FAST SOLVERS AND RIGOROUS MODELS: CAN BOTH BE ACCOMMODATED IN NMPC?

Victor M. Zavala, Carl D. Laird and Lorenz T. Biegler

Chemical Engineering Department
Carnegie Mellon University
Pittsburgh, PA USA 15213
{*vzavala, claird, lb01*}*@andrew.cmu.edu*

Abstract: In less than two decades, Nonlinear Model Predictive Control (NMPC) has evolved from a conceptual framework to an attactive, general approach for the control of constrained nonlinear processes. These advances were realized both through better understanding of stablity and robustness properties as well as improved algorithms for dynamic optimization. This study focuses on recent advances in optimization formulations and algorithms, particularly for the simultaneous collocation-based approach. Here we contrast this approach with competing approaches for on-line application and discuss further advances to deal with applications of increasing size and complexity. To address these challenges we adapt the real-time iteration approach developed in the context of a multiple shooting (Diehl *et al.*, 2006; Bock *et al.*, 2006) to a collocation-based approach with a full space nonlinear programming solver. We show that straightforward sensitivity calculations from the KKT system also lead to a real-time iteration strategy, with both shifted and non-shifted variants. This approach is demonstrated on a large-scale polymer process, where on-line calculation effort is reduced by about two orders of magnitude.

Keywords: nonlinear model predictive control, nonlinear programming, collocation, finite elements, sensitivity analysis

1. INTRODUCTION

Nonlinear Model Predictive Control (NMPC) has evolved over the past decade into an efficient general purpose method for process control of large systems. This approach has the key advantage that it is a general purpose multivariable control strategy that can handle constrained, nonlinear systems directly. On the other hand, efficient dynamic optimization strategies are needed for on-line, time critical solutions.

Recent developments in large-scale nonlinear programming (NLP) algorithms have enabled the on-line solution of these problems. In particular, in a recent IFAC workshop, numerous industrial applications have been presented including contributions from Exxon Mobil (Bartusiak, 2006), BASF (Nagy *et al.*, 2006) and ABB (Franke and Doppelhamer, 2006). In addition, to enabling NLP solvers, there is also a much better understanding of NMPC stability properties and associated dynamic optimization problem formulations that provide them (Camacho and Bordons, 2006). With these theoretical developments, NMPC robustness properties have also been developed and analyzed (Magni and Scattolini, 2006).

Moreover, with the ability to solve dynamic optimization problems on-line, the separation between model predictive control and real-time optimization tasks begin to disappear. A comprehensive treatment of dynamic real-time optimization is provided in (Groetschel *et al.*, 2001), and it is clear that with improved optimization formulations and algorithms, the role of NMPC can be greatly expanded. To realize this capability, performance issues also need to be emphasized in addition to stability properties. Related to these issues are:

- consideration of cost objectives into the NMPC problem (Toumi *et al.*, 2005; Groetschel *et al.*, 2001) instead of the typical deviation from a reference value,
- providing for longer time horizons with additional constraints and degrees of freedom to improve the cost objective,
- incorporation of multiple operating stages within the predictive horizon. These may include transitions in the predictive horizon to product change-overs, nonstandard cyclic operations, or anticipated shutdowns.

To bring these issues into practice, it is clear that much larger dynamic optimization problems need to be considered for on-line solution. Because these still remain time-critical steps, much more is demanded from enabling algorithms and their implementations. In fact, the need to consider these dynamic optimization applications leads to the challenging and difficult task of maintaining controller stability and performance. In particular, both of these properties are strongly affected by the issue of feedback delay. A frequent assumption is that on-line optimization for NMPC must be performed quickly relative to the process dynamics. If not, both the performance and stability characteristics deteriorate. The former was noted (Santos *et al.*, 2001) in an NMPC implementation of a laboratory reactor as well as in numerous industrial studies. Deterioration of stability was noted in (Findeisen and Allgöwer, 2004), where a detailed stability analysis is provided. To address this issue, Diehl, Bock and coworkers (Bock *et al.*, 2006; Diehl *et al.*, 2006) developed real-time iteration strategies where the optimization strategy is separated into *preparation* and *feedback response* phases, where the latter represents the on-line cost that affects the feedback delay. This approach was developed in the context of a multiple-shooting strategy and Successive Quadratic Programming (SQP). The current study relies closely on this previous work. It applies a similar strategy, but with a fully simultaneous optimization strategy based on collocation on finite elements and a sparse barrier NLP algorithm.

The next section provides a brief overview of *off-line* dynamic optimization strategies. Here we assess these computational strategies in terms of computational cost and complexity, and focus on characteristics of the simultaneous approach. The third section then discusses the NLP solver, IPOPT (Wächter and Biegler, 2006), as well as its adaptation to NLP sensitivity. These sensitivity calculations provide the basis for a fast partial solution strategy for NMPC, which is developed in the fourth section. The fifth section provides a demonstration of this particular real-time iteration approach on a large-scale industrial polymer process, with shifted and non-shifted variants, while the last section concludes the paper and presents areas for future work.

2. OFF-LINE SOLUTION OF DYNAMIC OPTIMIZATION PROBLEMS

For the purpose of this study, we consider the optimization problem stated in the following form:

$$Min \quad \sum_{k=1}^{N} \varphi(z(t_k), u_k) \tag{1a}$$

$$s.t. \ \frac{dz_k(t)}{dt} = f(z_k(t), y_k(t), u_k), \quad t \in [t_{k-1}, t_k] \tag{1b}$$

$$g(z_k(t), y_k(t), u_k) = 0 \tag{1c}$$

$$z_k(t_{k-1}) = z_{k-1}(t_{k-1}) \tag{1d}$$

$$u_k^L \leq u_k(t) \leq u_k^U \tag{1e}$$

$$y_k^L \leq y_k(t) \leq y_k^U \tag{1f}$$

$$z_k^L \leq z_k(t) \leq z_k^U \tag{1g}$$

$$k = 1, \ldots, N$$

where $z_k(t) \in \Re^{n_x}$ is the vector of state variables, $u_k \in \Re^{n_u}$ is the vector of manipulated variables, and $y_k(t) \in \Re^{n_y}$ is a vector of algebraic variables. These are functions of the scalar "time" parameter $t \in [t_0, t_f]$. As constraints we have the differential and algebraic equations (DAEs) (1b)-(1c) which we assume without loss of generality are index one.

A number of approaches can be taken to solve (1a) - (1g). Until the 1970s, these problems were solved using an *indirect* or *variational approach*, based on the first order necessary conditions for optimality obtained from Pontryagin's Maximum Principle (Pontryagin *et al.*, 1962; Bryson and Ho, 1975). For problems without inequality constraints, these conditions can be written as a two-point boundary value problem (TPBVP) which can be addressed with a number of solution strategies; a review of these approaches can be found in (Cervantes and Biegler, 2000). However, if the problem requires the handling of active inequality constraints, finding the correct switching structure as well as suitable initial guesses for state and adjoint variables

is often very difficult. This limitation has made the *indirect* approach less popular for NMPC applications.

On the other hand, direct methods that apply NLP solvers can be separated into two groups, *sequential* and the *simultaneous* strategies. In *sequential methods*, also known as *control vector parameterization*, the control variables are discretized as u_k. Often they are represented as piecewise polynomials (Vassiliadis *et al.*, 1994*a*; Vassiliadis *et al.*, 1994*b*; Barton *et al.*, 1998) and optimization is performed with respect to these controls. Given initial conditions and a set of control parameters, the DAE model is solved for $k = 1, \dots N$ within the inner loop of the NLP solver; the control variables are then updated by the NLP solver itself. Gradients of the objective function with respect to the control coefficients and parameters are calculated either from direct sensitivity equations of the DAE system or by integration of the adjoint equations. Several efficient codes have been developed for both sensitivity methods including DDASAC, DASPK and CVODES.

Sequential strategies are relatively easy to construct and to apply as they contain the components of reliable DAE solvers (e.g., DASSL, DASOLV, DAEPACK) as well as NLP solvers (NPSOL, SNOPT). On the other hand, repeated numerical integration of the DAE model is required, which may become time consuming for large scale problems. Moreover, it is well known that sequential approaches have properties of single shooting methods and cannot handle open loop instability (Ascher and Petzold, 1998; Flores-Tlacuahuac *et al.*, 2005). Finally, path constraints can be handled only approximately, within the limits of the control parameterization.

Multiple shooting is a simultaneous approach that inherits many of the advantages of sequential approaches. As seen in (1a) - (1g), the time domain is partitioned into N smaller time elements and the DAE models are integrated separately in each element (Bock, 1983; Bock and Plitt, 1984; Leineweber, 1999). Control variables are parametrized as in the sequential approach and gradient information is obtained for both control variables as well as the initial conditions of the states variables in each element. Finally, the equality constraints (1d) are added in the NLP to link the elements and ensure that the states are continuous across each element. As with the sequential approach, bound constraints for states and controls can be imposed directly at the grid points t_k. For piecewise constant or linear controls this approximation is accurate, but bounds for the states may be violated between grid points.

In the simultaneous approach, also known as *direct transcription*, we discretize both the state and control profiles in time using collocation on the finite elements $k = 1, \dots N$. This approach corresponds to a particular implicit Runge-Kutta method with highest order accuracy and excellent stability properties. Also known as fully implicit *Gauss* forms, these methods are usually too expensive (and rarely applied) as initial value solvers. However, for boundary value problems and optimal control problems, which require implicit solutions anyway, this discretization is an inexpensive way to obtain accurate solutions. On the other hand, the simultaneous approach also leads to large-scale NLP problems that require efficient optimization strategies (Betts and Huffman, 1992; Biegler *et al.*, 2002; Jockenhövel *et al.*, 2003). 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. In the simultaneous approach the control variables are discretized at the same level as the state variables and, under mild conditions, (see (Reddien, 1979; Cuthrell and Biegler, 1989; Hager, 2000; Kameswaram and Biegler, 2005)) the Karush Kuhn Tucker (KKT) conditions of the simultaneous NLP are consistent with the optimality conditions of the discretized variational problem, and convergence rates can be shown. Moreover, as with multiple shooting approaches, Simultaneous Approaches can deal with instabilities that occur for a range of inputs. Finally, simultaneous 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.

Nevertheless, simultaneous strategies require the solution of large nonlinear programs, and specialized methods are required to solve them efficiently. These NLPs are usually solved using either full or reduced space versions of Successive Quadratic Programming (SQP). While reduced space SQP subproblems are very simular to their sequential and multiple shooting counterparts, full space methods take advantage of the sparsity of the DAE optimization problem. They are best suited when the number of discretized control variables is large (Betts and Huffman, 1992). Moreover, second derivatives of the objective function and constraints are needed, as are measures to deal with directions of negative curvature in the Hessian matrix (Wächter and Biegler, 2006). Betts (Betts, 2001) provides a detailed description of the simultaneous approach with full space methods, along with mesh refinement strategies and case studies in mechanics and aerospace.

Table 1. Computational Complexity/NLP Iteration (with n_w state (differential and algebraic) variables, n_u manipulated variables, N time steps, $\alpha = 2-3$, $\beta = 1-2$)

	Sequential	Multiple Shooting	Simultaneous
i	$n_w^\beta N$	$n_w^\beta N$	—
ii	$n_w n_u N^2$	$n_w(n_w+n_u)N$	$(n_w+n_u)N$
iii	$n_w n_u^2 N^3$	$n_w(n_w+n_u)^2 N$	$(n_w+n_u)N$
iv	—	$n_w^3 N$	—
v	$(n_u N)^\alpha$	$(n_u N)^\alpha$	$((n_w+n_u)N)^\beta$
vi	—	—	$(n_w+n_u)N$

2.1 Comparison of Dynamic Optimization Strategies

Table 1 lists the complexity of the major algorithmic steps for dynamic optimization of (1a) - (1g) using the sequential, multiple shooting and simultaneous strategies. While a detailed comparison is often problem dependent, this table allows a brief overview of the computation effort for each method as well as a discussion of distinguishing features.

- *(i) Integration:* To solve the DAE system, the sequential and multiple shooting methods invoke a DAE solver that integrates forward in time and solves nonlinear equations at each time step. The integration is performed with a Newton solver invoked at each time step, and often with a sparse matrix routine embedded within the Newton solver. Sparse factorization of the Newton step occurs at a cost that scales little more that linearly with problem size. For the simultaneous approach, this step is part of the optimization steps (v and vi).
- *(ii) Sensitivity:* Both multiple shooting and sequential approaches obtain reduced gradients through direct sensitivity calculations of the DAE system. While this calculation is often implemented efficiently, the cost scales with the number of inputs times the size of the DAE system since previous factorizations can be reused. With the sequential approach, the number of inputs is $n_u N$; with multiple shooting sensitivity is calculated separately in each time step and the number of inputs is $n_w + n_u$. For the simultaneous approach the gradient calculation (through automatic differentiation) scales with the problem size.
- *(iii) Exact Hessian, (iv) Decomposition:* Step (iii) may be optional as NMPC often uses a Gauss-Newton approximation for second derivatives. This is often a reasonable choice for quadratic objectives and requires minimal additional computational cost. However, for cost-based objectives, exact second derivatives are preferred and extend from the sensitivity calculation. For both multiple shooting and sequential approaches the cost of reduced Hessians scales roughly with the number of inputs times the *sensitivity* time. In addition, multiple shooting executes a *decomposition* step which requires projection of the Hessian to dimension $n_u N$, through the factorization of *dense* matrices. For the simultaneous approach the Hessian is very sparse and its calculation (through automatic differentiation) scales with the problem size.
- *(v), (vi) Active Sets, Factorization and Step Determination:* Both multiple shooting methods require the solution of a quadratic program (QP) with $n_u N$ variables, and dense constraint and reduced Hessian matrices. These require factorizations (cubic complexity) and updates (quadratic complexity) to solve the QP. The QP also chooses an active constraint set, which is a combinatorial step. Nevertheless, choosing the active set is often accelerated in the QP by a "warm start." As seen in the next section, the simultaneous approach considered here applies a barrier approach where the active set is determined from the solution of nonlinear (KKT) equations through a Newton method. The corresponding Newton step is obtained through factorization of a sparse linear system (v) and a backsolve step (vi). On the other hand, warm starts may be difficult to implement with the barrier approach.

Table 1 shows that as the number of inputs $n_u N$ increases, one sees a significant advantage to the colllocation-based simultaneous approach; these complexity advantages are aided significantly by the barrier NLP solver. To conclude this section, we briefly describe another feature, the real-time iteration NMPC approach recently developed with multiple shooting.

2.2 On-line vs. Background Calculations

The *real time iteration* strategy was developed in (Diehl *et al.*, 2005*b*) in order to overcome off-line computational costs for dynamic optimization. This approach was developed in the context of multiple shooting, but the stability analysis (Diehl *et al.*, 2005*b*; Diehl *et al.*, 2005*a*) is general and applies to simultaneous and (open-loop stable) sequential approaches as well. In this approach the calculations are separated into *preparation, feedback response* and *transition* steps. Here, if the NLP cannot be solved in one sampling instant, it is solved in "background" for an initial condition "close" to the measured (or estimated) state. Once this state is obtained, a perturbed QP is solved to update the NLP solution.

For the multiple shooting approach, the perturbed QP can be formulated in two ways. To solve (1a)-

(1g) for the ℓ^{th} sampling time, the initial condition is $z_1(t_0) = \hat{z}(\ell)$, the measured state. In a *non-shift* strategy, an N-element QP is derived from the solution of this problem but with the initial condition, $z_1(t_0)$ perturbed to $\hat{z}(\ell+1)$, the measured state at the next sampling time. On the other hand, the *shift* strategy considers the solution of a perturbed QP with $N-1$ elements. Here the the initial condition is $z_2(t_1) = \hat{z}(\ell+1)$ because the first element is discarded, and gradient and Hessian information for the next elements are shifted backward, with control variable values replicated for the N^{th} element. In both cases, the only online cost is the solution of the reduced QP itself (item (v) in Table 1).

This study considers a similar separation of online and background calculations for the collocation-based approach. As seen from Table 1 this simultaneous approach has different bottleneck steps, especially with respect to barrier NLP algorithms and sparse linear solvers. These are explored in the next two sections.

3. NLP ALGORITHM AND SENSITIVITY

To develop the real-time iteration framework, we first consider methods for the solution of the NLP resulting from the simultaneous formulation. The discretized problem derived from (1a)-(1g) is represented by:

$$Min\ f(x,p) \tag{2a}$$
$$s.t.\ \ c(x,p) = 0 \tag{2b}$$
$$x_L \le x \le x_U \tag{2c}$$

with the parameter vector p. As this is a large-scale NLP with many inequalities and a potentially large number of degrees of freedom, we apply the IPOPT algorithm (Wächter and Biegler, 2006) for its efficient solution. The algorithm follows a barrier approach, where the bound constraints are replaced by logarithmic barrier terms and added to the objective function to give:

$$\begin{aligned} \min \varphi(x,p) = f(x,p) &- \mu \sum_{i=1}^{n} \ln(x^{(i)} - x_L^{(i)}) \\ &- \mu \sum_{i=1}^{n} \ln(x_U^{(i)} - x^{(i)}) \qquad (3) \\ s.t. \qquad & c(x,p) = 0 \end{aligned}$$

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 $x^{(i)}$ approaches either of its bounds, a local solution $x_*(\mu)$ of this problem lies in the interior of this set, i.e., $x_U > x_*(\mu) > x_L$. The degree of influence of the barrier is determined by the size of μ, and under mild conditions $x_*(\mu)$ converges to a local solution x_* of the original problem (2) as $\mu \to 0$. Consequently, a strategy for solving the original NLP is to solve a sequence of barrier problems (3), with index l, for decreasing values of μ_l.

IPOPT follows a primal-dual barrier approach and applies a Newton method to the KKT conditions derived from (3), leading to solution of the following sparse linear system at iteration j:

$$\begin{bmatrix} W_j & A_j & -I \\ A_j^T & 0 & 0 \\ V_j & 0 & X_j \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} \nabla f(x_j) + A_j \lambda_j - \nu_j \\ c(x_j) \\ X_j V_j e - \mu_j e \end{bmatrix} \tag{4}$$

where we use the convention, $X = diag(x)$, $V = diag(\nu)$, W_j is the Hessian of the Lagrangian function $\nabla_{xx} f(x_j) + c(x_j)^T \lambda_j$, and $A_j = \nabla c(x_j)$. IPOPT solves this system by first solving the smaller symmetric system that results from eliminating the last block row. Exact first and second derivatives for this method can be evaluated in a number of ways, including automatically through the AMPL interface (Fourer *et al.*, 1993). As a result, local convergence of the Newton method is fast and global convergence is promoted by a novel filter line search strategy. More information on IPOPT can be found in (Wächter and Biegler, 2006).

3.1 IPOPT and NLP sensitivity

The barrier NLP algorithm also allows for consideration of perturbed or parametric NLPs. Here we summarize a set of well-known results both for convergence of the barrier method as well as related sensitivity calculations. We first relate the solutions from the IPOPT algorithm to the solution of (2). Following this we discuss the properties of sensitivity information from IPOPT.

Property 1. (Properties of the barrier trajectory). Consider problem (2) with $p = p_0$ and let $f(x, p_0)$ and $c(x, p_0)$ be at least twice differentiable. Let $x(p_0)$ be a local constrained minimizer of (2) with the following sufficient optimality conditions at $x(p_0)$:

(1) $x(p_0)$ is a KKT point,
(2) the Linear Independence Constraint Qualification (LICQ) holds,
(3) strict complementarity holds at $x(p_0)$ for the bound multipliers ν^* satisfying the KKT conditions.
(4) there exists $\omega > 0$ such that $q^T W(x(p_0), \lambda(p_0))q \ge \omega \|q\|^2$ for equality constraint multipliers $\lambda(p_0)$ satisfying the KKT conditions and all nonzero $q \in \Re^{n_x}$ satisfying LICQ.

If we now solve a sequence of problems (3) with $\mu_l \to 0$, then:

- there is at least one subsequence of unconstrained minimizers $(x(\mu_l))$ of the barrier function converging to $x(p_0)$
- for every convergent subsequence, the corresponding sequence of barrier multiplier approximations is bounded and converges to multipliers satisfying the KKT conditions for $x(p_0)$.
- a unique, continuously differentiable vector function $x(\mu)$ of the minimizers of (3) exists for $\mu > 0$ in a neighborhood of $\mu = 0$
- $\lim_{\mu \to 0^+} x(\mu) = x(p_0)$.
- $\|x(\mu_\ell) - x(p_0)\| = O(\mu_\ell)$.

Proof: The proof follows by noting that LICQ implies MFCQ and invoking Theorem 3.12 and Lemma 3.13 in (Forsgren *et al.*, 2002).

This property indicates that nearby solutions of (3) provide useful information for bounding properties for (2) for small positive values of μ.

For the parametric NLP problem (2) with a solution at $p = p_0$ we would like to compute the sensitivities $\frac{dx(p_0)}{dp}$ and a perturbed solution $\Delta x = \frac{\partial x(p_0)}{\partial p}^T (p - p_0)$.

Property 2. (Sensitivity Properties). For problem (2) assume that $f(x, p)$ and $c(x, p)$ are m times differentiable in p and $m + 1$ times differentiable in x. Also, let the assumptions of Property 1 hold for problem (2) with $p = p_0$, then at the solution:

- $x(p_0) = x(p_0)$ is an isolated minimizer and the associated multipliers $\lambda(p_0)$ and ν^* are unique.
- for some p in a neighborhood of p_0 there exists an m times differentiable function $s(p)^T = [x(p)^T \ \lambda(p)^T \ \nu(p)^T]$ that corresponds to a locally unique minimum for (2) and $s(p_0) = s^*$.
- for p near p_0 the set of binding inequalities is unchanged and complementary slackness holds.

Proof: The result follows directly from Theorem 3.2.2 and Corollary 3.2.5 in (Fiacco, 1983).

We now relate sensitivity results between (3) and (2) with the following result.

Property 3. (Barrier Sensitivity Properties). For problem (3) assume that $f(x, p)$ and $c(x, p)$ are m times differentiable in p and $m+1$ times differentiable in x. Also, let the assumptions of Property 1 hold for problem (2), then at the solution of (3) with a small positive μ:

- $x(\mu, p_0)$ is an isolated minimizer and the associated barrier multipliers $\lambda(\mu, p_0)$ and $\nu(\mu, p_0)$ are unique.
- for some p in a neighborhood of p_0 there exists an m times differentiable function $s(\mu, p)^T = [x(\mu, p)^T \ \lambda(\mu, p)^T \ \nu(\mu, p)^T]$ that corresponds to a locally unique minimum for (3).
- $\lim_{\mu \to 0, p \to p_0} s(\mu, p) = s^*$.

Proof: The result follows from Theorem 6.2.1 and Corollary 6.2.2 in (Fiacco, 1983). These were originally proved for a mixed penalty function but the proofs are easily modified to deal with barrier functions.

Calculation of the sensitivities now proceeds from the implicit function theorem applied to the optimality conditions of (3) at p_0. Defining the quantities;

$$M(s(\mu, p_0)) = \begin{bmatrix} W(s(\mu, p_0)) & A(x(\mu, p_0)) & -I \\ A(x(\mu, p_0))^T & 0 & 0 \\ V(p_0) & 0 & X(p_0) \end{bmatrix} \quad (5)$$

$$N_p(s(\mu, p_0)) = \begin{bmatrix} \nabla_{x,p} L(s(\mu, p_0)) \\ \nabla_p c(x(\mu, p_0)) \\ 0 \end{bmatrix} \quad (6)$$

we see that if the assumptions of Property 1 hold, $M(s(\mu, p_0))$ is nonsingular and the sensivities can be calculated from:

$$\frac{ds(\mu, p_0)}{dp}^T = -M(s(\mu, p_0))^{-1} N_p(s(\mu, p_0). \quad (7)$$

For small values of μ and $\|p - p_0\|$ it can be shown from the above properties (Fiacco, 1983) that

$$s(p, \mu) = -M(s(\mu, p_0))^{-1} N_p(s(\mu, p_0)(p - p_0) + O\|p - p_0\|^2 \quad (8)$$

$$s(p, 0) = s(p, \mu) + O(\mu) \quad (9)$$

3.2 Sensitivity Implementation with IPOPT

The sensitivity calculation in (8) is inexpensive, and requires a only single factorization of $M(s(\mu, p_0))$ as well as a backsolve for each element in p. Furthermore, the implementation of this calculation is straightforward in the current IPOPT framework. The IPOPT algorithm requires the solution of (4) at each iteration. Note that $M(s(\mu, p_0))$ is directly related to the matrix in (4) at the solution.

The current version of IPOPT has been designed to explicitly separate the linear algebra implementation from the fundamental algorithm code. This separation, coupled with the similarities between $M(s(\mu, p_0))$ and the matrix in (4) allows an efficient implementation that can reuse an internal instance of the primal-dual solver object. After solving the NLP, the online cost of finding the

approximate solution of a perturbed problem is only a single backsolve.

The primary purpose of separating the linear algebra implementation from the fundamental algorithm code was to allow specialized linear algebra decomposition strategies for large-scale structured problems. Dynamic optimization problems produce an almost block diagonal structure in (4) and specialized decomposition strategies exist to exploit this structure. The sensitivity calculation, since it builds on the same framework, is also independent of the particular linear algebra implementation. Therefore, the sensitivity calculation is capable of using any specialized internal decomposition strategy developed for IPOPT, ensuring that it is not the bottleneck in a particular NMPC implementation.

4. FAST NMPC BASED ON IPOPT SENSITIVITY

We now combine concepts of the previous two sections to develop a real-time iteration approach for simultaneous dynamic optimization. In a manner similar to (Bock *et al.*, 2006), we classify the NMPC calculation into *off-line, background* and *on-line* components.

For the off-line component, we determine the number of finite elements and collocation points required for accurate solution of the DAE system. While this task is often problem specific and empirical, it is aided by off-line simulations and is not difficult. In addition, the barrier parameter in IPOPT can be tuned to improve performance (Jockenhövel *et al.*, 2003). For instance, μ can remain at a small fixed value for all iterations. This ensures feasibility and robustness by retreating from variable bounds, and it also allows for 'warm starts' for successive NLP solutions.

For the background component, we assume that the NLP can be solved within no more than a few sampling times, but that feedback delay needs to be minimized. As a result, we perform items (ii)-(v) in the background for the simultaneous approach and fully converge the solution with the simultaneous approach. As seen in the previous section, the simultaneous approach is generally faster than other competing approaches. Here the dominant calculation is the repeated factorization of the large sparse matrix in (4). Solution of (4) requires a further cheap backsolve, which is performed on-line.

Once a measurement is obtained, the on-line update of the solution vector requires a single backsolve (vi). As in the multiple shooting studies (Bock *et al.*, 2006; Diehl *et al.*, 2005*a*), this cost of the on-line component is orders of magnitude less than the background component. As seen in (8) the solution error in the perturbed update is small; it scales quadratically with the perturbation.

We now consider the development of this real-time iteration approach with non-shifted and shifted variants, similar to the multiple shooting counterpart. The NLP for the ℓ^{th} horizon of an NMPC problem is represented by:

$$\begin{aligned} P(\ell) \qquad \min \quad & g_f(z_{N+1}) + \sum_{k=1}^{N} g_k(z_k, u_k) \\ \text{s.t. } & z_{k+1} = z_k + \mathbf{B} y_k, \quad k = 1, \dots N \\ & h(z_k, y_k, u_k) = 0, \quad k = 1, \dots N \\ & z_1 = \hat{z}(\ell) = p_0 \end{aligned}$$

where we define $y_k \in \Re^{n_y}$ as a vector of intermediate variables and $\mathbf{B}$ denotes a projection matrix between variables y_k and z_k. This formulation allows a general Runge-Kutta discretization, including multiple shooting or collocation on (one or many) finite elements between sampling times. Here $\hat{z}(\ell)$ is the measured or estimated state of the plant at time t_ℓ. At the solution of problem $P(\ell)$, we provide the control $\hat{u}(\ell) := u_1$ to the plant. Also, all inequalities in the NLP are replaced by appropriate barrier terms in the objective function.

Having the solution of $P(\ell)$ we would like to use this solution for $P(\ell+1)$. To derive the sensitivity result, the Lagrange function associated to $P(\ell)$ is given by,

$$\begin{aligned} L = & g_f(z_{N+1}) + \lambda_1^T (z_1 - p_0) \\ & + \sum_{k=1}^{N} [g_k(z_k, u_k) + \lambda_{k+1}^T (z_{k+1} - z_k - \mathbf{B} y_k) \\ & + \gamma_k^T h(z_k, y_k, u_k)] \end{aligned} \tag{10}$$

and the corresponding optimality conditions are,

$$\left.\begin{array}{c} \nabla_x g_k - \lambda_{k+1} + \lambda_k + \nabla_x h_k \gamma_k = 0 \\ \nabla_y h_k \gamma_k - \mathbf{B}^T \lambda_{k+1} = 0 \\ \nabla_u g_k + \nabla_u h_k \gamma_k = 0 \\ z_{k+1} = z_k + \mathbf{B} y_k \\ h(z_k, y_k, u_k) = 0 \end{array}\right\} k = 1, \dots, N$$

$$\begin{aligned} z_1 &= p_0 \\ \nabla_x g_f + \lambda_{N+1} &= 0. \end{aligned} \tag{11}$$

Linearizing the optimality conditions at the solution of $P(\ell)$, we obtain,

$$
\begin{aligned}
&L^k_{zz}\Delta z_k + L^k_{zu}\Delta u_k + H^k_{zy}\Delta y_k + \nabla_x h_k \Delta\gamma_k \\
&-\Delta\lambda_{k+1} + \Delta\lambda_k = 0, \\
&H^k_{yz}\Delta z_k + H^k_{yu}\Delta u_k + H^k_{yy}\Delta y_k \\
&-\mathbf{B}^T\Delta\lambda_{k+1} + \nabla_y h_k \Delta\gamma_k = 0, \\
&L^k_{uz}\Delta z_k + L^k_{uu}\Delta u_k + H^k_{uy}\Delta y_k + \nabla_u h_k \Delta\gamma_k, \\
&+\Delta z_{k+1} - \Delta z_k - \mathbf{B}\Delta y_k = 0 \\
&\nabla_x^T h_k \Delta z_k + \nabla_u^T h_k \Delta u_k + \nabla_y^T h_k \Delta y_k = 0, \\
&\qquad\qquad k = 1, \ldots, N \\
&G^f_{zz}\Delta z_{N+1} + \Delta\lambda_{N+1} = 0 \\
&\Delta z_1 = p - p_0
\end{aligned}
\tag{12}
$$

where we adopt the notation: $G^k_{zz} = \nabla_{zz} g_k$, $L^k_{zz} = \nabla_{zz}(g_k + h_k^T\gamma_k)$, $H^l_{zz} = \nabla_{zz}(h_k^T\gamma_k)$, etc.

Non-shifted Variant

The linear system (12) can be written as $K\Delta v = r$ where Δv is the perturbation in the solution of $P(\ell)$, $r^T = [0\ 0\ \ldots (p-p_0)^T]$, and K is given by:

$$
\begin{bmatrix}
 & I & & & & & & & & & \\
I & L^1_{zz} & L^1_{zu} & L^1_{zy} & \nabla_x h_1 & -I & & & & & \\
 & L^1_{uz} & L^1_{uu} & L^1_{uy} & \nabla_u h_1 & & & & & & \\
 & H^1_{yz} & H^1_{yu} & H^1_{yy} & \nabla_y h_1 & -A^T & & & & & \\
 & \nabla_x h_1^T & \nabla_u h_1^T & \nabla_y h_1^T & & & & & & & \\
 & -I & & -A & & & I & & & & \\
 & & & & & I & \cdot & & & & \\
 & & & & & & & \cdot & I & & \\
 & & & & & & & I & \cdot & & \\
 & & & & & & & & & \cdot & I \\
 & & & & & & & & & I & G^f_{zz}
\end{bmatrix}.
\tag{13}
$$

The matrix K is already factored at the solution of $P(\ell)$ found by IPOPT. As a result, by setting $p = \hat{z}(\ell+1)$, the perturbed solution Δv from (12) requires only a single backsolve with K as the on-line optimization cost. The error in this solution is $O(\|\hat{z}(\ell+1) - \hat{z}(\ell)\|^2)$.

Shifted Variant

If N is sufficiently large and the plant has no disturbances or model mismatch, i.e., $\hat{z}(\ell+j) \approx z^*_{j+1}, j \geq 1$, then we can use $\hat{u}(\ell+j) := u^*_{j+1}$ directly from the solution of $P(\ell)$. In addition, if $\|\hat{z}(\ell+j) - z^*_{j+1}\|$ is small, we can determine $\hat{u}(\ell+j)$ from a different perturbed solution of $P(\ell)$. At time $t_{\ell+j}$, we can estimate the solution to $P(\ell+j)$ by adding the following constraints to $P(\ell)$,

$$
\begin{aligned}
\Delta z_1 &= \Delta p \\
\Delta z_{j+1} &= \hat{z}(\ell+j) - z^*_{j+1}
\end{aligned}
\tag{14}
$$

In other words, we force the state variable z_{j+1} to the plant measurement $\hat{z}(\ell+j)$ and we adjust the initial condition parameter Δp to compensate. Adding (14) to (12) leads to the condensed form written as

$$
\left[\begin{array}{c|c} K & E_1 \\ \hline E_j^T & 0 \end{array}\right]
\left[\begin{array}{c} \Delta v \\ \hline \Delta p \end{array}\right] =
\left[\begin{array}{c} 0 \\ \hline \hat{z}(\ell+j) - z^*_{j+1} \end{array}\right]
\tag{15}
$$

where $E_1^T = [-I\ 0 \ldots 0]$, $E_j^T = [0\ \ldots\ I\ 0 \ldots 0]$.

Again we take advantage of a previous factorization of K from $P(\ell)$ and shift to problem $P(\ell+j)$ by solving for Δp, where Δv relate to all of the variables in $P(\ell)$. Using the existing factorization of K, this system could be solved with a Schur complement approach, i.e.,

$$
-(E_j^T K^{-1} E_1)\Delta p = \hat{z}(\ell+j) - z^*_{j+1}
$$

The Schur complement requires n_z background backsolves with K. Although these backsolves are easily parallelized, this calculation may be expensive for systems with many states.

For the implementation of this method, the background and on-line computational tasks are:

- Background:
  - Obtain solution to $P(\ell)$.
  - For $j \leq j_{max}$, construct $E^T_{j+1}K^{-1}E_1$ using n_x backsolves with K and calculate their L/U factors. Set $j = 1$.
- On-line:
  - Having, $\hat{z}(\ell+j)$, evaluate
    $$\Delta p = -(E^T_{j+1}K^{-1}E_1)^{-1}(\hat{z}(\ell+j) - z^*_{j+1})$$
    using precomputed L/U factors of the Schur complement.
  - With Δp that matches z_{j+1} to $\hat{z}(\ell+j)$, calculate (the estimate for) $u(\ell+j)$. This requires a backsolve with K.
- If $j = 1$ start with the updated vector from the on-line calculation to converge the solution for $P(\ell+1)$ in background. If background calculations can be completed before the next sampling time, proceed to the Background step. Else, set $j = j+1$ and proceed to the On-line step.

To demonstrate this real-time iteration strategy, we consider a large-scale case study in the next section and examine the performance of both variants.

5. CASE STUDY: NMPC OF A LOW-DENSITY POLYETHYLENE PLANT

5.1 Description of the Process

The computational framework presented is demonstrated on a simulated NMPC scenario for a large-scale high-pressure low-density polyethylene (LDPE) process presented by (Cervantes *et al.*, 2002). A simplified flowsheet is presented in Figure (1). In this process, ethylene is polymerized in a long tubular reactor at high pressures (2000-3000 atm) and temperatures (150-300 ^{o}C) through a free-radical mechanism. Accordingly a large number of compression stages are required to obtain these extreme operating conditions. The final product is recovered by flash separation. These flexible processes can obtain several different grades by adjusting the reactor operating

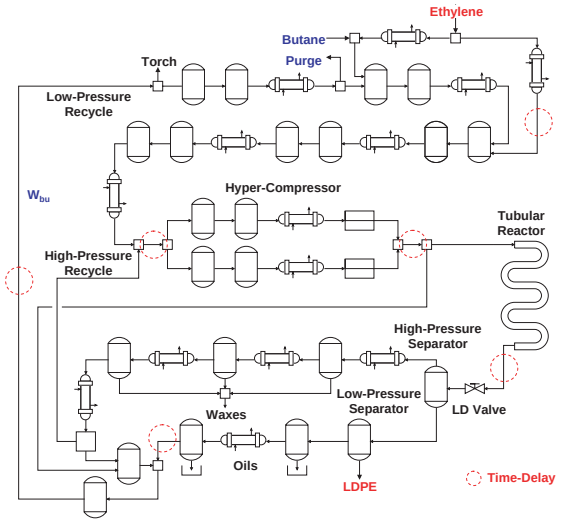

Fig. 1. High-pressure LDPE Process Flowsheet

conditions. Here, the desired final end-use properties such as the polymer melt index are obtained by control of the reactor temperature, pressure and concentration of a chain-transfer agent used to control the polymer molecular weight (usually butane and propylene).

The process presents a difficult dynamic system. The reactor dynamics are much faster as compared to the slow responses in the recycle loops. Furthermore, long time delays are present throughout the compression and separation systems. Due to the exothermic and complex nature of the polymerization, the reactor temperature and pressure are enforced strictly along the operating horizon following complex recipes. Following these lines, the main operational problem in these processes consist in providing fast adjustments to the butane feed and purge stream to keep the melt index at a desired reference value. This is done during different operating stages such as grade transitions (switching between two different operating points) and normal operation (disturbance rejection).

5.2 Dynamic Process Model

Due to the fast dynamics presented in the reactor and to the enforcement of strict operating conditions, an overall dynamic model of the plant can be described by a large number of differential balances around continuous stirred tanks (CST) representing the different compression and separation stages of the process. The dynamic balances considered include four components: ethylene (j=1), butane (j=2), methane (j=3) and impurities (j=4). This last component groups several components present in minor quantities. Details on the derivation of the model can be found were presented by (Cervantes *et al.*, 2002). Non-steady mass balances for three components are developed while the fourth component is obtained by difference. Equation (16) shows the corresponding balance for the j-th component in every piece of equipment in the plant:

$$\frac{d\left(V\rho w^j\right)}{dt} = F\,w_i^j - Fw^j \tag{16}$$

where F, mass flowrate (kg/h); V, equipment volume (m^3); t, time (s); ρ, gas density (kg/m^3); w_i^j, inlet weight component of j-th component; w_j, outlet weight component of j-th component. The gas density is calculated through nonlinear thermodynamic relations. Most of the complexity of the dynamic model is determined by the presence of time delays. For simplicity, these delays are lumped into six overall ones located along the process and are directly incorporated into the model by considering each one as a tube of length L where a plug flow is assumed. The resulting component material balances are given by the following set of partial differential equations:

$$\begin{gathered}\frac{\partial w^j}{\partial t} + \frac{1}{\tau}\frac{\partial w^j}{\partial z} = 0 \\ \left(\frac{\partial w^j}{dz}\right)_L = 0 \qquad w^j(z,0) = w_0^j\end{gathered} \tag{17}$$

where $\tau = \frac{\rho A}{F}$ represents the associated time delay. The PDEs are transformed to ordinary differential equations applying a finite differences scheme with N=10 intervals. The resulting large-scale DAE model contains 289 differential and 64 algebraic state variables. Furthermore, the incorporation of thermodynamic relations increases significantly the complexity of the model.

5.3 Formulation of NMPC Problem

In this work, we are interested on obtaining a optimal feedback policy that minimizes the switching time between to steady states corresponding to the production of polymer grades. Minimization is crucial since it lead to substantial saving in waste product. Since the melt index is inferred from the concentration of butane in the recycle loop, we can use this as the controlled variable and we use the butane feed and purge stream as manipulated variables. The NMPC problem solved on-line at every sampling time t_ℓ is given by,

$$\min \int_{t_\ell}^{t_\ell+t_p} \left(w_{C_4}(t) - w^r_{C_4}\right) + \left(F_{C_4}(t) - F^r_{C_4}\right) + \left(F_{pu}(t) - F^r_{pu}\right)$$

s.t.

$$\begin{aligned} &\text{Equations (16)-(17)} \\ &z(t = t_\ell) = \hat{z}(\ell) \\ &z_L \leq z \leq z_U \\ &y_L \leq y \leq y_U \\ &u_L \leq u \leq u_U \end{aligned} \tag{18}$$

where t_p is the prediction time, w_{C_4} is weight fraction of butane in the recycle loop and subscript r denotes a reference value. Here, we consider equal control and prediction times. Notice that this formulation can easily be reformulated to the general form (1a)-(1g) if we consider a fixed instant k and replace the objective function by introducing an additional state variable.

5.4 Solution of NMPC Problem

Following a simultaneous full-discretization approach, problem (18) is converted into a large-scale NLP problem. A total of 15 finite elements and 3 collocation points are used for the discretization. The finite elements are placed are placed in order to match sampling times along the moving horizon. The resulting NLP problem of form (2) contains 27,135 constraints, 9360 lower bounds, 9360 upper bounds and 30 degrees of freedom. Since grade transitions are usually slow, long prediction times on the order of hours are used with sampling times on the order of minutes. In all our experiments we set t_p= 1.5 min and $t_{\ell+1} - t_\ell$=6 min.

As a first step, we study the off-line solution of problem (18) using IPOPT. In all our numerical experiments a monotone barrier parameter μ update with an initial value of $1\text{x}10^{-6}$ is used, while the rest of the algorithmic parameters were specified with their default values. Computational times associated to the solution of the problem are presented in Table (2). It is clear that the vast majority of the total CPU time is devoted for the factorization of the KKT matrix. A single factorization can take up to 36 seconds. Although it seems feasible to solve full problems within the specified sampling time, a long feedback delay would be introduced, deteriorating the performance of the controller.

In order to minimize the feedback delay, we consider the proposed NLP sensitivity approaches in both its shifted and non-shifted variants and compare them with a conventional NMPC approach. Some general characteristics of the approaches are:

Table 2. Average computational times associated to the off-line solution of problem (18)

Algorithmic Step	**CPU Time (s)**
Full Solution (10 iterations)	351.5
Single Factorization of KKT Matrix	33.9
Step Computation (single backsolve)	0.94
Rest of Steps	0.12

Table 3. On-line and off-line average CPU times for the different NMPC approaches

NMPC Approach	**On-line (s)**	**Off-line (s)**
Full Solution	358.89	—
Non-shifted	0.94	—
Shifted (Total)	1.04	272.5
Schur Complement	—	272.5
Reduced System	0.1	—
Final Backsolve	0.94	—

- *Conventional NMPC Approach*: Solve full problems to optimality. Long feedback delays introduced.
- *NLP Sensitivity Approach - Non-shifted Variant*: Solve a full problem, obtain approximate solutions for N_{cycle} subsequent sampling times by perturbation of the initial conditions while an updated full problem is solved. Upon solution, obtain approximate solutions around this updated full solution. No shifting in the primal and dual variables is performed, leading to inconsistency on the active sets. Consequently, even in the absence of plant-model mismatch, the approximation is not exact for subsequent problems if active set changes are present along the prediction horizon.
- *NLP Sensitivity Approach - Shifted Variant*: Similar to previous approach but a shifting in primal and dual variables performed. The approximate solutions are exact for subsequent problems if there is no plant-model mismatch and sufficiently long prediction horizons. Consistency in the active sets along the prediction horizon is preserved.

In this first study we consider N_{cycle}=1, i.e.; one-to-one cycles of full solution-approximate solution are used to provide feedback. Different combinations of these cycles can be considered, depending on the permitted feedback delay for the particular dynamic system. Although the actual feedback delay introduced by the different approaches affect the actual closed-loop response, they have been ignored. The plant response is obtained by introducing strong and random perturbations to the time delays τ in the recycle loops.

The performance of the NMPC approaches is presented in Figure(2). Notice that the optimal feedback policy involves the saturation of both control valves for the first 2500 seconds of operation and

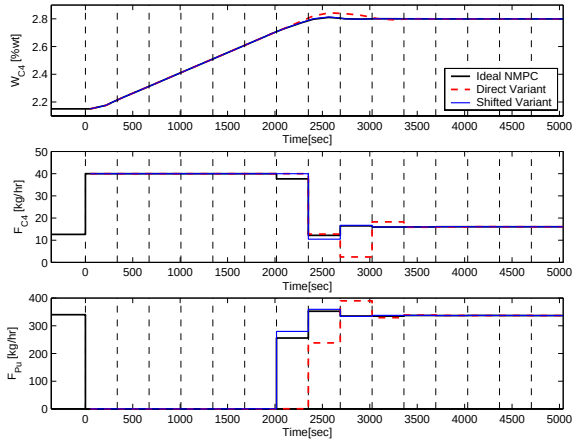

Fig. 2. Closed-loop performance of the different NMPC approaches

finally places the flow rates to values corresponding to the new operating point. It is interesting to observe that the NLP sensitivity shifted variant provides close-to-optimal performance of the controller, as opposed to the non-shifted variant, which encounters problems in the presence of active set changes along the prediction horizon. This is easily observed at the eight sampling time (2500 seconds) where a response delay of one sampling time is obtained. In this case, the non-shifted approach preserves the active set of the purge stream flow rate obtained from the solution of the previous full problem. The resulting suboptimal solution translates into a long overshoot of around 20 minutes leading to the production of large amounts of off-spec product. Although the sampling times in this initial experiment are quite large, the results imply that it is possible to provide close-to-optimal immediate feedback at a high frequency (in the order of seconds) by increasing N_{cycle} and using a finer discretization, while the full problem can still be solved at a lower frequency on the order of minutes.

The on-line and off-line computational times required by the different NMPC approaches are presented in Table (3). On average, the full solution requires around 369 seconds of on-line computation. On the other hand, the non-shifted variant requires a negligible amount of on-line time (around 0.94 seconds for a single backsolve) with no off-line tasks. Similarly, the shifted variant requires a negligible on-line time of around 1.04 seconds for the solution of the dense reduced system and a final backsolve to obtain the updated solution vector. However, it requires a considerable amount of off-line time (272 seconds on average) for the construction of the Schur complement. Although this can be done in the background and only once per cycle, it is still important to reduce this off-line time in order to improve the overall performance of the controller.

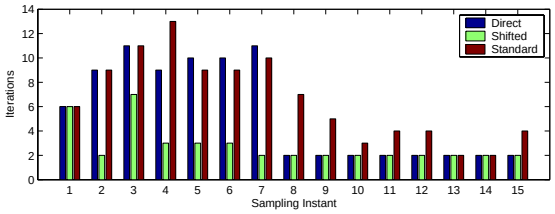

Fig. 3. Number of iterations required for the solution of the full problem using different initialization strategies

Finally, we consider the use of the proposed NLP sensitivity approaches as a base for effective warm-starting strategies. In this case, we consider an NMPC scenario involving the initialization of the full problem at each sampling instant using an approximate solution for the new initial conditions obtained around the solution of the previous problem. In order to compare the suitability of both the shifted and non-shifted variants, we compare the number of iterations taken for the solution of the full problem. The results are presented in Figure (3). Again, it is clear that the shifted variant provides an accurate warm-starts for all the sampling times. On the other hand, since the non-shifted variant provides inconsistent active sets, the optimizer takes a large number of iterations in order to correct this. In some cases it is even more convenient to initialize the problem with the solution of the previous problem (normal initialization). However, notice that the non-shifted approach provides exact approximations at the end of the horizon where no more active set changes are present along the prediction horizons.

6. CONCLUSIONS AND SUMMARY

The shifted variant allows a number of modifications. First, the estimate for $u(\ell + j)$ can be improved by additional iterations of the on-line calculation, by substituting the updated residuals $f(v)$ of the *nonlinear* KKT equations in the right hand side, i.e., $-(E_{j+1}^T K^{-1} E_1)\Delta p = \hat{z}(\ell + j) - z_{j+1}^* - E_{j+1}^T K^{-1} f(v)$. This requires a backsolve with the factored Schur complement and two backsolves with K.

Second, to reduce the number of backsolves in background, an iterative algorithm can be applied to select Δp directly to match the desired value for z_{j+1}, i.e.,

$$r^m = \hat{z}(\ell + j) - z_{j+1}(\Delta p^m)$$
$$\Delta p^{m+1} = \Delta p^m + \varphi(\Delta p^m, r^m)\ m = 1, \ldots$$

This can be done with a number of iterative methods, including quasi-Newton (e.g., Broyden) or a preconditioned Krylov method, such as GMRES, where a previous value of $(E_{j+1}^T K^{-1} E_1)^{-1}$ can be

used as the preconditioner; each iteration requires a backsolve with K.

REFERENCES

Ascher, U. M. and L. R. Petzold (1998). *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM. Philadelphia, PA.

Barton, P. I., R. J. Allgor, W. F. Feehery and S. Galan (1998). Dynamic optimization in a discontinuous world. *Industrial & Engineering Chemistry Research* **37**, 966–981.

Bartusiak, R. D. (2006). Nlmpc: A platform for optimal control of feed- or product-flexible manufacturing. In: *Assessment and Future Directions of NMPC*. p. to appear. Springer. Berlin.

Betts, J. (2001). *Practical Methods for Optimal Control Using Nonlinear Programming*. Philadelphia, PA, USA.

Betts, J. T. and W. P. Huffman (1992). Application of sparse nonlinear programming to trajectory optimization. *J. Guidance Dyn. Cont.*

Biegler, L.T., A.M. Cervantes and A. Wächter (2002). Advances in simultaneous strategies for dynamic process optimization. *Chem.Eng.Sci.* **57**, 575–593.

Bock, H. G. (1983). Numerical treatment of inverse problem in differential and integral equations. In: *Recent advances in parameter identification techniques for o.d.e.* Heidelberg, Federal Republic of Germany.

Bock, H. G. and K.J. Plitt (1984). A multiple shooting algorithm for direct solution of optimal control problems. *9th IFAC World Congress, Budapest*.

Bock, H. G., M. Diehl, E. Kostina and J. P. Schlöder (2006). Constrained optimal feedback control of systems governed by large differential algebraic equations. *Real-Time PDE-Constrained Optimization, SIAM, Philadelphia*.

Bryson, A. E. and Y. C. Ho (1975). *Applied Optimal Control*. Hemisphere.

Camacho, E.F. and C. Bordons (2006). Nonlinear model predictive conrol: an introductory survey. In: *Assessment and Future Directions of NMPC*. p. to appear. Springer. Berlin.

Cervantes, A. M. and L. T. Biegler (2000). Optimization strategies for dynamic systems. In: *Encyclopedia of Optimization* (C. Floudas and P. Pardalos, Eds.).

Cervantes, A. M., S. Tonelli, A. Brandolin, J. A. Bandoni and L. T. Biegler (2002). Large-scale dynamic optimization for grade transitions in a low density polyethylene plant. *Computers & Chemical Engineering* **26**, 227.

Cuthrell, J.E. and L.T. Biegler (1989). Simultaneous optimization and solution methods for batch reactor control profiles. *Comput. Chem. Eng.* **13**, 49–62.

Diehl, M., H.G. Bock and J.P. Schlöder (2005*a*). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization* **43, 5**, 1714–1736.

Diehl, M., R. Findeisen and F. Allgöwer (2006). A stabilizing real-time implementation of nonlinear model predictive control. *Real-Time PDE-Constrained Optimization, SIAM, Philadelphia*.

Diehl, M., R. Findeisen, H.G. Bock, J.P. Schlöder and F. Allgöwer (2005*b*). Nominal stability of the real-time iteration scheme for nonlinear model predictive control. *IEE Control Theory Appl.* **152, 3**, 296–308.

Fiacco, A. V. (1983). *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Academic Press. New York.

Findeisen, R. and F. Allgöwer (2004). Computational delay in nonlinear model predictive control. *Proc. Int. Symp. Adv. Control of Chemical Processes, ADCHEM'03*.

Flores-Tlacuahuac, A., L. T. Biegler and E. Saldivar-Guerra (2005). Dynamic optimization of hips open-loop unstable polymerization reactors. *I & EC Research* **44, 8**, 2659–2674.

Forsgren, A., P.E. Gill and M.H. Wright (2002). Interior methods for nonlinear optimization. *SIAM Review* **44/4**, 525–597.

Fourer, R., D. Gay and B. Kernighan (1993). *AMPL*. The Scientific Press. South San Francisco.

Franke, R. and J. Doppelhamer (2006). Integration of advanced model based control with industrial it. In: *Assessment and Future Directions of NMPC*. p. to appear. Springer. Berlin.

Groetschel, M., S. Krumke and J. Rambau (eds.) (2001). *Online Optimization of Large Systems*. Springer. Berlin.

Hager, W.W. (2000). Runge-kutta methods in optimal control and the transformed adjoint system. *Numer. Math.* **87**, 247–282.

Jockenhövel, T., L.T. Biegler and A. Wächter (2003). Dynamic optimization of the tennessee eastman process using the optcontrolcentre. *Computers and Chemical Engineering* **27,11**, 1513–1531.

Kameswaram, S. and L. T. Biegler (2005). Convergence rates for direct transcription of optimal control problems using collocation at radau points. *submitted for publication*.

Leineweber, D. B. (1999). Efficient reduced sqp methods for the optimization of chemical processes described by large sparse dae models. *University of Heidelberg, Heidelberg, Germany*.

Magni, L. and R. Scattolini (2006). Robustness and robust design of mpc for nonlinear systems. In: *Assessment and Future Directions of NMPC*. p. to appear. Springer. Berlin.

Nagy, Z. K., R. Franke, B. Mahn and F. Allgöwer (2006). Real-time implementation of nonlinear model predictive control of batch processes in an industrial framework. In: *Assessment and Future Directions of NMPC*. p. to appear. Springer. Berlin.

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

Reddien, G.W. (1979). Collocation at gauss points as a discretization in optimal control. *SIAM J. Control Optim* **17**, 298–306.

Santos, L. O., P. Afonso, J. Castro, N. Oliveira and L. T. Biegler (2001). On-line implementation of nonlinear mpc: An experimental case study. *Control Engineering Practice* **9**, 847.

Toumi, A., M. Diehl, S. Engell, H. G. Bock and J. P. Schlöder (2005). Finite horizon optimizing control of advanced smb chromatographic processes. *16th IFAC World Congress, Prague*.

Vassiliadis, V. S., R. W. H. Sargent and C. C. Pantelides (1994*a*). Solution of a class of multistage dynamic optimization problems. part i - algorithmic framework. *Ind. Eng. Chem. Res.* **33**, 2115–2123.

Vassiliadis, V. S., R. W. H. Sargent and C. C. Pantelides (1994*b*). Solution of a class of multistage dynamic optimization problems. part ii - problems with path constraints. *Ind. Eng. Chem. Res.* **33**, 2123–2133.

Wächter, A. and L.T. Biegler (2006). On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Program.* **106 (1)**, 25–57.