INTERNATIONAL JOURNAL OF ROBUST AND NONLINEAR CONTROL
Int. J. Robust Nonlinear Control 2008; **18**:800–815
Published online 9 July 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/rnc.1250

Fast implementations 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 15213, U.S.A.

SUMMARY

In less than two decades, nonlinear model predictive control has evolved from a conceptual framework to an attractive, general approach for the control of constrained nonlinear processes. These advances were realized both through better understanding of stability 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 online application and discuss further advances to deal with applications of increasing size and complexity. To address these challenges, we adapt the real-time iteration concept, developed in the context of multiple shooting (*Real-Time PDE-Constrained Optimization*. SIAM: Philadelphia, PA, 2007; 25–52, 3–24), to a collocation-based approach with a full-space nonlinear programming solver. We show that straightforward sensitivity calculations from the Karush–Kuhn–Tucker system also lead to a real-time iteration strategy, with both direct and shifted variants. This approach is demonstrated on a large-scale polymer process, where online calculation effort is reduced by over two orders of magnitude. Copyright © 2007 John Wiley & Sons, Ltd.

Received 11 November 2006; Revised 26 March 2007; Accepted 29 May 2007

KEY WORDS: 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 method for process control of large industrial 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 online, time critical solutions. Recent developments in large-scale nonlinear programming (NLP) algorithms have enabled the online solution of these problems. In particular, in a recent

*Correspondence to: Lorenz T. Biegler, Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.
†E-mail: lb01@andrew.cmu.edu

IFAC workshop several industrial applications were presented including contributions from Exxon Mobil [1], BASF [2] and ABB [3]. In addition to enabling NLP solvers, there is also a much better understanding of NMPC stability and robustness properties and associated dynamic optimization formulations that provide them [4].

Moreover, with the ability to solve dynamic optimization problems online, the separation between model predictive control and real-time optimization begins to disappear. A comprehensive treatment of dynamic real-time optimization is provided in [5], and it is clear that with improved optimization formulations and algorithms, the role of NMPC can be greatly expanded. On the other hand, time-critical solutions demand better enabling algorithms and their implementations. Also, the need to consider complex 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 *computational delay*. A frequent assumption is that online 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 in an NMPC implementation of a laboratory reactor [6] as well as in numerous industrial studies. Deterioration of stability was noted in [7], where a detailed stability analysis is provided. To address this issue, Diehl *et al*. [8] and Bock *et al*. [9] developed a real-time iteration in the context of multiple shooting and Successive Quadratic Programming. A related sensitivity-based approach was also developed for sequential dynamic optimization in [10]. This study extends the previous work to 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 discusses the NLP solver, IPOPT [11], as well as its adaptation to NLP sensitivity. These sensitivity calculations provide the basis for a fast NMPC strategy, 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 direct and 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:

$$
\begin{aligned}
&\min \sum_{k=1}^{N} \varphi(z(t_k), u_k) \qquad (1)\\
&\text{s.t.} \frac{\mathrm{d}z_k(t)}{\mathrm{d}t} = f(z_k(t), y_k(t), u_k), \quad t \in [t_{k-1}, t_k]\\
&\quad g(z_k(t), y_k(t), u_k) = 0\\
&\quad z_k(t_k) = z_{k+1}(t_k), \quad z_1(t_0) = z_0\\
&\quad u_k^{\mathrm{L}} \leqslant u_k(t) \leqslant u_k^{\mathrm{U}}, \quad y_k^{\mathrm{L}} \leqslant y_k(t) \leqslant y_k^{\mathrm{U}}\\
&\quad z_k^{\mathrm{L}} \leqslant z_k(t) \leqslant z_k^{\mathrm{U}}, \quad k = 1, \ldots, N
\end{aligned}
$$

DOI: 10.1002/rnc

where $z_k(t) \in \Re^{n_z}$ 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 equation (DAE) model which we assume without loss of generality as index one.

A number of approaches can be taken to solve (1). 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 [12, 13]. For problems without inequality constraints, these conditions can be formulated and solved as a two-point boundary value problem. 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 [14], and optimization is performed with respect to these controls. Given initial conditions and a set of control parameters, the DAE model is solved for $k = 1, \ldots, N$ 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. Sequential strategies are relatively easy to construct and to apply as they incorporate reliable DAE solvers (e.g. DASSL, DASOLV, DAEPACK) as well as NLP solvers (NPSOL, SNOPT) as components. 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 [15]. 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 (1), the time domain is partitioned into N smaller time elements and the DAE models are integrated separately in each element [9]. Control variables are parameterized as in the sequential approach and gradient information is obtained for both control variables as well as initial conditions of the states variables in each element. Finally, equality constraints are added in the NLP to link the elements with state profiles that 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 collocation approach, also known as *direct transcription*, we discretize both the state and control profiles in time using collocation on the finite elements $k = 1, \ldots, N$. This approach corresponds to a particular implicit Runge–Kutta method with high-order accuracy and superior 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, this approach also leads to large-scale NLP problems that require efficient optimization strategies [16, 17]. Because these methods directly couple the solution of the DAE system with

DOI: 10.1002/rnc

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 resulting approach, the control variables are discretized at the same level as the state variables and, under mild conditions, (see [18, 19]) the Karush–Kuhn–Tucker (KKT) conditions of the simultaneous NLP are consistent with the optimality conditions of the discretized variational problem, and convergence properties 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 full-space methods are often preferred for their efficient solution. Second derivatives of the objective and constraint functions along with measures to deal with directions of negative curvature in the Hessian matrix [11] are essential for the superior performance of this method. A detailed description of the simultaneous approach is provided in [20] for full-space methods, along with mesh refinement strategies and case studies in mechanics and aerospace.

2.1. Comparison of dynamic optimization strategies

Table I lists the complexity of the major algorithmic steps for dynamic optimization of (1) using the sequential, multiple shooting and simultaneous collocation 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. Step (i) requires sequential and multiple shooting methods to 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 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 than linearly with problem size. For the simultaneous approach, this step is replaced in the optimization steps, (v) and (vi). In Step (ii) 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 linearly 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.

Table I. Computational complexity/NLP iteration (with $n_w = n_z + n_y$ state variables, n_u manipulated variables, N time steps, $2 \leqslant \alpha \leqslant 3$, $1 \leqslant \beta \leqslant 2$).

	Sequential	Multiple shooting	Simultaneous collocation
i. Integration	$n_w^\beta N$	$n_w^\beta N$	—
ii. Sensitivity	$n_w n_u N^2$	$n_w(n_w + n_u)N$	$(n_w + n_u)N$
iii. Hessian evaluation	$n_w n_u^2 N^3$	$n_w(n_w + n_u)^2 N$	$(n_w + n_u)N$
iv. Decomposition	—	$n_z^3 N$	—
v. Factorization	$(n_u N)^\alpha$	$(n_u N)^\alpha$	$((n_w + n_u)N)^\beta$
vi. Backsolve	—	—	$(n_w + n_u)N$

DOI: 10.1002/rnc

Step (iii) is optional as NMPC often uses a Gauss–Newton approximation for second derivatives. This approximation is a reasonable choice for quadratic objectives and requires minimal additional computational cost. However, for economic objectives, exact second derivatives are preferred and extend from the sensitivity calculation. For both multiple shooting and sequential approaches, the cost of reduced Hessian's scales roughly with the number of input times the *sensitivity* cost. In addition, multiple shooting executes a *decomposition* (Step (iv)) that requires projection of the Hessian to dimension $n_u N$, through the factorization of *dense* matrices. With the collocation approach, the Hessian remains very sparse and its calculation (through automatic differentiation) scales with the problem size. Steps (v) and (vi) deal with the optimization step determination; sequential and 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*. On the other hand, simultaneous collocation 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).

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

2.2. Online vs background calculations

The *real-time iteration* strategy was developed in [21] 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 [21, 22] is general and applies to simultaneous and (open-loop stable) sequential approaches as well. We adapt this approach here by noting that when 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 the current state is obtained, a perturbed problem is solved efficiently to update the NLP solution. The perturbed problem can be formulated in two ways. To solve (1) for the ℓth sampling time, the initial condition is $z_0(t_0) = \hat{z}(\ell)$, the measured state. In the *direct* or nonshift strategy, an N-element problem is derived from the solution of this problem but with the initial condition, $z_0(t_0)$ 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 problem with $N - 1$ elements. Here, the initial condition is $z_1(t_1) = \hat{z}(\ell + 1)$ 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 Nth element. In both the cases, the only online cost is Step (v) in Table I.

3. NLP ALGORITHM AND SENSITIVITY

To develop and extend the real-time iteration framework, we first consider methods for the solution of the NLP resulting from the simultaneous collocation formulation. The discretized

DOI: 10.1002/rnc

problem derived from (1) can be rewritten for illustration in the following form:

$$\min f(x,p) \quad \text{s.t. } c(x,p)=0, \;\; x \geqslant 0 \tag{2}$$

with the parameter vector p. As this is a large-scale NLP with potentially many degrees of freedom and inequalities, we apply the IPOPT algorithm [11] for its efficient solution. The algorithm follows a barrier approach, where the nonnegativity constraint is replaced by logarithmic barrier terms and added to the objective function to give:

$$\min \varphi(x,p) = f(x,p) - \mu \sum_{i=1}^{n} \ln(x^{(i)}) \quad \text{s.t. } c(x,p)=0 \tag{3}$$

with a barrier parameter $\mu > 0$. Here, $x^{(i)}$ denotes the ith component of the vector x. Since the objective function of this barrier problem becomes arbitrarily large as $x^{(i)}$ approaches zero, a local solution $x(\mu)$ of this problem lies in the positive orthant, $x(\mu) > 0$. Under mild conditions, $x(\mu)$ converges to a local solution of the original problem (2) as $\mu \to 0$; 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. The IPOPT code follows this 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:

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

where we define $e = [1, 1, \ldots, 1]^{\mathrm{T}}$, $X = \mathrm{diag}(x)$, $V = \mathrm{diag}(v)$, the Hessian $W_j = \nabla_{xx} L(x_j, \lambda_j, v_j)$, the Lagrange function $L(x, \lambda, v) = f(x) + c(x)^{\mathrm{T}} \lambda - x^{\mathrm{T}} v$ and $A_j = \nabla c(x_j)$. IPOPT solves this system by first solving a 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 [23]. 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 [11].

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 with respect to x. Let $x(p_0)$ be a KKT point, where the linear independence constraint qualification (LICQ) holds, strict complementarity holds with the bound multipliers $v(p_0)$ satisfying the KKT conditions, and there exists $\omega > 0$ such that $q^{\mathrm{T}} W(x(p_0), \lambda(p_0)) q \geqslant \omega ||q||^2$ for equality constraint

DOI: 10.1002/rnc

multipliers $\lambda(p_0)$ satisfying the KKT conditions and all nonzero $q \in \Re^{n_x}$ in the nullspace of the active constraint normals.

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 $x(p_0)$. Also, 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 neighbourhood of $\mu = 0$ and $||x(\mu_\ell) - x(p_0)|| = O(\mu_\ell)$ with $\lim_{\mu \to 0^+} x(\mu) = x(p_0)$.

Proof
The proof follows by noting that LICQ implies MFCQ and invoking Theorem 3.12 and Lemma 3.13 in [24]. □

This property indicates that nearby solutions of (3) provide useful information for bounding properties for (3) 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 $\mathrm{d}x(p_0)/\mathrm{d}p$ and a perturbed solution $\Delta x = (\partial x(p_0)/\partial p)^{\mathrm{T}}(p - p_0)$.

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: $x(p_0)$ is an isolated minimizer and the associated multipliers $\lambda(p_0)$ and $v(p_0)$ are unique. Also, for some p in a neighbourhood of p_0, there exists an m times differentiable function $s(p)^{\mathrm{T}} = [x(p)^{\mathrm{T}}\ \lambda(p)^{\mathrm{T}}\ v(p)^{\mathrm{T}}]$ that corresponds to a locally unique minimum for (3), and for p near p_0 the set of active inequalities is unchanged and complementary slackness holds.

Proof
The results follow directly from Theorem 3.2.2 and Corollary 3.2.5 in [25]. □

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 $v(\mu,p_0)$ are unique, and for some p in a neighbourhood of p_0 there exists an m times differentiable function $s(\mu,p)^{\mathrm{T}} = [x(\mu,p)^{\mathrm{T}}\ \lambda(\mu,p)^{\mathrm{T}}\ v(\mu,p)^{\mathrm{T}}]$ that corresponds to a locally unique minimum for (3). Finally, $\lim_{\mu \to 0, p \to p_0} s(\mu,p) = s(p_0)$.

Proof
The results follow from Theorem 6.2.1 and Corollary 6.2.2 in [25]. These were originally proved for a mixed penalty function, but the proofs are easily modified to deal with barrier functions. □

DOI: 10.1002/rnc

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

$$M(s(\mu,p_0)) = \begin{bmatrix} W(s(\mu,p_0)) & A(x(\mu,p_0)) & -I \\ A(x(\mu,p_0))^{\mathrm{T}} & 0 & 0 \\ V(p_0) & 0 & X(p_0) \end{bmatrix}, \quad N_p(s(\mu,p_0)) = \begin{bmatrix} \nabla_{x,p}L(s(\mu,p_0))^{\mathrm{T}} \\ \nabla_p c(x(\mu,p_0))^{\mathrm{T}} \\ 0 \end{bmatrix}$$

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

$$\frac{\mathrm{d}s(\mu,p_0)^{\mathrm{T}}}{\mathrm{d}p} = -M(s(\mu,p_0))^{-1}N_p(s(\mu,p_0)) \tag{5}$$

For small values of μ and $||p-p_0||$, it can be shown from the above properties [25] that

$$s(\mu,p) = -M(s(\mu,p_0))^{-1}N_p(s(\mu,p_0))(p-p_0) + O||p-p_0||^2 \tag{6}$$

The sensitivity calculation in (6) is inexpensive and requires only a single factorization of $M(s(\mu,p_0))$ as well as a backsolve for each right-hand side. Furthermore, the implementation of this calculation is straightforward in the current IPOPT framework. The IPOPT algorithm requires the solution of (4) at each iteration and $M(s(\mu,p_0))$ is directly related to the matrix in (4) at the solution. The current version of IPOPT has been adapted for efficient sensitivity calculations. Specifically, by using a previously factorized form of the matrix in (4), the online cost of finding the approximate solution of a perturbed problem is only a single backsolve.

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 [9], we classify the NMPC calculation into *off-line*, *background* and *online* 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, it is aided by off-line simulations and is not difficult. In addition, the barrier parameter in IPOPT can be tuned to improve performance [17]. 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 (N_{cycle}), but that computational feedback delay needs to be minimal. As a result, we perform items (ii)–(v) in Table I in background and fully converge the solution with the simultaneous approach. As seen in the previous section, the simultaneous approach is generally faster than competing approaches. Here, the dominant calculation is the repeated factorization of the large sparse matrix in (4). Using the fully converged NLP from the background calculation, up to N_{cycle} ($\geqslant 1$) steps behind the current sampling time, the *online* component uses direct and shifted variants of NLP sensitivity to calculate an estimated solution with perturbed initial conditions. As in [9, 22], the cost of the online component is two orders of magnitude less than the background component. Once a measurement is obtained, the online update of the solution vector requires a single

DOI: 10.1002/rnc

backsolve (vi). As seen in (6), the solution error in the perturbed update is also small; it scales quadratically with the perturbation.

The NLP for the ℓth horizon of an NMPC problem is represented by

$$\begin{aligned} P(\ell)\ \min\ & g_f(z_N) + \sum_{k=0}^{N-1} g_k(z_k, u_k) \\ \text{s.t. } & z_{k+1} = z_k + \mathbf{B}y_k \\ & h(z_k, y_k, u_k) = 0, \quad k = 0, \ldots, N-1 \\ & z_0 = \hat{z}(\ell) = p_0 \end{aligned}$$

where $y_k \in \Re^{n_y}$ is 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. Any inequalities from (1) can be replaced by appropriate barrier terms as in (4). Also, $\hat{z}(\ell)$ is the measured or estimated state of the plant at time t_ℓ. At the solution of problem $P(\ell)$, we inject the control $\hat{u}(\ell) \coloneqq u_0$ into the plant.

Having the solution of $P(\ell)$ we would like to obtain a cheap estimate of the solution to $P(\ell+1)$. The optimality conditions for $P(\ell)$ are

$$\begin{gathered} z_0 = p_0 \\ \left.\begin{aligned} \nabla_z g_k - \lambda_{k+1} + \lambda_k + \nabla_z h_k \gamma_k &= 0 \\ \nabla_y h_k \gamma_k - \mathbf{B}^{\mathrm{T}} \lambda_{k+1} &= 0 \\ \nabla_u g_k + \nabla_u h_k \gamma_k &= 0 \\ z_{k+1} = z_k + \mathbf{B}y_k & \\ h(z_k, y_k, u_k) = 0 & \end{aligned}\right\} k = 0, \ldots, N-1 \\ \nabla_z g_f + \lambda_N = 0 \end{gathered}$$

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

$$\begin{aligned} 0 &= \Delta z_0 - (p - p_0) \\ 0 &= L^k_{zz}\Delta z_k + L^k_{zu}\Delta u_k + H^k_{zy}\Delta y_k + \nabla_z h_k \Delta\gamma_k - \Delta\lambda_{k+1} + \Delta\lambda_k \\ 0 &= H^k_{yz}\Delta z_k + H^k_{yu}\Delta u_k + H^k_{yy}\Delta y_k - \mathbf{B}^{\mathrm{T}}\Delta\lambda_{k+1} + \nabla_y h_k \Delta\gamma_k \\ 0 &= L^k_{uz}\Delta z_k + L^k_{uu}\Delta u_k + H^k_{uy}\Delta y_k + \nabla_u h_k \Delta\gamma_k \\ 0 &= \Delta z_{k+1} - \Delta z_k - \mathbf{B}\Delta y_k \\ 0 &= \nabla_z h_k^{\mathrm{T}}\Delta z_k + \nabla_u h_k^{\mathrm{T}}\Delta u_k + \nabla_y h_k^{\mathrm{T}}\Delta y_k \\ 0 &= G^f_{zz}\Delta z_N + \Delta\lambda_N \end{aligned} \tag{7}$$

DOI: 10.1002/rnc

with $k = 0, \ldots, N-1$. Here, we adopt the notation: $G_{zz}^k = \nabla_{zz} g_k$, $L_{zz}^k = \nabla_{zz}(g_k + h_k^{\mathrm{T}} \gamma_k)$, $H_{zz}^k = \nabla_{zz}(h_k^{\mathrm{T}} \gamma_k)$, etc.

Direct variant: The linear system (7) can be written as $K\Delta v = r$ where Δv is the perturbation in the solution of $P(\ell)$, $r^{\mathrm{T}} = [(p - p_0)^{\mathrm{T}}\ 0\ \ldots\ 0]$, and K has a block tridiagonal structure. Let $N_{\mathrm{cycle}}\Delta t$ be an upper bound on the CPU time needed to solve $P(\ell)$. Then sensitivity-based estimates are required for $P(\ell + j)$, $j = 1, \ldots, N_{\mathrm{cycle}}$. The matrix K is already factored in background at the solution of $P(\ell)$ found by IPOPT, and by setting $p = \hat{z}(\ell + j)$, the perturbed solution Δv from (7) requires only a single backsolve with K as the online optimization cost. The error in this solution is $O(||\hat{z}(\ell + j) - \hat{z}(\ell)||^2)$. However, if this perturbation is large, the solution with previous active set may not apply and controller performance will deteriorate by assuming an incorrect active set. This can be observed in the next section.

Shifted variant: If N is sufficiently large and the plant has no disturbances or model mismatch, i.e. $\hat{z}(\ell + j) \approx z_j^*, j \geqslant 1$, and we can use $\hat{u}(\ell + j) \coloneqq u_j^*$ directly from the solution of $P(\ell)$. On the other hand, for small $||\hat{z}(\ell + j) - z_j^*||$, 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 $\Delta z_j = \hat{z}(\ell + j) - z_j^*$ to (7), and making p a variable in (7). In other words, we force the state variable z_j to the plant measurement $\hat{z}(\ell + j)$ and we adjust the initial condition parameter $\Delta p = p - p_0$ to compensate. This corresponds to the sensitivity equations for $P(\ell + j)$ (starting from z_j) and leads to the condensed form written as

$$\left[\begin{array}{c|c} K & E_0 \\ \hline E_j^{\mathrm{T}} & 0 \end{array}\right] \left[\begin{array}{c} \Delta v \\ \hline \Delta p \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \hat{z}(\ell + j) - z_j^* \end{array}\right] \tag{8}$$

where $E_0^{\mathrm{T}} = [-I\ 0 \ldots 0]$, $E_j^{\mathrm{T}} = [0\ \ldots\ I\ 0 \ldots 0]$. 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 relates 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.

$$-(E_j^{\mathrm{T}} K^{-1} E_0)\Delta p = \hat{z}(\ell + j) - z_j^* \tag{9}$$

The Schur complement requires n_z backsolves with K in background. Although these backsolves are easily parallelized, this calculation may be expensive for systems with many states. For the implementation of this method, the main computational tasks are to solve $P(\ell)$, factorize K and construct and factorize $E_j^{\mathrm{T}} K^{-1} E_0$ for $j \leqslant N_{\mathrm{cycle}}$, using n_z backsolves with K in background. Once $\hat{z}(\ell + j)$ is obtained, then Δp can be obtained online from (9) and $u(\ell + j)$ is then calculated from $K\Delta v = -E_0 \Delta p$ using a single backsolve with factors of K. Once $j = N_{\mathrm{cycle}}$ the process begins again with a new background solution of $P(\ell)$.

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

To demonstrate this real-time iteration strategy, we examine the performance of both variants on a large-scale case study. Here, we consider a simulated NMPC scenario for a large-scale high-pressure low-density polyethylene (LDPE) process presented in [26]. 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°C) through a free-radical mechanism. Accordingly, a large number of compression stages are required to obtain these extreme

DOI: 10.1002/rnc

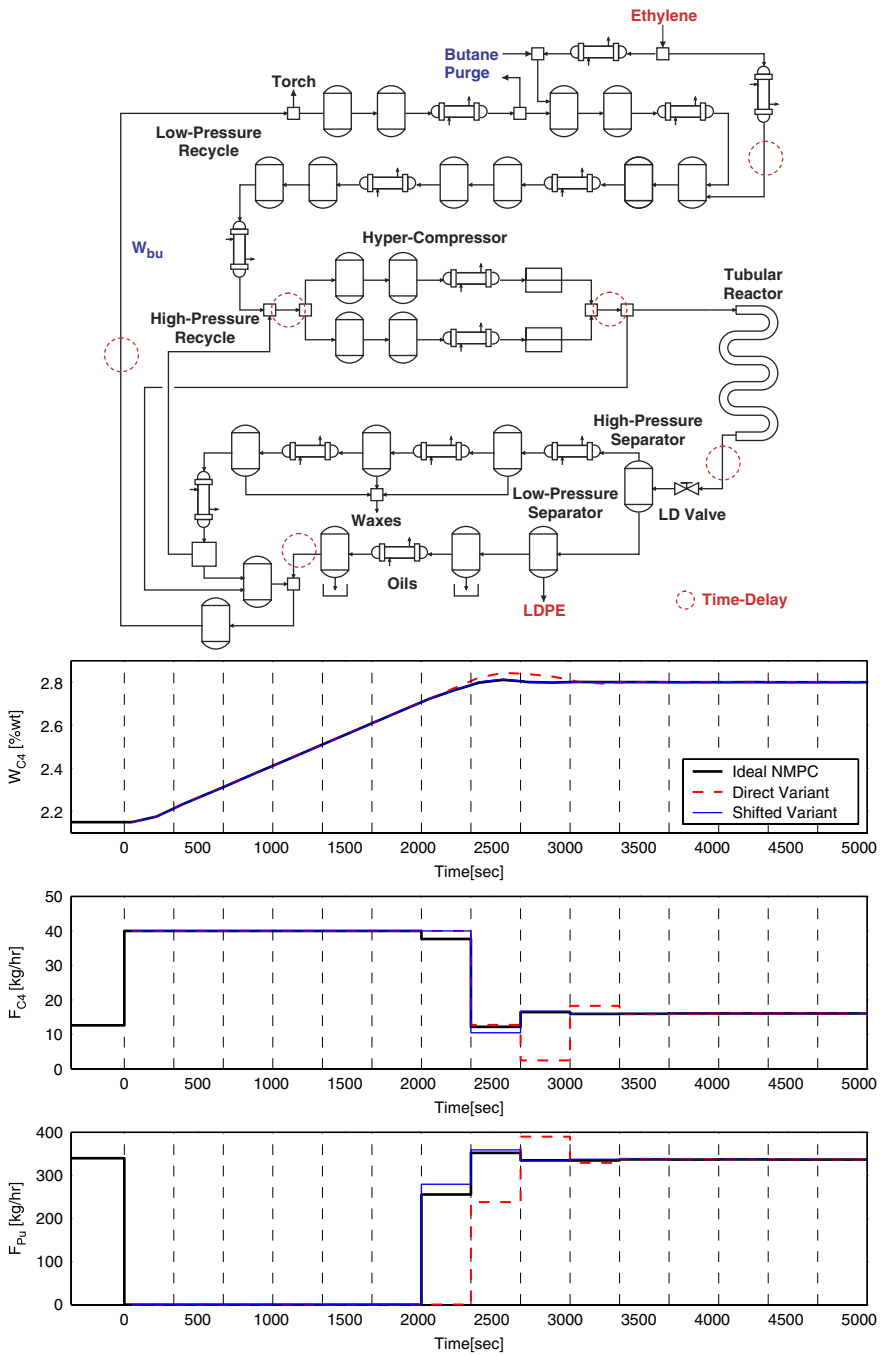

Figure 1. Closed-loop response of different NMPC variants in LDPE process.

DOI: 10.1002/rnc

operating conditions. The final product is recovered by flash separation. These flexible processes can obtain several different grades by adjusting the reactor operating 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 (usually butane and propylene), which is used to control the polymer molecular weight.

The process represents a difficult dynamic system. The reactor dynamics are much faster 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. The main operational problem in these processes consists of providing fast adjustments to the butane feed and purge stream to keep the melt index at a desired reference value. This is performed during different operating stages such as grade transitions (switching between two different operating points) and normal operation (disturbance rejection).

5.1. Dynamic process model

Due to fast dynamics in the reactor and the enforcement of strict operating conditions, the overall dynamic model of the plant can be described by a large number of differential balances around continuous stirred tanks 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. Nonsteady mass balances for three components are developed while the fourth component is obtained by difference. Equation (10) shows the balance for the jth component for every plant unit:

$$\frac{\mathrm{d}(V\rho w^j)}{\mathrm{d}t} = Fw^j_{\mathrm{in}} - Fw^j \tag{10}$$

where F is the mass flow rate (kg/h), V the equipment volume (m^3), t the time (s), ρ the gas density (kg/m^3), w^j_{in} the inlet weight component of jth component and w_j the outlet weight component of jth component. The gas density is calculated through nonlinear thermodynamic relations, which significantly increase the difficulty of the model. Most of the complexity of the dynamic model is caused by the presence of time delays. For simplicity, these delays are lumped into six overall locations along the process and are directly incorporated into the model by considering each one as a tube of length L where plug flow is assumed. The resulting component material balances are given by the following partial differential equations with boundary conditions:

$$\frac{\partial w^j}{\partial t} + \frac{1}{\tau}\frac{\partial w^j}{\partial z} = 0, \quad w^j(z,0) = w^j_0, \quad w^j(0,t) = w^j_{\mathrm{in}}(t) \tag{11}$$

where $\tau = \rho A/F$ represents the associated time delay. The PDEs are transformed to ordinary differential equations by applying a spatial finite difference scheme with $N = 10$ intervals. The resulting large-scale DAE model contains 294 differential and 64 algebraic state variables.

5.2. Formulation of NMPC problem

In this work, we are interested in obtaining an optimal feedback policy that minimizes the switching time between steady states corresponding to the production of different polymer

DOI: 10.1002/rnc

grades. This minimization is crucial as it leads to substantial reduction of waste product. Since the melt index is inferred from the butane concentration in the recycle loop, we use this as the controlled variable and we use the butane feed and purge stream as manipulated variables. The NMPC problem solved online at every sampling time t_ℓ is given by

$$\min \int_{t_\ell}^{t_\ell+t_\mathrm{p}} (w_{C_4}(t) - w_{C_4}^\mathrm{r})^2 + (F_{C_4}(t) - F_{C_4}^\mathrm{r})^2 + (F_{pu}(t) - F_{pu}^\mathrm{r})^2 \,\mathrm{d}t$$

$$\text{s.t. Equations (10)}-\text{(11)}$$

$$z(t_\ell) = \hat{z}(\ell), \quad z_\mathrm{L} \leqslant z \leqslant z_\mathrm{U}, \quad y_\mathrm{L} \leqslant y \leqslant y_\mathrm{U}, \quad u_\mathrm{L} \leqslant u \leqslant u_\mathrm{U} \tag{12}$$

where t_p is the prediction time, w_{C_4} is the weight fraction of butane in the recycle loop and superscript r denotes a reference value. Here, we consider equal control and prediction horizons. Note that this formulation corresponds to the general form (1) if we consider time intervals k and replace the objective function by introducing an additional state variable and equation.

5.3. Solution of NMPC problem

Following a simultaneous full-discretization approach, problem (12) is converted into a large-scale NLP problem. A total of 15 finite elements with three collocation points in each element is sufficient for the time discretization. The finite elements 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 $t_\mathrm{p} = 1.5$ h and $\Delta t = t_{\ell+1} - t_\ell =$ 6 min.

As a first step, we study the off-line solution of problem (12) using IPOPT. Here, we use a monotone barrier parameter μ update with an initial value of 10^{-6}, while the rest of the algorithmic parameters were specified with their default values. Using an average of 10 iterations, the full solution requires 351.5 CPU seconds (3.0 GHz Pentium IV processor, 1 Gb RAM), while a single factorization requires 33.9 CPU seconds, a single backsolve requires 0.94 CPU seconds and the remaining steps take only 0.12 CPU seconds. Note that the dominant cost of the total CPU time is the factorization of the KKT matrix. Although it seems feasible to solve full problems within the specified sampling time, a long computational delay would be introduced, thus degrading the performance of the controller.

In order to minimize the computational delay, we consider the direct and shifted NLP sensitivity variants, and compare them with a ideal NMPC approach. For this case study, we set $N_\mathrm{cycle} = 1$ and we introduced model mismatch through strong and random perturbations of the time delays τ in the recycle loops. Along with the two variants, we consider the exact solution of $P(\ell)$ by neglecting the effect of computational delay on the actual closed-loop response. This leads to an *ideal* NMPC performance profile, which cannot be achieved in practice. Performance of the NMPC approaches is presented in Figure 1. Notice that the ideal policy involves the saturation of both control valves for the first 2500 s of operation and finally places the flow rates to values corresponding to the new operating point. It is interesting to observe that the shifted variant provides close-to-optimal performance of the controller, while the direct variant encounters problems with active set changes along the prediction horizon. This is easily

DOI: 10.1002/rnc

observed at the eighth sampling time (at $t = 2500$ s) where a response delay of one sampling time is obtained. In this case, the direct approach retains the previous active set of the purge stream flow rate, which is one step behind the exact solution. This translates to a large overshoot for around 10 min of operation, leading to the production of large amounts of off-spec product. Also, note that it is possible to provide close-to-optimal immediate feedback more often (every few seconds) by increasing N_{cycle} and using a finer discretization, even though the full NLP problem is solved in background every few minutes.

To compare the online computations, the full NLP solution requires an average of 200 s of online computation. On the other hand, the direct variant requires a negligible amount of online time (around 0.94 CPU seconds for a single backsolve) with no additional background tasks. Similarly, the shifted variant requires a negligible online time of around 1.04 CPU 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 additional background time (272 CPU seconds on average) for the construction of the Schur complement. Although this is done in background and only once for N_{cycle} steps, it may still be important to reduce this computational cost in order to improve the overall performance of the controller.

Finally, we note that the proposed NLP sensitivity approaches lead to effective warm-starting strategies for background NLP. 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 particular, the shifted variant provides accurate warm-starts for all the sampling times and reduces the average number of background NLP iterations from 10 to about two. On the other hand, the direct variant has inconsistent active sets and does not lead to improved warm starts, although it does provide exact approximations at the end of the horizon, where no further active set changes take place.

6. CONCLUSIONS AND FUTURE WORK

With wider application and appreciation of nonlinear model predictive control (NMPC), there is a continuing need for advanced problem formulations and algorithms that lead to improved controller performance. This is especially important as NMPC assumes broader roles for online dynamic optimization beyond regulatory control. Here, we summarize dynamic optimization strategies and component algorithms that are currently used for NMPC and explore their computational complexity in addressing more challenging online applications. We note that a simultaneous collocation-based approach has desirable complexity characteristics, particularly since the dominant solution steps increase little more than linearly with problem size. Nevertheless, these calculations are time critical, so these advantages alone do not completely address future challenges. Instead, we consider the real-time iteration strategy recently developed in the context of multiple shooting and adapt it to the full-space, collocation-based approach.

In our context, real-time iterations can be analysed using convergence properties of barrier NLP algorithms and NLP sensitivity. These lead to strategies where the online component is very cheap, and requires only a single backsolve of the associated KKT system. As with the approach in [9], this approach is developed in both *direct* and *shifted* variants and the stability analysis developed in [21, 22] can be applied directly. On a large-scale industrial process case

DOI: 10.1002/rnc

study, we show that this approach reduces online computations from six CPU minutes to about 1.0 CPU second with almost no sacrifice in controller performance. *Therefore, this study demonstrates that large, rigorous dynamic optimization models can be incorporated efficiently for NMPC and solved online with minimal computational delay.*

A number of improvements can be considered for future work. First, active set handling needs to be improved. The direct variant handles active set changes poorly in the prediction horizon, while the shifted variant avoids this problem but with additional background cost. Here, the estimate for $u(\ell + j)$ can be improved by additional online iterations, by substituting the updated residuals of the *nonlinear* KKT equations ($f(v)$) into the right-hand side of (8). Second, the real-time iteration scheme needs to be analysed further to assess nominal and robust stability properties, using recent results from [4], and to generalize this approach to a broader class of online dynamic optimization problems. Finally, recent work [27, 28] has shown that real-time iteration concepts can also be applied to Moving Horizon Estimation problems. Using the same concepts of background NLP calculations and NLP sensitivity, significant reductions in online computation have been observed with no loss in performance for the estimator.

REFERENCES

1. Bartusiak RD. Nlmpc: a platform for optimal control of feed- or product-flexible manufacturing. *Assessment and Future Directions of NMPC*. Springer: Berlin, 2007; 367–382.
2. Nagy ZK, Franke R, Mahn B, Allgöwer F. Real-time implementation of nonlinear model predictive control of batch processes in an industrial framework. *Assessment and Future Directions of NMPC*. Springer: Berlin, 2007; 465–472.
3. Franke R, Doppelhamer J. Integration of advanced model based control with industrial it. *Assessment and Future Directions of NMPC*. Springer: Berlin, 2007; 399–406.
4. Allgöwer F, Findeisen R, Biegler L (eds). *Assessment and Future Directions of Nonlinear Model Predictive Control*. Springer: Berlin, 2007.
5. Grötschel M, Krumke S, Rambau J (eds). *Online Optimization of Large Systems*. Springer: Berlin, 2001.
6. Santos LO, Afonso P, Castro J, Oliveira N, Biegler LT. On-line implementation of nonlinear MPC: an experimental case study. *Control Engineering Practice* 2001; **9**:847–852.
7. Findeisen R, Allgöwer F. Computational delay in nonlinear model predictive control. *Proceedings of the International Symposium on Advanced Control of Chemical Processes, ADCHEM'03*, Hong Kong, 2004.
8. Diehl M, Findeisen R, Allgöwer F. A stabilizing real-time implementation of nonlinear model predictive control. *Real-Time PDE-Constrained Optimization*. SIAM: Philadelphia, PA, 2007; 25–52.
9. Bock HG, Diehl M, Kostina E, Schlöder JP. Constrained optimal feedback control of systems governed by large differential algebraic equations. *Real-Time PDE-Constrained Optimization*. SIAM: Philadelphia, PA, 2007; 3–24.
10. Kadam J, Marquardt W. Sensitivity-based solution updates in closed-loop dynamic optimization. *Proceedings of the DYCOPS 7 Conference*, Boston, MA, 2004.
11. Wächter A, Biegler LT. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming* 2006; **106**(1):25–57.
12. Pontryagin VV, Boltyanskii Y, Gamkrelidze R, Mishchenko E. *The Mathematical Theory of Optimal Processes*. Interscience Publishers Inc.: New York, NY, 1962.
13. Bryson AE, Ho YC. *Applied Optimal Control*. Hemisphere: Washington, DC, 1975.
14. Vassiliadis VS, Sargent RWH, Pantelides CC. Solution of a class of multistage dynamic optimization problems. *Industrial and Engineering Chemistry Research* 1994; **33**:2123–2133.
15. Flores-Tlacuahuac A, Biegler LT, Saldivar-Guerra E. Dynamic optimization of hips open-loop unstable polymerization reactors. *Industrial and Engineering Chemistry Research* 2005; **44**(8):2659–2674.
16. Biegler LT, Cervantes AM, Wächter A. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Sciences* 2002; **57**:575–593.
17. Jockenhövel T, Biegler LT, Wächter A. Dynamic optimization of the tennessee eastman process using the optcontrolcentre. *Computers and Chemical Engineering* 2003; **27**(11):1513–1531.
18. Hager WW. Runge–Kutta methods in optimal control and the transformed adjoint system. *Numerical Mathematics* 2000; **87**:247–282.
19. Kameswaram S, Biegler LT. Convergence rates for direct transcription of optimal control problems using collocation at radau points. *Computational Optimization and Applications* 2007, to appear.

DOI: 10.1002/rnc

20. Betts J. *Practical Methods for Optimal Control Using NLP*. SIAM: Philadelphia, PA, 2001.
21. Diehl M, Findeisen R, Bock HG, Schlöder JP, Allgöwer F. Nominal stability of the real-time iteration scheme for nonlinear model predictive control. *IEE Control Theory and Applications* 2005; **152**(3):296–308.
22. Diehl M, Bock HG, Schlöder JP. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization* 2005; **43**(5):1714–1736.
23. Fourer R, Gay D, Kernighan B. *AMPL*. The Scientific Press: South San Francisco, 1993.
24. Forsgren A, Gill PE, Wright MH. Interior methods for nonlinear optimization. *SIAM Review* 2002; **44**(4):525–597.
25. Fiacco AV. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Academic Press: New York, 1983.
26. Cervantes AM, Tonelli S, Brandolin A, Bandoni JA, Biegler LT. Large-scale dynamic optimization for grade transitions in a low density polyethylene plant. *Computers and Chemical Engineering* 2002; **26**:227–237.
27. Kraus T, Kühl P, Wirsching L, Bock HG, Diehl M. A moving horizon state estimation algorithm applied to the Tennessee Eastman benchmark process. *IEEE Conference on Multisensor Fusion and Integration*, Heidelberg, Germany, 3–6 September 2006; Paper TuB03.4.
28. Zavala VM, Laird CD, Biegler LT. A fast computational framework for large-scale moving horizon estimation. *Proceedings of 8th International Symposium on Dynamics and Control of Process Systems*, Cancun, Mexico, June 2007.

DOI: 10.1002/rnc