k + Q_k^T \Sigma_k Q_k)^{-1}$. Again the BFGS update to B_k is applied at *every PCG iteration* and the preconditioner itself is applied only once to solve the linear system at x_k.

3.2 The filter line search method

Having computed the primal-dual search directions (d_k^x, d_k^v) from (19), (34) and (18), we need to choose a step length $\alpha_k \in (0, 1]$ to obtain the next iterate

$$x_{k+1} = x_k + \alpha_k d_k^x \tag{36}$$

$$v_{k+1} = v_k + \hat{\alpha}_k d_k^v. \tag{37}$$

Among other things, we need to ensure that the implicit positivity constraints $x_{k+1} > 0$ and $v_{k+1} > 0$ are satisfied, since a full step with $\alpha_k = 1$ might violate these constraints. For this, we compute a maximal step sizes $\tilde{\alpha}, \hat{\alpha} \in (0, 1]$ such that the "fraction-to-the-boundary-rule"

$$x_k + \tilde{\alpha} d_k^x \geq (1 - \tau)x_k \tag{38}$$

$$v_k + \hat{\alpha} d_k^v \geq (1 - \tau)v_k \tag{39}$$

with $\tau = 0.99$ is satisfied. Starting from this maximal step size, a suitable value is then determined for α. In [16] we performed an Armijo line search using a primal-dual ℓ_1-penalty function

$$\phi_\nu(x, v) = \varphi_\mu(x) + \mathcal{V}_\mu(x, v) + \nu\theta(x) \tag{40}$$

where

$$\mathcal{V}_\mu(x,v) = x^T v - \mu \sum_{i=1}^{n} \ln(x^{(i)} v^{(i)})$$

and $\theta(x) = \|c(x)\|$. The penalty parameter ν can be updated without the explicit knowledge of the multipliers λ. In this study, we apply a line search strategy based on a filter approach; this leads to larger stepsizes and more robustness. A detailed derivation and convergence analysis of this method can be found in [44]. Here we sketch only some of the basic concepts.

The filter is based on a bicriterion minimization with $\varphi_\mu(x)$ and $\theta(x)$ as competing objectives. Instead of decreasing a linear combination of $\theta(x)$ and $\varphi_\mu(x)$, we require sufficient decrease in only *one* of those objectives. The conditions for sufficient decrease are given by:

$$\varphi_\mu(x_k + \alpha d_k^x) \leq \varphi_\mu(x_k) - \gamma_{\varphi_\mu}[\theta(x_k)] \tag{41}$$

$$\theta(x_k + \alpha d_k^x) \leq \theta(x_k) - \gamma_\theta[\theta(x_k)] \tag{42}$$

where γ_{φ_μ} and γ_θ are small positive constants. Note that the feasibility measure, $\theta(x)$ dominates these conditions. Moreover, to avoid cycling, we store a set of some previous $(\theta(x_k), \varphi_\mu(x_k))$-pairs (called *filter*) and force future points to improve on points in this set. Points that fall in this category are *acceptable to the filter.* Moreover, to avoid convergence to feasible but not optimal points, we require an Armijo-type decrease in $\varphi_\mu(x)$, i.e.:

$$\varphi_\mu(x_k + \alpha d_k^x) \leq \varphi_\mu(x_k) + \alpha\eta\nabla\varphi_\mu(x_k)^T d_k^x. \tag{43}$$

instead of (41), (42) if $\theta(x_k)$ is small. Finally, if no admissible step size α can be found that satisfies the conditions for sufficient decrease, we switch to a *feasibility restoration phase* in which $\theta(x)$ is minimized directly in order to find a less infeasible point.

3.3 Description of the Algorithm

Algorithm:

Given: Initial barrier parameter $\mu_0 > 0$, initial point $x_0 \in \mathbb{R}^n, v_0 \in \mathbb{R}^n$ with $x_0, v_0 > 0$, initial estimate of the reduced Hessian B_0, desired tolerance ϵ, and constants $L \in (0,1)$, $M > 0$.

Initialize iteration counter $k := 0$.

1. Evaluate $f(x_k)$, $\nabla f(x_k)$, c_k, A_k; compute R_k and Q_k from (17).

2. Check convergence of barrier problem

$$\max\left\{\left\|Q_k^T(\nabla f(x_k) - v_k)\right\|_\infty, \|c(x_k)\|_\infty, \|X_k V_k e - \mu_k e\|_\infty\right\} < M\mu_k.$$

   If satisfied,

   (a) Check convergence of original NLP

$$\max\left\{\left\|Q_k^T(\nabla f(x_k) - v_k)\right\|_\infty, \|c(x_k)\|_\infty, \|X_k V_k e\|_\infty\right\} < \epsilon.$$

   (b) If satisfied, STOP [converged].

   (c) Otherwise, decrease barrier parameter $\mu_k \leftarrow L\mu_k$ and go back to step 2.

3. Compute range space step from (19).

4. Compute cross term from (35).

5. Compute null space step from

$$d_Q = -[Q_k^T H_k Q_k + Q_k^T \Sigma_k Q_k]^{-1}\left(Q_k^T \nabla\varphi_\mu(x_k) + w_k\right).$$

   Here a number of options can be used based on quasi-Newton updates or preconditioned conjugate gradient methods.

6. Compute primal search direction d_k^x from (18), and the dual search direction $d_k^v = \mu_k X_k^{-1} e - v_k - \Sigma_k d_k^x$.

7. Do the line search:

   (a) Determine maximal stepsizes $\tilde{\alpha}_k, \hat{\alpha}_k \in (0, 1]$ that satisfies (38)–(39).

   (b) Apply back-tracking linesearch to find stepsize $\alpha_k \leq \tilde{\alpha}_k$ acceptable to the filter mechanism.

8. Accept the new iterate (36)–(37).

9. Increase iteration counter $k \leftarrow k + 1$, set $\mu_k := \mu_{k-1}$, and go back to step 1.

4 Distillation Optimization: A Case Study

In this section we consider the dynamic optimization of a ternary distillation column between two desired periods of operation. This problem was considered in [16] and here we compare the options of the improved interior point algorithm derived in the previous section.

The dynamic model for both cases consists of dynamic MESH equations as described in [16]. The general model of the columns consists of the following equations:

$$\frac{dM_i}{dt} = F_i + V_{i+1} + L_{i-1} - V_i - L_i + \sum_{n=1}^{nr} R_{i,n}. \tag{44}$$

Here, M_i corresponds to the molar holdup on tray i, V_i and L_i are the vapor and liquid flowrates, F_i is the feed flow rate, nr is the number of reactions, and $R_{i,n}$ is the difference between the rates of production and consumption of each reaction n.

The liquid mole fraction x of each component j can be expressed as

$$M_i \frac{dx_{i,j}}{dt} = F_i(z_{i,j} - x_{i,j}) + V_{i+1}(y_{i+1,j} - x_{i,j}) + L_{i-1}(x_{i-1,j} - x_{i,j}) - V_i(y_{i,j} - x_{i,j}) + \widehat{R}_i \tag{45}$$

where

$$\widehat{R}_i = \sum_{n=1}^{nr} \upsilon_{j,n} \frac{M_i}{\rho_i} r_{i,n} - x_{i,j} \sum_{n=1}^{nr} R_{i,n}. \tag{46}$$

ρ_i is the liquid molar density, $z_{i,j}$ is the molar fraction of j in the feed, $y_{i,j}$ is the vapor mole fraction, $\upsilon_{j,n}$ is the the stoichiometric coefficient, and $r_{i,n}$ is the rate of production per unit volume. The phase equilibrium relationship is represented by

$$y_{i,j} = K_{i,j} x_{i,j} \tag{47}$$

$$K_{i,j} = f_j(T_i) \tag{48}$$

while the sum of the vapor fractions on each tray is required to be equal to one

$$\sum_{j=1}^{nc} y_{i,j} = 1. \tag{49}$$

The vapor flowrates are calculated using a modified index one energy balance (see [13])

$$\frac{1}{M_i}\left(V_{i+1}(h^v_{i+1}) + L_{i-1}(h^v_{i-1} - h^l_i) - V_i(h^v_i - h^l_i) + \sum_{n=1}^{nr} \frac{M_i}{\rho_i} r_{i,n} \Delta H^R_n\right) = \text{RHS}_i \tag{50}$$

where

$$\text{RHS}_i = \sum_{j=1}^{nc} \frac{\partial h^l_i}{\partial x_{i,j}} \cdot \frac{dx_{i,j}}{dt} \; - \; \frac{\partial h^l_i}{\partial T_i} \cdot \frac{\sum_{j=1}^{nc} K_{i,j} \frac{dx_{i,j}}{dt}}{\sum_{j=1}^{nc} x_{i,j} \frac{dK_{i,j}}{dT_i}} \; . \tag{51}$$

h^v and h^l are the vapor and liquid enthalpies, and ΔH^R_n is the heat of reaction.

In this example we simulate a simple air separation using a continuous distillation column with 15 trays without the reaction terms. The basic model is given by equations (44)-(51). The feed is assumed to have a composition of 78.1% Nitrogen, 21% Oxygen, and 0.9% Argon. The purity of Nitrogen taken out at the top of the column is 99.8%. The complete model consists of 70 differential equations and 356 algebraic equations. Here, we simulate a change of set point in the distillate flow rate from $D(0) = 301.8 \; mol/s$ to $D^{set} = 256.0 \; mol/s$. The objective is to minimize the offset produced during the change from one steady state to another by controlling the feed flowrate F.

$$\begin{aligned} \min \quad & \int_0^{t_f} (D - D^{set})^2 dt \\ \text{s.t.} \quad & \text{DAE model (44)-(51)} \\ & 0 \; kmol/s \leq F \leq 2 \; kmol/s. \end{aligned} \tag{52}$$

Elements	n/m	QN-BFGS [Iter/CPU]	QN-SR1 [Iter/CPU]	Exact [Iter/CPU]	PCG1 [Iter/CPU]	PCG2 [Iter/CPU]	PCG2* [Iter/CPU]
50	67620/67470	206/5.0	82/1.93	10/7.35	15/3.78	12/1.45	12/5.98
100	135170/134870	238/11.35	124/6.03	9/25.6	15/13.03	12/3.43	12/19.6
300	405370/404470	288/52.47	151/27.65	11/269.98	17/99.93	16/17.93	16/209.98
600	810670/808870	336/210.52	218/133.83	11/366.67	20/280.6	18/66.22	18/949.62

Table 1: Computational Results for Air Separation Optimization

Here we compare the following cases for solution of this problem:

- The *QN-BFGS* option uses a BFGS update to approximate $Q_k^T H_k Q_k$ as in our previous study [16]. In order to obtain the overall reduced Hessian in (34), the term $Q_k^T \Sigma_k Q_k$ is computed explicitly.
- Similarly, the *QN-SR1* option uses the SR1 update to approximate $Q_k^T H_k Q_k$ as in our previous study [16]. Here the overall reduced Hessian is modified if the update is not positive definte.
- The *exact* option uses finite difference approximations to calculate $Q_k^T H_k Q_k$.
- The *PCG1* option due to [29] constructs a preconditioner from a BFGS update. Here $P_k = B_k \approx \left[Q_k^T(H_k + \Sigma_k)Q_k\right]^{-1}$.
- The *PCG2* option applies a damped BFGS for $B_k \approx Q_k^T H_k Q_k$. The preconditioner is constructed from $P_k = \left[B_k + Q_k^T \Sigma_k Q_k\right]^{-1}$.

We note that for this example only control variable bounds are imposed on this problem. As a result $Q_k^T \Sigma_k Q_k = \Sigma_u$, which is a diagonal matrix corresponding to the control bounds and their multipliers, is much cheaper to evaluate. This accounts for the fast performance in options *PCG2, QN-BFGS* and *QN-SR1.* To present more realistic performance behavior (here for the *PCG2* case) for general control problems we also provide the calculation for a complete $Q_k^T \Sigma_k Q_k$, even though it is not used. This option is labeled as $PCG2^*$.

All of these cases were run on an 800 MHz Pentium III Intel processor and were initialized to a feasible solution corresponding to the steady state at $t = 0$. Table 1 shows the computational results for this example for three collocation points and for different numbers of elements.

From the results in this table several features are worth noting. First it is clear that the Newton-based options (*exact, PCG1, PCG2* and *PCG2**) require far fewer interior point iterations than the quasi-Newton methods. In most cases this factor exceeds an order of magnitude. However, the Newton methods clearly require more work per iteration, with the exact approach incurring a large cost due to $n - m$ of Hessian vector products for IP iteration. An interesting feature also can be seen with the two *PCG* options. The first preconditioner is four to six times slower than the second one. This can be explained because the first preconditioner requires about an order of magnitude more *PCG* iterations and Hessian vector products. *PCG2* requires less work because its preconditioner has separated terms which are updated more accurately, particularly when μ changes. On the other hand, *PCG2* (as well as *QN-BFGS* and *QN-SR1*) is greatly aided by the simple form of $Q_k^T \Sigma_k Q_k$ for this particular example. For more generally constrained problems, $Q_k^T \Sigma_k Q_k$ is more costly to update. This can be seen with option *PCG2**, which requires the same low number of *PCG* iterations but is now about four times as expensive as *PCG1*.

Nevertheless, we see that the improved IP algorithm is quite attractive for solving very large nonlinear programming problems. As can be seen from the table, even the largest problem with over $800,000$ variables and 1800 degrees of freedom could be solved in less than 67 CPU minutes.

5 Moving Finite Elements in an Interior Point Strategy

In this section we extend the interior point algorithm to include moving finite elements. The sizes of the finite elements are governed primarily by accuracy and the optimal location of the control variable breakpoints. By introducing the element lengths α_i as variables and aggregating the variable bounds as inequality constraints (written as $Q(z_i, y_{i,j}, u_{i,j}, p) \leq 0$) for convenience, the nonlinear programming problem takes the form:

$$\min \varphi(z_{ne}, y_{ne,ncol}, u_{ne,ncol}, t_f, p) \tag{53}$$

Semi-explicit DAEs

$$\begin{array}{rl} s.t. & \frac{dz}{d\tau}_{i,j} = F(z_{i-1}, \frac{dz}{d\tau}_{i,j}, y_{i,j}, u_{i,j}, p)\alpha_i \quad i = 1, ..., ne, \quad j = 1, ..., ncol \\ & G(z_{i-1}, \frac{dz}{d\tau}_{i,j}, y_{i,j}, u_{i,j}, p) = 0 \quad i = 1, ..., ne \quad j = 1, ..., ncol \end{array} \tag{54}$$

Continuity equations

$$z_i = z_{i-1} + h_i^o \sum_{j=1}^{ncol} \Omega_j(1) \frac{dz}{d\tau}_{i-1,j} \qquad i = 1, ..., ne \tag{55}$$

Initial conditions

$$z_0 = z(0) \tag{56}$$

Final time constraints

$$\sum_{i=1}^{ne} h_i = t_f \tag{57}$$

Point conditions

$$\widehat{H}_s(z_{i-1}, \frac{dz}{d\tau}_{i,j}, y_{i,j}, u_{i,j}, p) = 0 \tag{58}$$

Bound constraints

$$Q(z_i, y_{i,j}, u_{i,j}, p) \leq 0 \quad i = 1, ..., ne \quad j = 1, ..., ncol \tag{59}$$

where τ is defined as

$$\tau = \frac{t - t_{i-1}}{\alpha_i} \tag{60}$$

and

$$\alpha_i = \frac{h_i}{h_i^o} \tag{61}$$

where h_i^o is the initial mesh for h_i.

This problem has been addressed before in different ways. The initial approaches considered the solution of this problem in a single stage [41]. Here the element lengths were considered as decision variables, error constraints are imposed in each finite element and a larger nonlinear problem was solved. The error constraints guarantee the accuracy of the discretization while variable elements lengths locate the optimal breakpoints of the control variables. However, the main difficulties of this approach are that the resulting NLP problem is more nonlinear than with fixed elements, and the error constraints induce inconsistencies in the linearization if the problem is poorly initialized. In [41] this strategy required good problem initializations to be successful.

To overcome this difficulty, Tanartkit and Biegler [36] proposed a bi-level strategy in which the solution of an outer problem determines the element lengths. Then, the solution of an inner problem (fixed mesh) determines the control and state variables. With this approach, the solution of a highly nonlinear problem is avoided. However, the outer problem is non-smooth because the active constraints sets in the inner problem can change from one mesh to another. Thus, an additional procedure is added to maintain the descent property of the search direction. Here, a bundle procedure that adds cutting planes at the nondifferentiabilities is implemented by generating directional derivatives as local underestimators.

In this study, we develop a new approach that minimizes the nonlinearities created by the introduction of the elements lengths and at the same time is not affected by the discontinuities of the profiles. The basic engine of the algorithm is the interior point method. A few modifications allow us to take advantage of the approximate solutions generated after each barrier problem is solved.

The main idea of the algorithm is to detect active constraints during the solution procedure and obtain an answer where the active set does not change within any finite element. With a sufficient number of finite elements to insure accuracy of the state profiles, a constant active set in each of the finite elements is often enough to determine an (arbitrarily) accurate optimal control profile. To ensure this, we apply the procedure used by Tanartkit and Biegler [36].

By forming the Lagrange function of (53), (59) and simplifying the expression through integration by parts, the following optimality conditions can be derived for this NLP:

$$\forall_i : \sum_{i=1}^{ncol} F(z_{ij}, u_{ij}, p)^T \lambda_{ij} = -\eta \tag{62}$$

$$\forall_{i,j(j\neq 0)} : \frac{\partial F^T}{\partial u}(z_{ij}, u_{ij}, p)\lambda_{ij} + \frac{\partial G^T}{\partial u}(z_{ij}, u_{ij}, p)\mu_{ij} + \frac{\partial Q^T}{\partial u}(z_{ij}, u_{ij}, p)\pi_{ij} = 0 \tag{63}$$

$$\forall_{i,j(j\neq 0)} : \sum_{k=0}^{ncol} \lambda_{ik}\dot{\psi}_k(\tau_j) + h_i \frac{\partial F^T}{\partial z}(z_{ij}, u_{ij}, p)\lambda_{ij} + \frac{\partial G^T}{\partial z}(z_{ij}, u_{ij}, p)\mu_{ij} + h_i \frac{\partial Q^T}{\partial z}(z_{ij}, u_{ij}, p)\pi_{ij} = 0 \tag{64}$$

The above conditions (62), (63) and (64) are always satisfied by the NLP solver. Moreover, if the active constraint set remains the same within each element the multipliers can be approximated by a piecewise polynomial in each element. Consequently, for a sufficient number of elements, the construction of an algorithm that guarantees no changes in the active set within each element should be sufficient to obtain an accurate optimal control profile. Our condition for this control profile is a (discretized) Hamiltonian function that remains constant over time, i.e.:

$$\forall_{i,j} : F^T(z_{ij}, u_{ij}, p)\lambda_{ij} = constant \tag{65}$$

5.1 Main Algorithm

The main idea of the algorithm is presented in Figure 1. Given a fixed number of elements, each of length h_i.

1. Set $\mu = \mu^o$, tolerance $\epsilon = \epsilon^o$ and fix all elements to normalized lengths $\alpha_i = 1$.

2. Solve the barrier problem for given μ to given tolerance ϵ.

3. Check convergence of NLP. If $\epsilon \leq \epsilon_{NLP}$, stop.

4. For each element i, check for active constraints (bounds) for each discretized state and control variable, represented by x_{ij} at collocation point j, according to the following test:

   $$max\ [\|x_{ij} - x_{ij}^L\|, \|x_{ij} - x_{ij}^U\|] \leq \vartheta \cdot \epsilon. \tag{66}$$

   (Here we used $\vartheta = 0.01$)

5. Check for changes in active constraints within any two consecutive elements. If there are no changes or they are already identified go to 8.

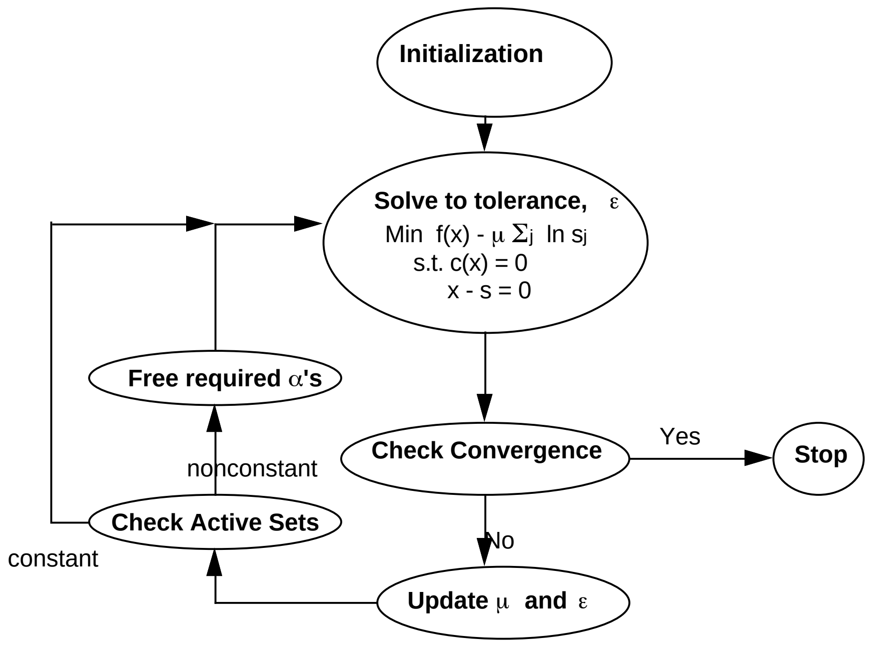

Figure 1: Flowchart for moving finite element elements

6. Otherwise, free the element lengths as variables, α_i, that correspond to consecutive elements in which a change of active set occurs.

7. Initialize additional terms in the reduced Hessian matrix for the new decision variables, α_i, to identity. Go to 2.

8. Decrease μ and ϵ according to the barrier method algorithm in Section 3. Go to 2.

This algorithm has the following important characteristics:

- The nonlinear program with a fixed mesh is not solved completely in the inner loop. This makes the algorithm more efficient than the bi-level strategy proposed by Tanartkit [36], in which each inner loop has to be solved to a tight tolerance.

- Because this is no longer a nested formulation the influence of the new variables α_i in the problem is continuous and differentiable. As a consequence, a nonsmooth algorithm used for the nested inner/outer strategy in [36] is no longer needed.

- Because the algorithm introduces new variables α_i whenever elements are found with active set changes, it should converge to solutions that do not allow changes in the active set within an element.

5.2 Element Addition

The above algorithm assumes that a sufficient number of elements is used for the discretization of the DAE system. To check that the number of elements is large enough to guarantee optimality and accuracy, two additional tests have to be performed.

1. Error constraints for the state profiles need to be imposed and satisfied.
2. From optimal control theory, the Hamiltonian function should remain constant for the whole time horizon.

These two tests are performed only after the DAE optimization problem is solved for a given number of elements. According to our experience, a good initial mesh selection with a sufficient number of elements is not difficult to find and additional elements are rarely required. The mesh selection can usually be obtained by comparison of the state profiles with those obtained with an initial value DAE integrator at the base point for the control profiles.

5.2.1 Error constraints.

To ensure accurate state profiles for the first test, different kinds of error constraints can be used, as discussed in [36]. The most common methods include extrapolation techniques and calculation of residuals bounds. For the first group, the problem is solved twice, once for the given mesh size and a second time for a finer mesh, usually with twice as many elements. Then, the error is calculated by comparing the two solutions. Although this technique is the most accurate and most commonly used, it is also the most expensive.

A less accurate but more efficient error constraint can be imposed by calculating the residual equations $r(t_{nonc})$ of the DAE system at non-collocation points t_{nonc} (residual bounds). The residual-based error estimates are incorporated into the algorithm as the following constraint:

$$\|r(t_{nonc})h\| \leq \epsilon \tag{67}$$

which has to be satisfied but is not enforced directly in the NLP. We check the constraint satisfaction by calculating the residuals for each equation, including algebraic equations, at non-collocation points in each element. If this constraint is not satisfied for a given element, the element can be divided into two and the problem resolved, using the previous solution as the starting guess. According to our experience, this can often be avoided with a good initial mesh selection.

5.3 Hamiltonian Function

Analogous to optimal control theory, a discretized Hamiltonian function can be defined as

$$\mathcal{H}_{ij} = \Gamma_{ij}^T f(z_{ij}, u_{ij}, p) + \mu_{ij}^T G(z_{ij}, u_{ij}, p) \tag{68}$$

where $\Gamma = \begin{bmatrix} \lambda \\ \pi \end{bmatrix}$ and $f = \begin{bmatrix} F \\ Q \end{bmatrix}$

The last term is exactly equal to zero because of the complementarity condition. From optimal control theory, for an autonomous system to be optimal, the Hamiltonian is constant:

$$\forall_{i,j} : \mathcal{H}_{ij} = \text{constant} \tag{69}$$

The computation of the Hamiltonian function can be performed using the same technique used by Tanartkit and Biegler [36]. From the KKT conditions of the NLP

$$\Gamma = -\nabla_x c^{-T} \left(\nabla_x \varphi + v\right) \tag{70}$$

where Γ is a vector of multipliers associated with the states variables, v are the multipliers associated with variable bounds, and φ is the objective function. Moreover, constraint multipliers are not required to calculate products with Γ. Instead, we multiply (70) by f^T

$$f^T\Gamma = -f^T \nabla_x c^{-T} \left(\nabla_x \varphi + v\right) = -\left(\nabla_x \varphi + v\right)^T \nabla_x c^{-1} f \tag{71}$$

and use the discretized Hamiltonian definition (68)

$$\mathcal{H}_{ij} = -\left(\nabla_x \varphi + v\right)^T \nabla_x c^{-1} f_{ij}, \tag{72}$$

where f_{ij} is a vector of the DAE right-hand sides at element i and collocation point j (with all other vector entries equal to zero). This leads to the following expression for the original NLP (53):

$$f_{ij} = \begin{bmatrix} 0 \\ \vdots \\ F(z_{i-1}, \frac{dz}{d\tau}_{i,j}, y_{i,j}, u_{i,j}, p)\alpha_i \\ Q(z_{i-1}, \frac{dz}{d\tau}_{i,j}, y_{i,j}, u_{i,j}, p) \\ \vdots \\ 0 \end{bmatrix}. \tag{73}$$

The computation of $\mathcal{H}_{ij}$ involves the solution of a linear system of equations at each finite element i and collocation point j. As the only change is in the right-hand side, the solution of the system can be performed efficiently using the elemental decomposition described in Section 2.

If the computed Hamiltonian function is not constant for all the finite elements and collocation points, the solution is not optimal, and the mesh should be refined. Here we used the same criteria used by Tanartkit and Biegler [36] for a constant Hamiltonian:

$$\left\| \frac{d\mathcal{H}(t_{ij})}{dt} \right\| \leq tol \tag{74}$$

where tol is the desired tolerance defined as $max_{ij}(0.001, 0.001 \left\|\mathcal{H}_{ij}\right\|)$. The location of any additional elements is determined via heuristics by dividing the element which has the highest $d\mathcal{H}/dt$.

6 Moving Element Examples

In this section we revisit three process problems solved in [16]. We first start with a simple batch reactor to illustrate the procedure and then follow with a batch reactive distillation. We then conclude with the distillation optimization problem from Section 4. Because of previous interior point implementations we used the QN-BFGS option with a merit function substituted for the filter line search. In addition to these examples we note that a much larger example was solved with this moving finite element algorithm. This example represents an optimal grade transition for a low density polyethylene process and consists of 532 DAEs and over 83,800 discretized variables. More details of this case study are reported in [17].

6.1 Batch reactor

We consider a small batch reactor [16], where the following reactions take place.

$$A \rightarrow B \rightarrow C$$

The objective is to maximize the mole fraction of B at a given final time, by controlling the reactor temperature T:

$$\begin{aligned} \max \quad & x_B(t_f) && (75)\\ \text{s.t.} \quad & \frac{dx_A}{dt} = -k_1(T)x_A && (76)\\ & \frac{dx_B}{dt} = k_1(T)x_A - k_2(T)x_B && (77)\\ & 0 = x_A + x_B + x_C - 1 && (78)\\ & 680K \leq T \leq 750K. && (79) \end{aligned}$$

Using the moving element interior point algorithm, we solved the problem with 6 finite elements and 3 collocation points. We initialize the problem with a feasible solution for a constant temperature value. After 16 iterations the problem is solved to a tolerance $\epsilon = 0.1$, and the algorithm detects two changes in active sets for the control variable. The first change in active set is located between the first and second elements and the second one between the fourth and fifth elements. The objective function value at this point is 0.539231. The algorithm then frees α_1, α_2, α_4, and α_5. The algorithm continues without any other change in active sets until it satisfies the desired tolerance. The final value of the objective function is 0.539267. The algorithm converged in 50 iterations and 1.23 s of CPU time on a 400 MHz DEC Alpha workstation. The optimal profiles are presented in Figure 2.

In order to demonstrate the optimality of the control profile the Hamiltonian function was computed and can be seen in Figure 3. The Hamiltonian remains constant through the time horizon when the given α_i's are free. This can be compared with the Hamiltonian obtained for all fixed α_i's. As it is shown in Figure 3 no elements need to be added after moving the required α_i's.

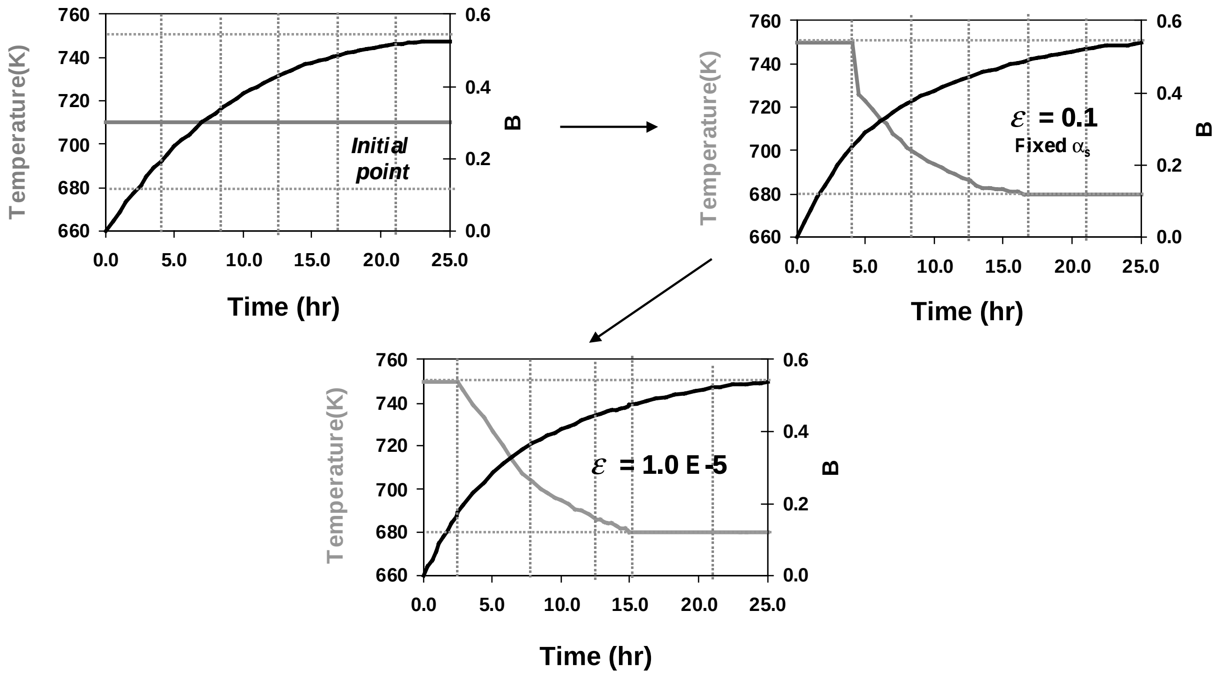

Figure 2: Element placement for batch reactor problem

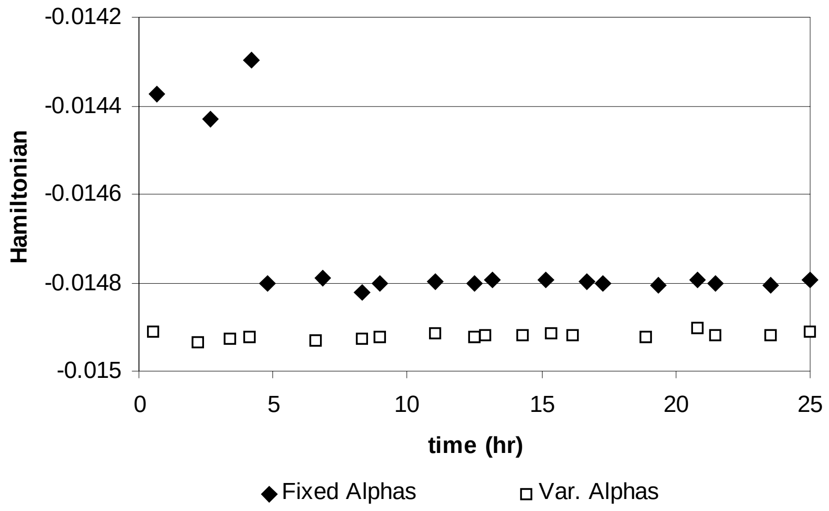

Figure 3: Hamiltonian Function for Batch Reactor

6.2 Batch reactive distillation

This example considers the reversible reaction between acetic acid and ethanol as described in [16],

$$CH_3COOH + CH_3CH_2OH \leftrightarrow CH_3COOCH_2CH_3 + H_2O.$$

The model consists of 13+5nt differential and 6+5nt algebraic equations, where nt is the number of trays which we set to 8 in this study. The column, in all cases, was fed with an equimolar mixture of ethanol, acetic acid, ethyl acetate and water. The objective is to maximize the amount of distillate D produced within one hour by manipulating the reflux ratio as a function of time, as follows:

$$\begin{aligned} \max \quad & \int_0^{t_f=1} D dt \\ \text{s.t.} \quad & \text{DAE model (44)-(51)} \\ & x_D^{Ester} \geq 0.4800. \end{aligned}$$

This example considers the reversible reaction between acetic acid and ethanol described [16]. We solved the problem using 7 finite elements and 3 collocation points. The results are shown in Figure 4. After reaching a tolerance of 0.1 (76 iterations) the algorithm detects a break point between elements 2 and 3. The value of the objective function at this point is 6.7965. After 131 iterations and 15.6 s of CPU time, the algorithm reaches the desired tolerance. The final optimal control profile is a perfect bang-bang and the objective function value is 6.8308.

When the Hamiltonian function is calculated for fixed and free α_i's, it can be seen (Figure 5) that it is constant through the time horizon and no additional elements are needed.

6.3 Case Study Revisited

In this section we revisit the distillation optimization described in Section 4, see (52). As described in [16], we observe underdamped oscillations in the control profile. This leads to an interesting study for moving finite elements.

We started the algorithm with just 12 elements and 3 collocation points, with a feasible solution and a constant value for the control variable. We also use fewer elements than in Section 4 in order to test the algorithm. The results are presented in Figure 6. The algorithm reaches a tolerance of 0.1 after 23 iterations and 17.4 s of CPU time with an objective function value of 17.4260. After freeing α_1 and α_2, the algorithm continues and after a total of 36 iterations and 40.2 s the desired tolerance is reached. The value of the objective function is 17.3498. These calculations were done on a DEC Alpha 400 and the examples were initialized to a feasible solution and the CPU time includes this computation.

The Hamiltonian function was calculated when the required α_i's are free. However, a nonconstant Hamiltonian (Figure 7) is obtained with 12 elements. For this reason, elements 2 and 4 are partitioned and the problem is resolved using the previous solution as the starting point.

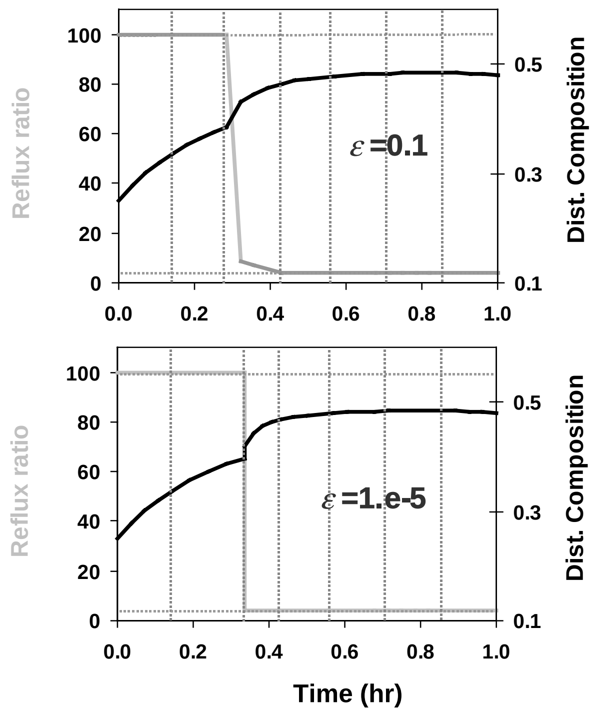

Figure 4: Element Placement Batch Reactive Distillation

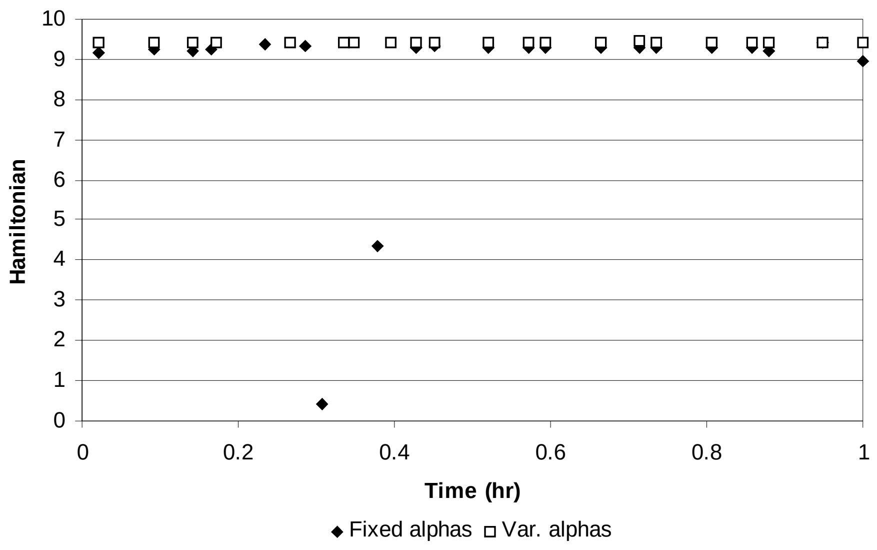

Figure 5: Hamiltonian Function for Batch Reactive Distillation

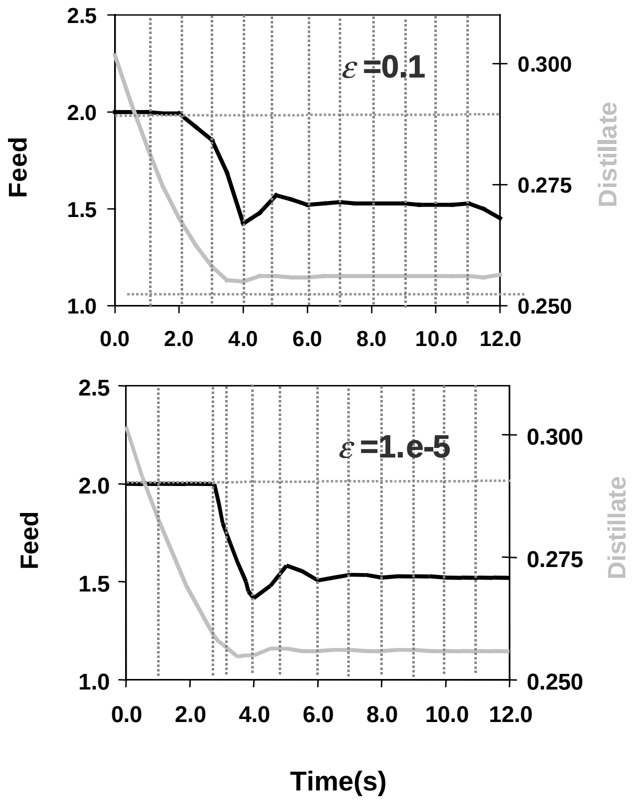

Figure 6: Optimal Feed Profile for Air Separation Column with 12 elements

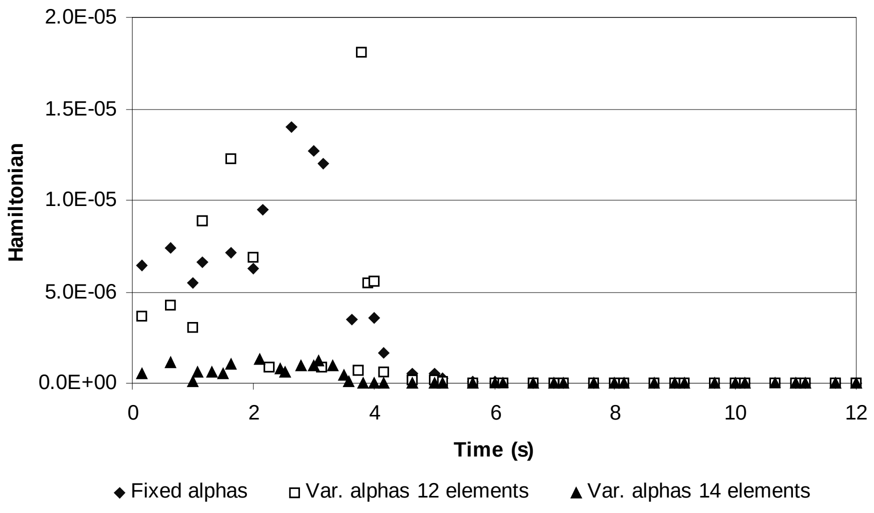

Figure 7: Hamiltonian Function for Air Separation Column

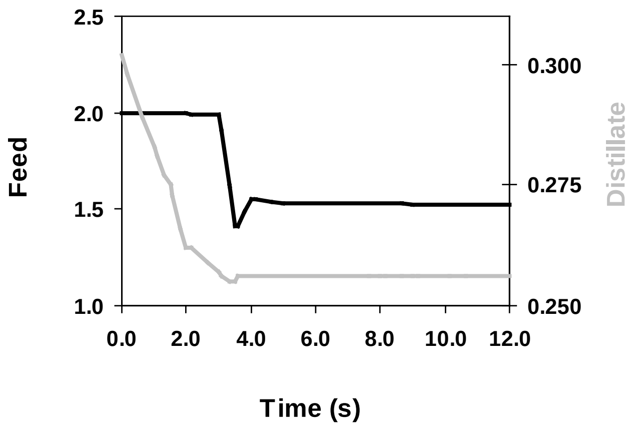

Figure 8: Optimal Feed Profile for Air Separation Column with 14 elements

The algorithm now converges in 29 iterations and a CPU time of 29.5 s. The total CPU time for the solution of the problem is then 69.7 s and the total number of iterations 65. The Hamiltonian using 14 elements is now constant and the objective function value is 17.3329. The optimal control profiles are shown in Figure 8.

7 Conclusions

We have presented a new reduced space NLP interior point algorithm which has proved to be efficient and well suited for the solution of DAE optimization problems. This is especially the case for large-scale problems and when a large number of bounds are active, as this approach eliminates the necessity of choosing an active set. Here, an efficient implementation allows us to solve problems with more than 800,000 variables in less than 67 CPU minutes on an 800 MHz computer. This algorithm incorporated two new features: solution of exact Newton steps for null space directions and incorporation of a novel filter line search. As shown in the case study, this approach led to significant performance improvements over our earlier algorithm. In addition, we have incorporated the sizes of one elements as additional decision variables for a moving finite element strategy. Here we develop a heuristic strategy that is embedded within the interior point algorithm. Moreover, by using an estimate of the Hamiltonian function, one can determine the optimum control profile to an arbitrary level of accuracy.

Finally, we are currently studying convergence properties of the improved IP method. These issues will be addressed in a separate paper [44]. With this approach, we believe that there are a number of very challenging problems that can now be addressed by simultaneous methods. These include problems with discrete decisions as well as multistage applications of dynamic optimization. While aspects of these problems have already been addressed in part by sequential approaches, there is much more scope for more efficient and reliable optimization algorithms in this area.

Acknowledgments

Funding from the the Universidad Nacional Autónoma de México, the National Science Foundation (CTS9729075, DMS9706950, and CCR9875559) and the Center for Advanced Process Decision-making is gratefully acknowledged. The authors thank Professor Jorge Nocedal from Northwestern University for his insightful comments on the interior point method.

References

[1] J. Albuquerque, V. Gopal, G. Staus, L. T. Biegler and B. E. Ydstie, Interior Point SQP Strategies for Structured Process Optimization Problems, *Computers and Chemical Engineering.* **21** (1997) 283.

[2] G. Bader and U. Ascher, A New Basis Implementation for Mixed Order Boundary Value ODE Solver, *SIAM J. Sci. Comput.* **8** (1987) 483–500.

[3] P.I. Barton and R.J. Allgor and W.F. Feehery and S. Galan, "Dynamic optimization in a Discontinuous World," *Ind. Eng. Chem. Res.*, 37, p. 966-981 (1998)

[4] Barton, P. and T. Park, "Analysis and control of combined discrete/continuous systems: Progress and challenges in the chemical process industries," *AIChE Symposium Series*, 93 (316), p. 102, (1997)

[5] Bausa, J. and G. Tsatsaronis, "Discretization Methods for Direct Collocation of Optimal Control Problems," submitted to *Comp. Chem. Engr.* (2001)

[6] Betts, J., N. Biehn and S. L. Campbell, "Convergence of nonconvergent IRK discretizations of optimal control problems with state inequality constraints," Technical Report, Department of Mathematics, North Carolina State University (2000)

[7] J. T. Betts and W. P. Huffman, Application of Sparse Nonlinear Programming to Trajectory Optimization, *J. Guidance, Dyn. Cont.* **15** (1992) 198.

[8] J. T. Betts and P. D. Frank, A Sparse Nonlinear Optimization Algorithm, *J. Opt. Theo. Applics.* **82** (1994) 543.

[9] Bhatia, T. and L. T. Biegler, "Dynamic Optimization in the Design and Scheduling of Multiproduct Batch Plants," *I & EC Research*, 35, 7, p. 2234, (1996)

[10] Biegler, L. T., "Efficient Solution of Dynamic Optimization and NMPC Problems," pp. 219-245, Nonlinear Model Predictive Control, F. Allgoewer and A. Zheng (eds.), Birkhaeuser, Basel (2000)

[11] H.G. Bock and K.J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," 9th IFAC World Congress, 1984, Budapest

[12] R. H. Byrd, M. E. Hribar and J. Nocedal, An Interior Point Algorithm for Large Scale Nonlinear programming, *SIAM J. Opt.* **9** (1999) 877–900

[13] Cervantes, A. and L. T. Biegler, Large-Scale DAE Optimization using Simultaneous Nonlinear Programming Formulations, *AIChE Journal* **44** (1998) 1038.

[14] Cervantes, A. and L. T. Biegler, A Stable Elemental Decomposition for Dynamic Process Optimization, *Journal of Computational and Applied Mathematics* 120, pp. 41-57 (2000)

[15] Cervantes and L. T. Biegler, "Optimization Strategies for Dynamic Systems," accepted for publication, *Encyclopedia of Optimization*, C. Floudas and P. Pardalos (eds.), Kluwer (1999)

[16] Cervantes, A. M., A. Wächter, R. Tütüncü and L. T. Biegler, A Reduced Space Interior Point Strategy for Optimization of Differential Algebraic Systems, *Comp. Chem. Engr.* , 24, pp. 39-51 (2000)

[17] Cervantes, A. M., S. Tonelli, A. Brandolin, J. A. Bandoni and L. T. Biegler, "Large-Scale Dynamic Optimization for Grade Transitions in a Low Density Polyethylene Plant," submitted for publication (2000)

[18] Cuthrell, J.E. and L.T. Biegler, "Simultaneous Optimization and Solution Methods for Batch Reactor Control Profiles," Computers and Chemical Engineering 13, 1/2, p. 49 (1989).

[19] R. De Hoog and M. Mattheij. On Dichotomy and Well Conditioning in BVP, *SIAM J. Numer. Anal.* **24** (1987) 89–105.

[20] Dontchev, A., W. Hager and V. Veliov, "Second order Runge Kutta approximations in constrained optimal control," *SIAM J. Num. Anal.*, 38, p. 202 (2000)

[21] A. S. El-Bakry, R. A. Tapia, T. Tsuchiya, and Y. Zhang, On the Formulation and Theory of the Newton Interior-Point Method for Nonlinear Programming, *J. Optim. Theory App.* **89** (1995) 507–541.

[22] Fiacco, A. and G. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, New York (1968).

[23] Fletcher, R. and S. Leyffer, "Nonlinear programming without a penalty function," Technical Report NA/171, University of Dundee Numerical Analysis Report, Dundee, Scotland, UK (1998)

[24] Forsgren, A., and P. E. Gill, "Primal-Dual Interior Point Methods for Nonconvex Nonlinear Programming," UCSD Dept. of Math., Tech. Report NA-3 (1996)

[25] Gay D. M., M. L. Overton and M. H. Wright, A Primal-Dual Interior Method for Nonconvex Nonlinear Programming, in Y. Yuan, ed., "Proceedings of the 1996 International Conference on Nonlinear Programming, Beijing, China", Kluwer Academic Publishers, Dordrecht, The Netherlands (1998) 31–56.

[26] Griewank A., D. Juedes, and J. Utke, ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++, *TOMS* 22(2) (1996), pp. 131-167

[27] Leineweber, D. B., "Efficient Reduced SQP Methods for the Optimization of Chemical Processes Described by Large Sparse DAE Models", University of Heidelberg, 1999, Heidelberg, Germany

[28] Logsdon, J.S. and L.T. Biegler, "On the Accurate Solution of Differential-Algebraic Optimization Problems, I & EC Research, 28, p. 1628 (1989).

[29] Morales, J. L. and J. Nocedal, "Automatic Preconditioning by Limited Memory Quasi-Newton Updating," *SIAM J. Optimization*, 10,4, pp. 1079-1096 (2000)

[30] S. Nilchan and C. Pantelides, "On the optimization of periodic adsorption processes," *Adsorption*, 4, 113, (1998)

[31] Pistikopoulos, E. N., and V. Sakizlis, "Simultaneous Design and Control Optimization under Uncertainty in Reaction/Separation Systems," Chemical Process Control VI, AIChE Symposium Series, J. B. Rawlings and B. A. Ogunnaike (eds.), to appear (2001)

[32] V. V. Pontryagin, Boltyanskii, R. Gamkrelidge and E. Mishchenko, *The Mathematical Theory of Optimal Processes.* Interscience Publishers Inc. New York, NY (1962).

[33] Reddien, G. W., "Collocation at Gauss points as a discretization in optimal control," *SIAM J. Cont. Opt.*, 17, p. 298 (1979)

[34] C. A. Ruiz, M. S. Basualdo and N. J. Scenna, Reactive Distillation Dynamic Simulation, *TransIChemE*, **73** (1995) 363–378.

[35] P. Tanartkit and L. T. Biegler, Stable Decomposition for Dynamic Optimization, *Ind. Eng. Chem. Res.* **34** (1995) 1253–1266.

[36] P. Tanartkit and L.T. Biegler, "A Nested Simultaneous Approach for Dynamic Optimization Problems II: The Outer Problem," *Computers Chem. Eng.*, 21, p. 1365 (1997)

[37] D. Ternet and L. T. Biegler, Recent Improvements to a Multiplier-Free Reduced Hessian Quadratic Programming Algorithm, *Comp. and Chem. Eng.* **22** (1997) 963.

[38] R. J. Vanderbei and D. F. Shanno, *An interior point algorithm for nonconvex nonlinear programming*, Technical Report SOR-97-21, CEOR, Princeton University, Princeton, NJ, 1997.

[39] J. van Schijndel and E.N. Pistikopoulos, "Towards the Integration of Process Design, Process Control and Process Operability - Current Status and Future Trends", Foundations of Computer Aided Process Design 99, AIChE Symposium Series No. 323, M. Malone, J. Trainham, B. Carnahan (eds.), pp. 99-112 (2000)

[40] Ulbrich, M., S. Ulbrich and L. Vicente, "A globally convergent primal dual interior point filter method for nonconvex nonlinear programming," Technical Report TR00-12, Rice University (2000)

[41] Vasantharajan, S. and L.T. Biegler, "Simultaneous Strategies for Parameter Optimization and Accurate Solution of Differential-Algebraic Systems," *Computers and Chemical Engineering,* 14, 8, p. 1083 (1990)

[42] V. Vassiliadis.*Computational Solution of Dynamic Optimization Problems with General Differential Algebraic Constraints.* (PhD Thesis, University of London, London, UK., 1993).

[43] H. Yamashita, H. Yabe and T. Tanabe, "A globally and superlinearly convergent primal-dual interior point trust region method for large scale constrained optimization" Technical Report, Mathematical Systems, Inc., Tokyo, Japan, July, 1997 (revised July, 1998)

[44] Wächter, A. and L. T. Biegler, "Line Search Filter Methods for Nonlinear Programming," Technical Report, CAPD, Carnegie Mellon University (2001)