Math. Prog. Comp. (2012) 4:307–331
DOI 10.1007/s12532-012-0043-2

FULL LENGTH PAPER

Optimal sensitivity based on IPOPT

Hans Pirnay · Rodrigo López-Negrete · Lorenz T. Biegler

Received: 14 April 2011 / Accepted: 22 May 2012 / Published online: 20 June 2012

Abstract We introduce a flexible, open source implementation that provides the optimal sensitivity of solutions of nonlinear programming (NLP) problems, and is adapted to a fast solver based on a barrier NLP method. The program, called *sIPOPT* evaluates the sensitivity of the Karush–Kuhn–Tucker (KKT) system with respect to perturbation parameters. It is paired with the open-source IPOPT NLP solver and reuses matrix factorizations from the solver, so that sensitivities to parameters are determined with minimal computational cost. Aside from estimating sensitivities for parametric NLPs, the program provides approximate NLP solutions for nonlinear model predictive control and state estimation. These are enabled by pre-factored KKT matrices and a fix-relax strategy based on Schur complements. In addition, reduced Hessians are obtained at minimal cost and these are particularly effective to approximate covariance matrices in parameter and state estimation problems. The *sIPOPT* program is demonstrated on four case studies to illustrate all of these features.

Keywords NLP · Sensitivity · Interior point

Mathematics Subject Classification (2000) 90C30 · 90C31 · 90C51 · 90-08

H. Pirnay
AVT-Process Systems Engineering, RWTH Aachen University,
Turmstr. 46, Aachen 52064, Germany

R. López-Negrete · L. T. Biegler (✉)
Department of Chemical Engineering, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
e-mail: biegler@cmu.edu

1 Introduction

Sensitivity analysis for nonlinear programming problems is a key step in any optimization study. This analysis provides information on regularity and curvature conditions at Karush–Kuhn–Tucker (KKT) points, assesses which variables play dominant roles in the optimization, and evaluates first-order sensitivities (i.e., first derivatives or directional derivatives) of the solution vector with respect to perturbation parameters. Moreover, for NLP algorithms that use exact second derivatives, sensitivity can be implemented very efficiently within NLP solvers and provide valuable information with very little added computation. In this study we discuss a capability for sensitivity analysis that is coupled to IPOPT [1], a barrier NLP solver that incorporates a filter-based line search.

The basic sensitivity strategy for NLP solvers is derived through application of the implicit function theorem (IFT) to the KKT conditions of a parametric NLP. As shown in Fiacco [2], sensitivities can be obtained from a solution with suitable regularity conditions, merely by solving a linearization of the KKT conditions. Nevertheless, relaxation of some of these regularity conditions induces singularity and inconsistency in this linearized system, and may make the sensitivity calculation much more difficult. Reviews of sensitivity analysis can be found in Fiacco [2], Fiacco and Ishizuka [3], and Büskens and Maurer [4]. More advanced cases have been analyzed by Kyparisis [5], Kojima [6], Kojima and Hirabayashi [7], and Jongen et al. [8,9]. An early implementation of sensitivity analysis applied to barrier methods can be found in Fiacco and Ghaemi [10].

For over 25 years, NLP sensitivity analysis has been applied to numerous applications in process engineering, as these can be viewed as parametric programs, with parameters that represent uncertain data or unknown inputs. These applications include flowsheet optimization [11,12], steady state real-time optimization [13], robust optimization and parameter estimation, nonlinear model predictive control [14], and dynamic real-time optimization [15].

With improvements in fast Newton-based barrier methods, such as IPOPT, sensitivity information can be obtained very easily. As a result, fast on-line algorithms can be constructed where more expensive optimization calculations can be solved in advance, and fast updates to the optimum can be made on-line. This has motivated algorithms for control, estimation and dynamic optimization that execute large optimization models with very little on-line computation [16]. This has been especially successful for so-called *advanced step* strategies that work with large, sparse NLP solvers. To obtain the sensitivity information, we exploit the full space information available through the linearization of the KKT conditions that appears naturally in the solution process. The resulting code that incorporates this strategy, called *sIPOPT*, is implemented in C++ and appended to the NLP solver IPOPT.

This study is organized as follows. In Sect. 2, we derive the equations that provide sensitivity information from perturbation of the KKT conditions. In addition, the implementation of the numerical solution and the handling of changes in the active set are discussed. This is based on a fix-relax strategy that modifies the KKT system through a Schur complement strategy. A parametric programming case study is presented at the end of this section to illustrate the strategy and the software. Section 3 presents two case studies that showcase different facets of the implementation. The first example deals with a nonlinear model predictive controller (NMPC) for a continuous stirred tank reactor (CSTR), where set-point changes occur. In the next example, the sensitivity based update is applied to NMPC control of a large-scale distillation column model. Using the advanced step approach, the controller stays close to the ideal case, where we assumed that there are no computational delays. Section 4 then develops a straightforward procedure to extract the inverse of the reduced Hessian from the pre-factored KKT system. As an illustration, the last case study deals with a parameter estimation problem for a small dynamic system. Here the sensitivity-based extraction of the inverse reduced Hessian leads to an estimate of the covariance matrix for the estimated parameters. The paper then concludes with a discussion of these contributions and future work.

2 Sensitivity concepts and derivation

The following are well known properties of nonlinear programs. They are shown here for convenience as they will be used for the derivations in the following sections.

Consider the parametric nonlinear program of the form:

$$\min_x \; f(x; p) \tag{1a}$$

$$\text{s.t.} \quad c(x; p) = 0, x \geq 0 \tag{1b}$$

with the variable vector $x \in \mathbb{R}^{n_x}$, the vector of perturbation parameters $p \in \mathbb{R}^{n_p}$, and equality constraints $c(x; p) : \mathbb{R}^{n_x+n_p} \to \mathbb{R}^m$. The IPOPT NLP algorithm substitutes a barrier function for the inequality constraints and solves the following sequence of problems with $\mu_\ell \to 0$:

$$\min_x \; f(x; p) - \mu_\ell \sum_{i=1}^{n_x} ln(x_i) \tag{2a}$$

$$\text{s.t.} \quad c(x; p) = 0. \tag{2b}$$

The *sIPOPT* program relies on the IPOPT solver [1], which solves (1) through a sequence of barrier problems (2). Having $x^* = x(p_0)$, the solution of (1) with $p = p_0$ (the nominal value), *sIPOPT* computes the sensitivity information of this solution with respect to p, given by the first derivatives $\frac{dx(p_0)}{dp}$ and

$$\frac{df(x^*; p_0)}{dp} = \frac{\partial f(x^*; p_0)}{\partial p} + \frac{dx(p_0)}{dp}\frac{\partial f(x^*; p_0)}{\partial x}. \tag{3}$$

To obtain these derivatives, we first consider properties of the solutions of (1) obtained by IPOPT when $p = p_0$ [2,17]. We then derive the sensitivity calculation implemented in *sIPOPT* based on these properties in Sect. 2.2.

In Sect. 2.4, we consider finite perturbations of the parameters p, which can lead to changes in active sets for $x(p)$. To deal with this issue, we need to relax the conditions of Sect. 2.2 to consider cases where the active set is nonunique and first derivatives no longer exist. In particular, we discuss relaxed conditions under which *directional derivatives*, which reflect the sensitivity of the x^* to directional perturbations of p, can still be obtained. Although automatic calculation of these directional derivatives has not yet been implemented in *sIPOPT*, Sect. 2.5 describes an enabling tool, the "fix-relax" strategy, that allows more general sensitivity strategies to be implemented. Also, more details on the implementation and features of *sIPOPT* are described in Sect. 2.7.

2.1 Sensitivity properties

For the NLP (1), the KKT conditions are defined as:

$$\nabla_x L(x^*, \lambda^*, \nu^*; p_0) = \nabla_x f(x^*; p_0) + \nabla_x c(x^*; p_0)\lambda^* - \nu^* = 0 \tag{4a}$$

$$c(x^*; p_0) = 0 \tag{4b}$$

$$0 \leq \nu^* \perp x^* \geq 0 \tag{4c}$$

For the KKT conditions to serve as necessary conditions for a local minimum of (1), constraint qualifications are needed, such as the Linear Independence Constraint Qualification (LICQ) or the Mangasarian–Fromowitz Constraint Qualification (MFCQ). Note that in the following definitions Q_j means the jth column of a given matrix Q, and I represents the identity matrix of appropriate dimension.

Definition 1 (*Linear independence constraint qualification (LICQ)*) Given a local solution of (1), x^* with a set of active bound indices $\mathscr{A}(x^*) \subseteq \{1, \dots, n\}$ with $x^*_j = 0,\ j \in \mathscr{A}(x^*)$ and $n_b = |\mathscr{A}(x^*)|$. These active bounds can be written as $E^T x^* = 0$, where $E_i = I_{j_i}$ for $i = 1, \dots, n_b$ and j_i is the i^{th} element of $\mathscr{A}(x^*)$. The LICQ is then defined by linear independence of the active constraint gradients, i.e.,

$$[\nabla_x c(x^*; p_0) \mid E]$$

has full column rank. Satisfaction of the LICQ leads to unique multipliers λ^*, ν^* at a KKT point.

Definition 2 (*Mangasarian–Fromowitz constraint qualification (MFCQ)*) Given a local solution of (1), x^* and an active set, with $E^T x^* = 0$, the MFCQ is defined by linear independence of the equality constraint gradients and the existence of a search direction d, such that:

$$E^T d > 0, \nabla_x c(x^*; p_0)^T d = 0.$$

In addition, we consider the following sufficient second-order conditions.

Definition 3 (*Strong second-order sufficient conditions (SSOSC)*) Given x^* and some multipliers $\lambda^*,\ \nu^*$ that satisfy the KKT conditions (4), then the SSOSC hold if

$$d^T \nabla_{xx} L(x^*, \lambda^*, \nu^*; p_0) d > 0 \text{ for all } d \neq 0 \text{ with } \nabla_x c(x^*)^T d = 0,$$
$$d_i = 0 \text{ for } \nu_i^* > 0. \tag{5}$$

We define $LM(x; p)$ as the set of multipliers (λ, ν) such that the KKT conditions hold at x for parameter p with (λ, ν) [5]. If the MFCQ and SSOSC hold at (x^*, p_0) then there exists a unique, continuous path of optimal points $x(p)$ of (1) in a neighborhood of (x^*, p_0) [18]. In addition, the point-to-set map $(x, p) \to LM(x; p)$ is then upper semi-continuous in a neighborhood of (x^*, p_0) and the set $LM(x(p); p)$ is a nonempty polyhedron [19]. Finally, we define $K_{ex}(x(p), p)$ as the set of vertex points of the polyhedron $LM(x(p); p)$.

The conditions defined above allow us to state the following property on the solution barrier problems (2).

Property 1 (Properties of the central path/barrier trajectory) *Consider problem (1) with $f(x; p)$ and $c(x; p)$ at least twice differentiable in x and once in p. Let x^* be a local constrained minimizer of (1) with the following sufficient optimality conditions at x^*:*

- *x^* is a KKT point that satisfies* (4),
- *the LICQ holds at x^*,*
- *strict complementarity (SC) holds at x^* for the bound multipliers ν^*, i.e., $x_i^* + \nu_i^* > 0$, and*
- *The SSOSC holds for x^*, λ^*, and ν^* that satisfy* (4).

If we now solve a sequence of barrier problems (2) *with $p = p_0$ and $\mu_\ell \to 0$, then:*

1. *There is at least one subsequence of solutions to* (2), *$x(\mu_\ell; p_0)$, converging to x^*.*
2. *For every convergent subsequence, the corresponding sequence of barrier multiplier approximations is bounded and converges to multipliers satisfying the KKT conditions for x^*.*
3. *A unique, continuously differentiable vector function $x(\mu, p_0)$, the minimizer of* (2), *exists for $\mu > 0$ in a neighborhood of $\mu = 0$.*
4. $\lim_{\mu \to 0^+} x(\mu; p_0) = x^*$.
5. $\|x(\mu, p_0) - x^*\| = O(\mu)$.

Proof The proof follows by noting that the LICQ implies the MFCQ and by invoking Theorem 3.12 and Lemma 3.13 in [17]. □

This property indicates that nearby solutions of (2) provide useful information for bounding properties for (1) for small positive values of μ. For such cases we now consider the sensitivity of these solutions with respect to changes in values of p.

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

- *$x^* = x(p_0)$ is an isolated minimizer and the associated multipliers λ^* and ν^* are unique.*

– *For some p in a neighborhood of p_0 there exists a k times differentiable function*

$$s(p)^T = [x(p)^T \ \lambda(p)^T \ \nu(p)^T]$$

that corresponds to a locally unique minimum for (1) *and* $s(p_0) = s^*$.

– *For p near p_0 the set of active inequalities is unchanged and complementary slackness holds.*

Proof The result follows directly from Theorem 3.2.2 and Corollary 3.2.5 in [2]. □

We now consider the barrier formulation and relate sensitivity results between (1) and (2) with the following result.

Property 3 (Barrier sensitivity properties) *For the barrier problem* (2) *assume that $f(x;p)$ and $c(x;p)$ are k times differentiable in p and $k+1$ times differentiable in x. Also, let the assumptions of Property* 1 *hold for problem* (1)*, then at the solution of* (2) *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 a k times differentiable function*

$$s(\mu; p)^T = \left[x(\mu; p)^T \ \lambda(\mu; p)^T \ \nu(\mu; p)^T\right]$$

that corresponds to a locally unique minimum for (2).

– $s(0; p_0) \equiv \lim_{\mu \to 0, p \to p_0} s(\mu; p) = s^*$.

Proof The result follows from Theorem 6.2.1 and Corollary 6.2.2 in [2]. These results were proved first for a mixed penalty function, but the proofs are easily modified to deal with barrier functions. □

2.2 Barrier sensitivity calculation

Calculation of the sensitivity of the primal and dual variables with respect to p now proceeds from the implicit function theorem (IFT) applied to the optimality conditions of (2) at p_0. Defining the quantities:

$$M(s(\mu; p_0)) = \begin{bmatrix} W(s(\mu; p_0)) & A(x(\mu; p_0)) & -I \\ A(x(\mu; p_0))^T & 0 & 0 \\ V(\mu; p_0) & 0 & X(\mu; p_0) \end{bmatrix} \tag{6}$$

and

$$N_p(s(\mu; p_0)) = \begin{bmatrix} \nabla_{xp} L(s(\mu; p_0)) \\ \nabla_p c(x(\mu; p_0))^T \\ 0 \end{bmatrix}, \quad N_\mu = \begin{bmatrix} 0 \\ 0 \\ -\mu e \end{bmatrix} \tag{7}$$

where $W(s(\mu; p_0))$ denotes the Hessian $\nabla_{xx}L(x, \lambda, \nu)$ of the Lagrangian function evaluated at $s(\mu; p_0)$, $A(x(\mu; p_0)) = \nabla_x c(x)$ evaluated at $x(\mu; p_0)$, $X = diag\{x\}$ and $V = diag\{\nu\}$, application of IFT leads to:

$$M(s(\mu; p_0))\frac{ds(\mu; p_0)}{dp}^T + N_p(s(\mu; p_0)) = 0. \tag{8}$$

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

$$\frac{ds(\mu; p_0)}{dp}^T = -M\left(s\left(\mu; p_0\right)\right)^{-1} N_p\left(s\left(\mu; p_0\right)\right). \tag{9}$$

Depending of the choice of the sparse solver selected in IPOPT, one can often check the assumptions of the LICQ, SC and SSOSC at the solution of (2) by using an estimate of the inertia of M (see [1]). Moreover, in IPOPT $M(s(\mu; p_0))$ is directly available in factored form from the solution of (2), so the sensitivity can be calculated through a simple backsolve. For small values of μ and $\|p - p_0\|$ it can be shown from the above properties [2] that

$$s(\mu; p) = s(\mu; p_0) - M(s(\mu; p_0))^{-1}N_p(s(\mu; p_0))(p - p_0) + o\|p - p_0\| \tag{10}$$

or

$$\begin{aligned} s(0; p) = s(\mu; p_0) - M(s(\mu; p_0))^{-1}\left[N_p(s(\mu; p_0))(p - p_0) + N_\mu\right] \\ +o\|p - p_0\| + o(\mu). \end{aligned} \tag{11}$$

Finally, the implementation of NLP sensitivity is simplified if the parameters can be separated in the NLP formulation. In this case, we write:

$$\min_{x,w} \; f(x, w) \tag{12a}$$
$$\text{s.t.} \quad c(x, w) = 0, \; x \geq 0 \tag{12b}$$
$$w - p_0 = 0 \tag{12c}$$

Note that the NLP solution is equivalent to (1), and it is easy to see that the NLP sensitivity is equivalent as well. Writing the KKT conditions for (12) gives:

$$\nabla_x f(x, w) + \nabla_x c^T(x, w)\lambda - \nu = 0 \tag{13a}$$
$$\nabla_w f(x, w) + \nabla_w c^T(x, w)\lambda + \bar{\lambda} = 0 \tag{13b}$$
$$c(x) = 0 \tag{13c}$$
$$XVe = 0 \tag{13d}$$
$$w - p_0 = 0. \tag{13e}$$

In (13a) $\bar{\lambda}$ represents the multiplier corresponding to the equation $w - p_0 = 0$. For the Newton step we write:

$$\begin{bmatrix} W & \nabla_{xw}L(x,w,\lambda,\nu) & A & -I & 0 \\ \nabla_{wx}L(x,w,\lambda,\nu) & \nabla_{ww}L(x,w,\lambda,\nu) & \nabla_w c(x,w) & 0 & I \\ A^T & \nabla_w c(x,w)^T & 0 & 0 & 0 \\ V & 0 & 0 & X & 0 \\ 0 & I & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta\lambda \\ \Delta\nu \\ \Delta\bar{\lambda} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \Delta p \end{bmatrix}. \tag{14}$$

Since $\Delta w = \Delta p$, it is easy to see that $[\Delta x^T \Delta\lambda^T \Delta\nu^T]^T$ from (14) is equivalent to $\frac{ds(\mu;p_0)}{dp}^T \Delta p$ obtained from (8) as $\mu \to 0$.

2.3 Directional derivatives

From Property 1, existence of ds^*/dp requires the SSOSC, SC and LICQ. When these assumptions fail to hold, we are still interested in cases where directional derivatives can be obtained. To deal with this case, we first need to consider another constraint qualification and more general second-order conditions.

Definition 4 (*Constant rank constraint qualification (CRCQ)*) Given a local solution of (1), x^* and an active set, $\mathscr{A}(x^*)$, then the CRCQ holds if for any subset $K \subset \mathscr{A}(x^*)$, the rank of matrix

$$[\nabla_x c(x;\, p) \mid E_K]$$

remains constant within a neighborhood of (x^*, p_0), where the i^{th} column of $E_{K,i} = I_{j_i}$ where $i = 1, \ldots |K|$ and j_i is the ith element of K.

Definition 5 (*General strong second-order sufficient conditions (GSSOSC)*) Suppose that the KKT conditions and the MFCQ hold at x^*, and multipliers $\lambda^*, \nu^* \in LM(x^*;\, p_0)$ satisfy the KKT conditions (4), then the GSSOSC hold if the SSOSC hold for every $\lambda^*, \nu^* \in LM(x^*;\, p_0)$.

The following property, due to Kyparisis [5], defines the existence of directional derivatives, and shows how they can be obtained.

Property 4 (Directional derivatives) *Suppose that the KKT conditions, MFCQ, CRCQ and GSSOSC hold at x^*, the solution of* (1) *with $p = p_0$. Then for p in some neighborhood of p_0, there exists a unique solution $x(p)$. Let $K_{ex}(x^*, p_0)$ be the set of extreme points of the multiplier values in $LM(x^*;\, p_0)$, then $x(p)$ is directionally differentiable in any direction $\Delta p \neq 0$ and the directional derivative Δx uniquely solves the following quadratic program (QPs), for some multipliers* $(\lambda, \nu) \in K_{ex}(x^*, p_0)$.

$$\min_{\Delta\bar{x}} \ \Phi = \Delta\bar{x}^T \nabla_{xp} L(s(p_0))\Delta p + \frac{1}{2}\Delta\bar{x}^T \nabla_{xx} L(s(p_0))\Delta\bar{x} \tag{15a}$$

$$\textit{s.t.} \quad \nabla_p c(x^*;\, p_0)^T \Delta p + \nabla_x c(x^*;\, p_0)^T \Delta\bar{x} = 0 \tag{15b}$$

$$\Delta\bar{x}_j \geq 0,\ j \in \mathscr{A}(x^*) \tag{15c}$$

$$\Delta\bar{x}_j = 0,\ \textit{for some}\ \nu_j \in K_{ex}(x^*, p_0),\ \textit{with}\ \nu_j > 0. \tag{15d}$$

This property provides the weakest assumptions under which a unique directional derivative can be shown [5,6]. Note that the GSSOSC assumption provides sufficient conditions for positive definiteness of the Hessian in the null space of the strongly active bounds (for every $x_i^* = 0$, there exists $\nu_i^* > 0$). This property, along with the MFCQ and the CRCQ, allows uniqueness of the directional derivative to be proved. Examples on the application of (15) to calculate directional derivatives are given in [5].

2.4 Active set changes

Now we want to consider finite perturbations ($\Delta p = p - p_0$) that lead to active set changes for the perturbed solution of (1). When Δp provokes an active set change, the bound of a positive variable, $x_j(p)$ may become active at zero or a zero variable may need to become positive. Here, even if the LICQ and SC hold at the solution at p_0, Δp may be too large to maintain the same active set for the perturbed estimate.

A widely applied approach is through the solution of the following QP [11,12,15,20]:

$$\min_{\Delta \bar{x}} \; \Phi = \Delta \bar{x}^T \nabla_{xp} L(s(p_0)) \Delta p + \frac{1}{2} \Delta \bar{x}^T \nabla_{xx} L(s(p_0)) \Delta \bar{x} \tag{16a}$$

$$\text{s.t.} \quad \nabla_p c(x^*; p_0)^T \Delta p + \nabla_x c(x^*; p_0)^T \Delta \bar{x} = 0, \;\; x^* + \Delta \bar{x} \geq 0 \tag{16b}$$

Of course, for non-linear problems with an arbitrary Δp there are no guarantees that the solution of this QP has any relation to the sensitivity of the original problem. For example, the Hessian of the QP might not be positive definite in the null space of the strongly active bounds. In practice this approach may be quite effective if Δp is sufficiently small. However, if Δp is too large, the local information through the KKT matrix of the optimal solution, on which *sIPOPT* in its current form is based, is not sufficient. In this case, path following methods like those presented in [18] have to be applied.

2.5 Fix-relax strategy

The current version of *sIPOPT* does not include an implementation to solve (15) or (16). Nevertheless, from the solution of the barrier problems, *sIPOPT* provides the elements of a QP-based sensitivity approach through the use of a "fix-relax strategy" which accounts for active set changes [21]. Moreover, a full-space, dual-feasible QP algorithm, called QPSchur, has already been developed that is based on fix-relax steps [22]. As a result, we do not present a detailed discussion of QP methods for (15) or (16) here. Instead we describe the fix-relax strategy for two important cases, addition of an active bound, and relaxation of an active bound, which form the ingredients of a QP solver.

First, when a perturbed variable (determined from (8)) violates its bound, i.e., $x_i = x_i^* + \Delta x_i < 0$, an additional condition is introduced that sets the perturbed variable x_i to its bound. Here we write $\Delta x_i + x_i^* = e_{x_i}^T \Delta x + x_i^* = 0$ and define the corresponding unit vector e_{x_i}. At the same time, the corresponding complementarity condition in (8) is relaxed with the addition of a new variable $\Delta \bar{\nu}$.

$$\left[\begin{array}{ccc|c} W & A & -I_n & 0 \\ A^T & 0 & 0 & 0 \\ V & 0 & X & e_{x_i} \\ \hline e_{x_i}^T & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \Delta x \\ \Delta \lambda \\ \Delta \nu \\ \hline \Delta \bar{\nu} \end{array}\right] = - \left[\begin{array}{c} \nabla_{xp} L \Delta p \\ \nabla_p c^T \Delta p \\ 0 \\ \hline x_i^* \end{array}\right] \tag{17}$$

Second, when a perturbed bound multiplier (determined from (8)) becomes negative, i.e., $\nu_i = \nu_i^* + \Delta \nu_i < 0$, the bound multiplier is set to zero, i.e., $\Delta \nu_i + \nu_i^* = e_{\nu_i}^T \Delta \nu + \nu_i^* = 0$, with unit vector e_{ν_i}. Also, the complementarity condition is relaxed, and again a new variable $\Delta \bar{\nu}$ is added.

$$\left[\begin{array}{ccc|c} W & A & -I_n & 0 \\ A^T & 0 & 0 & 0 \\ V & 0 & X & e_{\nu_i} \\ \hline 0 & 0 & e_{\nu_i}^T & 0 \end{array}\right] \left[\begin{array}{c} \Delta x \\ \Delta \lambda \\ \Delta \nu \\ \hline \Delta \bar{\nu} \end{array}\right] = - \left[\begin{array}{c} \nabla_{xp} L \Delta p \\ \nabla_p c^T \Delta p \\ 0 \\ \hline \nu_i^* \end{array}\right] \tag{18}$$

With suitable definition of $M,\ E,\ r_s$ and r_1, the systems (17) and (18) can be written in the following form:

$$\left[\begin{array}{c|c} M & E \\ \hline E^T & 0 \end{array}\right] \left[\begin{array}{c} \Delta s \\ \hline \Delta \bar{\nu} \end{array}\right] = - \left[\begin{array}{c} r_s \\ \hline r_1 \end{array}\right] \tag{19}$$

and can be solved using a Schur decomposition for Δs. This system can be solved in two steps by defining a Schur Matrix C:

$$C = -E^T M^{-1} E \tag{20}$$

and solving two linear systems

$$C \Delta \bar{\nu} = E^T M^{-1} r_s - r_1 \tag{21}$$

$$M \Delta s = -\left(r_s + E \Delta \bar{\nu}\right) \tag{22}$$

An example for relaxing a bound is given in [21], and the process of activating a bound is illustrated in Sect. 2.8.

2.6 Multiple sequential parameter perturbations

In the derivations in the previous sections we considered changes to the parameter vector. However, in some cases we may be interested in making multiple parameter perturbations in a sequential manner. For example we may want to perturb the current solution $s\,(\mu;\, p_0)$ using the parameter vectors $p_1, \ldots, p_{n_{\text{pert}}}$. This amounts to solving system (8) with different right hand sides $N_p\,(s\,(\mu;\, p_0))$ (Eq. (7)). Note that, because we already have (6) factorized at the solution, it is very cheap to obtain the n_{pert} sensitivities. From these and Equation (10) (or (11)) we can determine the approximated solutions $s\,(\mu;\, p_1), \ldots, s\left(\mu;\, p_{n_{\text{pert}}}\right)$.

2.7 Implementation and software

The *sIPOPT* library is implemented in C++, and is appended to the IPOPT solver. The library takes full advantage of solver options and more general problem formulations (e.g., double bounded constraints and variables), as well as the factorized matrix used by IPOPT, without regard for the sparse matrix solver or particular decomposition used. When the optimal solution is found, the factorized KKT matrix at the solution is held, and it then becomes available to *sIPOPT* as well as the sparse linear solver used. Moreover, the library provides a basic implementation of the Schur Complement required by the fix-relax strategy.

The code can be obtained directly from the IPOPT website https://projects.coin-or.org/Ipopt as part of the contributed software section, and the simplest way to access the *sIPOPT* library is through the AMPL [23] interface. This is done by way of creating suffixes that communicate the parameter values and perturbations to IPOPT. AMPL also provides exact first and second derivatives of the NLP, which are necessary to obtain the sensitivity based updates. A C++ interface is also provided, but in this case the derivatives must be provided by the user. More details on the implementation via AMPL and C++ are described in the documentation found in [24].

The *sIPOPT* library in its current form was designed to work with a default IPOPT installation. This constraint led to the introduction of parameters as variables, which are then fixed through additional equations as shown in (12). While convenient, this formulation increases the size of the original problem that IPOPT has to solve. It also has the disadvantage of making the optimization problem more difficult to solve. To overcome this inefficiency, we have created a branch of IPOPT development, which allows the explicit declaration of parameters. This development is also motivated by the concept that the code will be used as a building block in more complex algorithms, such as path-following of parametric solutions for (1). This code is not yet as mature as the original version, but a development version can be downloaded at https://github.com/athrpf/sipopt2. Just like IPOPT, all code is open source and published under the Eclipse Public License.

2.8 Example problem

To conclude this section we consider a small parametric optimization problem from [11] and also considered in [21]. Here we discuss in detail the updating procedure, and also cover a change in the active set. This problem illustrates the capabilities of the fix-relax strategy described above.

Consider the parametric programming problem:

$$\begin{aligned} \min \quad & x_1^2 + x_2^2 + x_3^2 \\ \text{s.t.} \quad & 6x_1 + 3x_2 + 2x_3 - p_1 = 0 \\ & p_2 x_1 + x_2 - x_3 - 1 = 0 \\ & x_1, x_2, x_3 \geq 0, \end{aligned} \tag{23}$$

with variables x_1, x_2, and x_3 and parameters p_1, and p_2. As programmed, the IPOPT code does not distinguish variables from parameters, but the problem can be reformulated as (12) by introducing equations that fix the parameters p_1, p_2 to their nominal values $p_{1,a}$, $p_{2,a}$.

$$\min \quad x_1^2 + x_2^2 + x_3^2 \tag{24a}$$

$$\text{s.t.} \quad 6x_1 + 3x_2 + 2x_3 - p_1 = 0 \tag{24b}$$

$$p_2 x_1 + x_2 - x_3 - 1 = 0 \tag{24c}$$

$$p_1 = p_{1,a} \tag{24d}$$

$$p_2 = p_{2,a} \tag{24e}$$

$$x_1, x_2, x_3 \geq 0. \tag{24f}$$

The KKT conditions for this problem are

$$2x_1 + 6\lambda_1 + p_2\lambda_2 - \nu_1 = 0 \tag{25a}$$

$$2x_2 + 3\lambda_1 + \lambda_2 - \nu_2 = 0 \tag{25b}$$

$$2x_3 + 2\lambda_1 - \lambda_2 - \nu_3 = 0 \tag{25c}$$

$$-\lambda_1 + \lambda_3 = 0 \tag{25d}$$

$$\lambda_2 x_1 + \lambda_4 = 0 \tag{25e}$$

$$6x_1 + 3x_2 + 2x_3 - p_1 = 0 \tag{25f}$$

$$p_2 x_1 + x_2 - x_3 - 1 = 0 \tag{25g}$$

$$p_1 - p_{1,a} = 0 \tag{25h}$$

$$p_2 - p_{2,a} = 0 \tag{25i}$$

$$\nu_1 x_1 - \mu = 0 \tag{25j}$$

$$\nu_2 x_2 - \mu = 0 \tag{25k}$$

$$\nu_3 x_3 - \mu = 0 \tag{25l}$$

$$x_1, x_2, x_3, \nu_1, \nu_2, \nu_3 \geq 0. \tag{25m}$$

The corresponding Newton step is

$$\left[\begin{array}{cccccccccccc}
2 & & & & \lambda_2 & 6 & p_2 & & & -1 & & \\
 & 2 & & & & 3 & 1 & & & & -1 & \\
 & & 2 & & & 2 & -1 & & & & & -1 \\
 & & & & & -1 & & 1 & & & & \\
\lambda_2 & & & & & & x_1 & & 1 & & & \\
6 & 3 & 2 & -1 & & & & & & & & \\
p_2 & 1 & -1 & & x_1 & & & & & & & \\
\nu_1 & & & & & & & & & x_1 & & \\
 & \nu_2 & & & & & & & & & x_2 & \\
 & & \nu_3 & & & & & & & & & x_3 \\
 & & & 1 & & & & & & & & \\
 & & & & 1 & & & & & & &
\end{array}\right]
\left[\begin{array}{c}
\Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta p_1 \\ \Delta p_2 \\ \Delta \lambda_1 \\ \Delta \lambda_2 \\ \Delta \nu_1 \\ \Delta \nu_2 \\ \Delta \nu_3 \\ \Delta \lambda_3 \\ \Delta \lambda_4
\end{array}\right]
= - \left[\begin{array}{c}
2x_1^* + 6\lambda_1^* + p_2\lambda_2^* - \nu_1^* \\
2x_2^* + 3\lambda_1^* + \lambda_2^* - \nu_2^* \\
2x_3^* + 2\lambda_1^* - \lambda_2^* - \nu_3^* \\
-\lambda_1^* + \lambda_3^* \\
\lambda_2^* x_1^* + \lambda_4^* \\
6x_1^* + 3x_2^* + 2x_3^* - p_1^* \\
p_2^* x_1^* + x_2^* - x_3^* - 1 \\
\nu_1^* x_1^* - \mu \\
\nu_2^* x_2^* - \mu \\
\nu_3^* x_3^* - \mu \\
p_1^* - p_{1,a} \\
p_2^* - p_{2,a}
\end{array}\right] \tag{26}$$

where the right hand side is zero at the solution (note that the ordering of the variables is now as in Eq. (14)). Also, note that this Newton step can easily be transformed into (14) by rearranging rows and columns. Now consider the exact solutions of two neighboring parameter sets $p_a = \left[p_{1,a}, p_{2,a}\right] = [5, 1]$ and $p_b = [4.5, 1]$. Using the "bound relax" option in IPOPT to satisfy bounds exactly, the corresponding NLP solutions are

$$s(p_a) = [0.6327, 0.3878, 0.0204, 5, 1, | - 0.1633, -0.2857, |0, 0, 0, | - 0.1633, 0.1808],$$

and

$$s(p_b) = [0.5, 0.5, 0, 4.5, 1, |0, -1, |0, 0, 1, |0, 0.5].$$

Clearly, there is a change in the active set, when changing the parameters from p_a to p_b. This is easily verified from the decrease of x_3 to zero. When using p_b the bound is active, while it is inactive when the parameters are set to p_a. This change in the active set is not captured by the linearization of the KKT conditions. For example, using (10) would require that we first solve the linear system (9), and use the solution to update the values of the variables. In this example we would solve the linear system

$$\begin{bmatrix} 2 & & & & \lambda_2 & 6 & p_2 & -1 & & & & \\ & 2 & & & & 3 & 1 & & -1 & & & \\ & & 2 & & & 2 & -1 & & & -1 & & \\ & & & & & -1 & & & & & 1 & \\ \lambda_2 & & & & & & x_1 & & & & & 1 \\ 6 & 3 & 2 & -1 & & & & & & & & \\ p_2 & 1 & -1 & & x_1 & & & & & & & \\ \nu_1 & & & & & & & x_1 & & & & \\ & \nu_2 & & & & & & & x_2 & & & \\ & & \nu_3 & & & & & & & x_3 & & \\ & & & 1 & & & & & & & & \\ & & & & 1 & & & & & & & \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta p_1 \\ \Delta p_2 \\ \Delta \lambda_1 \\ \Delta \lambda_2 \\ \Delta \nu_1 \\ \Delta \nu_2 \\ \Delta \nu_3 \\ \Delta \lambda_3 \\ \Delta \lambda_4 \end{bmatrix} = - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \Delta p_1 \\ 0 \end{bmatrix}, \tag{27}$$

where $\Delta p_1 = p_{1,a} - p_{1,b} = 5 - 4.5$. This Newton step yields an updated iterate of

$$\begin{aligned} s(p_b) = [&0.5765, 0.3775, -0.0459, 4.5, 1, | - 0.1327, \\ &-0.3571, |0, 0, 0, | - 0.1327, 0.2099] + o(\|\Delta p\|) \end{aligned} \tag{28}$$

which violates the bounds on x_3. On the other hand, taking into consideration the active set change, we use (17) and augment the KKT system fixing the variable to the bound, and relaxing the complementarity condition. The Newton step now becomes:

$$\left[\begin{array}{cccccccccccc|c} 2 & & & & \lambda_2 & 6 & p_2 & -1 & & & & & \\ & 2 & & & & 3 & 1 & & -1 & & & & \\ & & 2 & & & 2 & -1 & & & -1 & & & \\ & & & & & -1 & & & & & 1 & & \\ \lambda_2 & & & & & & x_1 & & & & & 1 & \\ 6 & 3 & 2 & -1 & & & & & & & & & \\ p_2 & 1 & -1 & & x_1 & & & & & & & & \\ \nu_1 & & & & & & & x_1 & & & & & \\ & \nu_2 & & & & & & & x_2 & & & & \\ & & \nu_3 & & & & & & & x_3 & & & 1 \\ & & & 1 & & & & & & & & & \\ & & & & 1 & & & & & & & & \\ \hline & & 1 & & & & & & & & & & \end{array}\right] \left[\begin{array}{c} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta p_1 \\ \Delta p_2 \\ \Delta \lambda_1 \\ \Delta \lambda_2 \\ \Delta \nu_1 \\ \Delta \nu_2 \\ \Delta \nu_3 \\ \Delta \lambda_3 \\ \Delta \lambda_4 \\ \hline \Delta \bar{\nu}_3 \end{array}\right] = - \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \delta p_1 \\ 0 \\ \hline x_3^* \end{array}\right], \tag{29}$$

which yields an updated iterate of

$$s(p_b) = [0.5, 0.5, 0, 4.5, 1, |0, -1, |0, 0, 0, |0, 0.5948] + o(\|\Delta p\|), \tag{30}$$

and this is a very good approximation to the optimal solution to the problem with problem data p_b. Some differences are expected for the linear system, as λ_4, x_1 and λ_2 appear in the nonlinear constraint (25e) in the KKT conditions.

3 Advanced step NMPC

In this section we describe the advanced step nonlinear model predictive control (asNMPC) strategy that makes direct use of IPOPT coupled with the sensitivity code, *sIPOPT*. Following a brief description of this strategy, we apply the advanced step control scheme to the continuous stirred tank reactor (CSTR) model proposed by [25]. A similar case study was used by [21] to illustrate stability and optimality results for sensitivity based NMPC. In addition, we then consider a large-scale case study where a distillation column is modeled.

In this section we assume that the dynamics of the plant can be described with the following discrete time dynamic model

$$z_{k+1} = f\left(z_k, u_k\right), \tag{31}$$

with initial condition z_0, and where $z_k \in \mathbb{R}^{N_z}$ are the state variables and $u_k \in \mathbb{R}^{N_u}$ are the control inputs. In order to drive the system to a desired operating condition we solve a dynamic optimization problem that generates the needed optimal input trajectory. The NMPC formulation used here solves a dynamic optimization problem over a predictive horizon of the following form:

$$\min \sum_{l=0}^{N} \left(z_l - \hat{z}_l\right)^T P \left(z_l - \hat{z}_l\right) + \sum_{l=0}^{N-2} \left(u_l - u_{l+1}\right)^T Q \left(u_l - u_{l+1}\right) \tag{32a}$$

$$\text{s.t.}\quad z_{l+1} = f\left(z_l, u_l\right) \quad \forall\ l = 0, \ldots, N-1 \tag{32b}$$

$$z_0 = \bar{z}(k) \tag{32c}$$

$$z_l \in \mathbb{X}, \quad u_l \in \mathbb{U}, \tag{32d}$$

where $\hat{z}$ is the desired value of the state and P and Q are diagonal positive definite weight matrices of appropriate dimensions. Also, we have considered a zero order hold on the inputs, that is the input variables are constant at each sampling time.

At the kth sampling time, after the actual state of the system ($\bar{z}(k)$) is obtained (or estimated), we solve problem (32) using this state as the initial condition ($z_0 = \bar{z}(k)$). This generates the optimal control trajectory for the predictive time horizon. From this, we take the value from the first sampling time, u_0, and inject it into the plant as the actual input, $\bar{u}(k)$. Once the plant reaches the next sampling time the NLP is solved again, with the initial condition set to the current state of the system. However, by the time we obtain the next NLP solution the plant has moved away from $\bar{z}(k)$. This *computational delay* could potentially destabilize the system, and this motivates the need for faster NMPC methods that generate the input online with as little delay as possible.

To address this issue Advanced Step NMPC was proposed in [16]. At sampling time $k-1$, this strategy generates a prediction of the state of the system at time k (i.e., z_1 from (32b)) using the injected input, $\bar{u}(k-1)$ and the model of the plant. The prediction z_1 is used instead of $\bar{z}(k)$ as the initial condition of the plant in the NLP (32), and an approximate problem is solved in background between sampling times. Once the new state of the system is measured or estimated at sampling time

Table 1 Objective function weights, set-points, and variances used in the CSTR example

	Objective function weight	Set-point 1	Set-point 2	Set-point 3	Variance
C_A (mol/L)	10^3	7.4699×10^{-1}	7.2222×10^{-1}	7.2222×10^{-1}	10^{-4}
T_R (K)	10^3	264.5327	253.0397	268.5262	10^{-2}
T_{cw} (K)	1	262.2587	252.5121	265.3109	10^{-2}
F (L/min)	10^3	–	–	–	–
F_{cw} (L/min)	10^3	–	–	–	–

k, Equation (10) or (11) is used to update the solution with the perturbed state, i.e., $\Delta z = \bar{z}(k) - z_1$. Since, we have the factorized KKT matrix from the NLP solved with z_1, the only online computational cost comes from solving the linear system given by Equation (16) using a new right hand side.

In the following subsections, two case studies are described to illustrate the use of asNMPC with *sIPOPT*. The first one consists of a small CSTR, and the second one considers a large-scale distillation column.

3.1 Example 1: Control of a CSTR

The first example is a non-isothermal CSTR where an exothermic reaction occurs between sodium thiosulfate and hydrogen peroxide. For this example the states are the concentration of sodium thiosulfate (C_A), the reactor temperature (T_R), and the cooling water temperature (T_{cw}). The model, as reported by [25], is given by the following index one system of differential-algebraic equations.

$$\frac{dC_A}{dt} = \frac{F}{V}\left(C_A^{in} - C_A\right) - 2k\left(T_R\right)C_A^2 \tag{33a}$$

$$\frac{dT_R}{dt} = \frac{F}{V}\left(T_R^{in} - T_R\right) + \frac{2(-\Delta H_R)k\left(T_R\right)C_A^2}{\rho C_P} - \frac{UA}{V\rho C_p}\left(T_R - T_{cw}\right) \tag{33b}$$

$$\frac{dT_{cw}}{dt} = \frac{F_{cw}}{V_{cw}}\left(T_{cw}^{in} - T_{cw}\right) + \frac{UA}{V_{cw}\rho_{cw}C_{pcw}}\left(T_R - T_{cw}\right) \tag{33c}$$

$$k\left(T_R\right) = k_o \exp\left(-\frac{E_a}{RT_R}\right) \tag{33d}$$

$$C_A \in [0, 1], \;\; T_R \in [200, 420], \;\; T_{cw} \in [200, 420]. \tag{33e}$$

For the NMPC formulation given by (32) a discrete time model is generated using orthogonal collocation on finite elements. For more details on this, the reader is referred to [26]. The control variables considered here are the feed (F) and cooling water (F_{cw}) flow rates. In the objective function (32a) the desired concentration and temperatures are considered, as well as the penalization of the inputs. In addition, zero mean Gaussian noise has been added to the plant behavior, and the variance associated with it is given in Table 1. The values of the desired set-points, as well as the objective function weights, and variance of each of the states are summarized in Table 1.

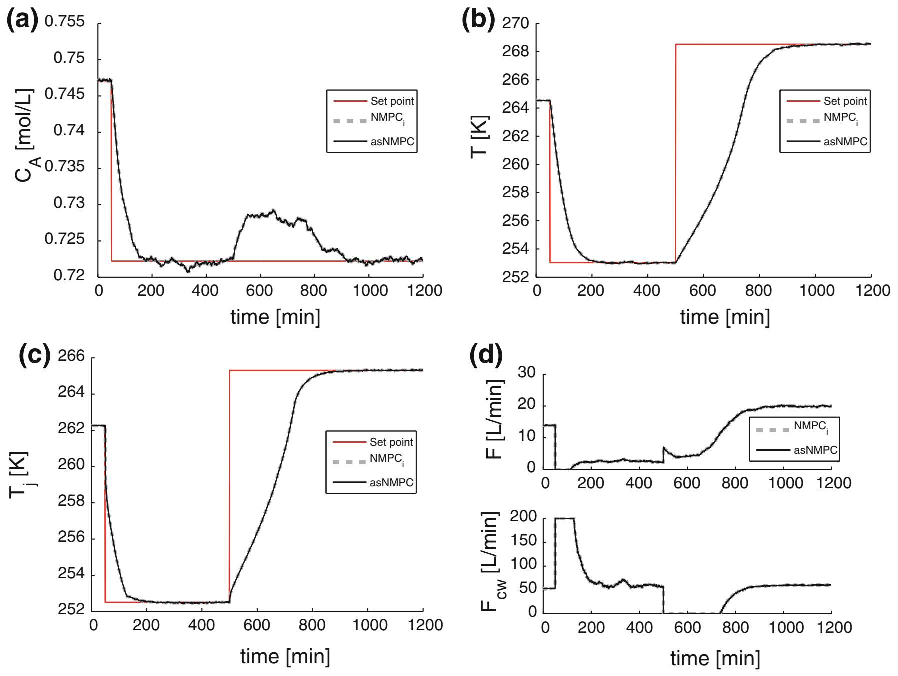

Fig. 1 asNMPC control of a CSTR **a** concentration of main component, **b** reactor temperature, **c** cooling water temperature, and **d** manipulated variables

Figure 1 shows the closed loop behavior of the CSTR. Two cases are shown for comparison: ideal NMPC (denoted by $NMPC_i$), where no computational delay is assumed, and asNMPC. In this case the average solution time was 0.05 CPU s (on an Intel *i*7-930 QuadCore CPU at 2.8 GHz), while the sensitivity update takes practically no time. Moreover, in this example the sensitivity update is very close to the optimal solution, and because of this the responses shown in Fig. 1 are all overlapping.

3.2 Example 2: Large-scale binary distillation column

In this section we present the application of asNMPC to a large-scale binary distillation problem proposed by [20]. The column is used to separate Methanol and n-Propanol, and it consists of 40 trays, a total condenser, and a reboiler.

The column is modeled with an index 1 system of differential algebraic (DAE) equations consisting of the so-called MESH equations (Mass balances, Equilibrium, Summation of compositions, and Heat balances). In this case the vapor molar hold up is assumed negligible. This assumption is well known to yield high index DAEs, and thus index reduction was performed. More details on this can be found in López-Negrete and Flores-Tlacuahuac [27] and Cervantes and Biegler [28]. Moreover, the equilibrium is modeled using Raoult's law, and non-ideal behavior is added through tray efficiencies. Finally, the liquid flow rate from the trays is modeled using Francis' weir equation. In total, the model consists of 84 differential equations and 168 algebraic equations. The reader is referred to Diehl [20] for more details on the model.

Table 2 Set-point information and objective function weights for the distillation column example

	Objective function weight	Set-point 1	Set-point 2
T_{14} (K)	10^4	351.0321	356.6504
T_{28} (K)	10^4	337.4470	346.9235
D (mol/s)	10^{-1}	1.1152	18.3859
Q_C (J)	10^{-1}	8.9955×10^5	1.6221×10^6
R (–)	10^2	–	–
Q_R (K)	10^2	–	–

For this example we considered 60 s sampling times, and 10 sampling times in the predictive horizon. As described in the CSTR example, the continuous time model is transformed into a discrete time model using collocation. Thus, using 3 point Radau collocation, the NLP consists of 19,814 variables and 19,794 equality constraints. To account for plant model mismatch we added noise to the differential variables (total molar holdup and liquid compositions at each tray). The noise was assumed uncorrelated, zero mean and Gaussian with variance 10^{-4} for the holdups and 10^{-6} for the compositions.

The control variables of this problem are the reboiler heat Q_R and the reflux rate of the top product R. In the objective function for the NLP we consider only temperatures from trays 14 and 28, since they are much more sensitive to changes than the head and bottom product streams. We also include the distillate flow rate and condenser heat duty in the objective. Note that we have numbered the trays from top to bottom. Also, we consider the penalization to changes in the inputs. Finally, set-point information and objective function weights are summarized in Table 2.

Figure 2 shows the simulation results comparing the advanced step NMPC with the ideal case. The average solution time of the NLP was 9.4 CPU s. Both the ideal and advanced step NMPC strategies were able to perform the set-point change. However, the latter adds practically no computational delays, and this is important for real time optimization schemes.

4 Parameter estimation

In this last section, we consider the capabilities of *sIPOPT* for recovery of reduced Hessian information, and this is of particular importance for parameter estimation problems to estimate the covariance of the calculated parameters. We begin with the extraction of reduced Hessian information from the solution of (1) with IPOPT. Following this, we consider the parameter estimation problem directly.

4.1 Extraction of reduced Hessian information

An important byproduct of the sensitivity calculation is information related to the Hessian of the Lagrange function pertinent to the second-order conditions. At the solution of (1) we consider a sensitivity system, $MS = N_{rh}$, with M defined in (6),

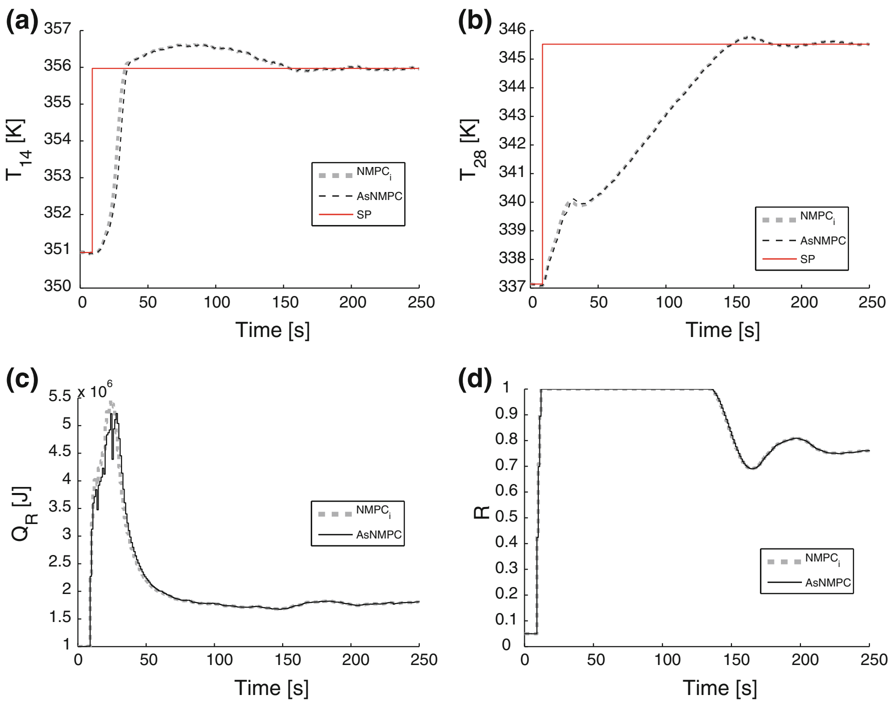

Fig. 2 asNMPC control of a binary distillation column. **a** Temperature of tray 14, **b** temperature of tray 28, **c** reboiler heat, and **d** reflux rate

and partition the variables into free and bounded variables, i.e., $x^* = [x_f^T \; x_b^T]$ where $x_f^* > 0$, $x_b^* = 0$ and $\nu_b^* > 0$, $\nu_f^* = 0$. Using (6) the sensitivity system can be partitioned with:

$$M = \begin{bmatrix} W_{ff}(x^*, \lambda^*) & W_{fb}(x^*, \lambda^*) & A_f(x^*) & -I_f & 0 \\ W_{bf}(x^*, \lambda^*) & W_{bb}(x^*, \lambda^*) & A_b(x^*) & 0 & -I_b \\ A_f(x^*)^T & A_b(x^*)^T & 0 & 0 & 0 \\ 0 & 0 & 0 & X_f^* & 0 \\ 0 & V_b^* & 0 & 0 & 0 \end{bmatrix}, S = \begin{bmatrix} S_{x_f} \\ S_{x_b} \\ S_\lambda \\ S_{\nu_f} \\ S_{\nu_b} \end{bmatrix},$$

$$\text{and } N_{rh} = \begin{bmatrix} T \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{34}$$

where the matrix T is defined below. From (34) it is easy to see that $S_{x_b} = 0$, $S_{\nu_f} = 0$. These variables and the last two rows can therefore be removed, leading to:

$$\begin{bmatrix} W_{ff}(x^*, \lambda^*) & A_f(x^*) & 0 \\ A_f(x^*)^T & 0 & 0 \\ W_{bf}(x^*, \lambda^*) & A_b(x^*) & -I_b \end{bmatrix} \begin{bmatrix} S_{x_f} \\ S_\lambda \\ S_{\nu_b} \end{bmatrix} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix}$$

We now define $S_{xf} = ZS_Z + YS_Y$, with $A_f(x^*)^T Z = 0$, and $A_f(x^*)^T Y$ and $R = [Y \mid Z]$ nonsingular. Here, we choose Z and Y by partitioning $x_f^T = [x_D^T \; x_I^T]$ with dependent and independent variables $x_D \in \mathbb{R}^m$, $x_I \in \mathbb{R}^{n_I}$, so that $A_f^T = [A_D^T \mid A_I^T]$, with A_D square and nonsingular. This leads to $Y^T = [I_m \mid 0]$, $Z^T = [-A_I(A_D)^{-1} \mid I_{n_I}]$.

Using the linear transformation, $H^T M H \tilde{S} = H^T N_{rh}$ with $H = \begin{bmatrix} R & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}$ and $\tilde{S} = H^{-1} S$ leads to:

$$\begin{bmatrix} Y^T W_{ff}(x^*, \lambda^*) Y & Y^T W_{ff}(x^*, \lambda^*) Z & Y^T A_f(x^*) & 0 \\ Z^T W_{ff}(x^*, \lambda^*) Y & Z^T W_{ff}(x^*, \lambda^*) Z & 0 & 0 \\ A_f(x^*)^T Y & 0 & 0 & 0 \\ W_{bf}(x^*, \lambda^*) Y & W_{bf}(x^*, \lambda^*) Z & A_b(x^*) & -I_b \end{bmatrix} \begin{bmatrix} S_Y \\ S_Z \\ S_\lambda \\ S_{\nu_b} \end{bmatrix} = \begin{bmatrix} Y^T T \\ Z^T T \\ 0 \\ 0 \end{bmatrix}. \tag{35}$$

From (35) we have $S_Y = 0$ and $S_Z = (Z^T W_{ff} Z)^{-1} Z^T T$. Choosing $Z^T T = I$ reveals S_Z as the inverse of the reduced Hessian matrix $H_R = Z^T W_{ff} Z$.

Note that the transformation of the sensitivity system (35) need not be implemented. Instead, for a chosen set of $n_I \le n_x - m$ independent variables, A_D nonsingular, $T^T = [0 \mid I_{n_I}]$ and the matrices defined in (34), the reduced Hessian can be found directly by solving $M\ S = N_{rh}$. From the choice of Z, $S_f = ZS_Z$ and $S_Z = H_R$, the reduced Hessian can be extracted easily from the rows of S. In the next section we will see that this property provides an efficient approach to extract an estimate of the covariance matrix for parameter estimation problems.

4.2 Reduced Hessian/Covariance relation

Consider the parameter estimation problem given by:

$$\min_{\theta,\gamma,y} \frac{1}{2}\left[(y - \hat{y})^T \hat{V}_y^{-1} (y - \hat{y}) + (\theta - \hat{\theta})^T \hat{V}_\theta^{-1} (\theta - \hat{\theta})\right] \tag{36a}$$

$$\text{s.t.} \quad c(\theta, \gamma, y) = 0 \tag{36b}$$

where we define the variables $x = [\theta^T, \gamma^T, y^T]^T$, y are state variables and θ and γ represent estimated parameters with and without prior measurement information, respectively. Measurements for the state variables y and parameters θ are denoted by $\hat{y}$ and $\hat{\theta}$, respectively, and covariance of these measurements is given by $\hat{V}_y$ and $\hat{V}_\theta$, respectively. KKT conditions for (36) are given by:

$$\hat{V}_y^{-1}(y^* - \hat{y}) + \nabla_y c(x^*)\lambda^* = 0 \tag{37a}$$

$$\hat{V}_\theta^{-1}(\theta^* - \hat{y}) + \nabla_\theta c(x^*)\lambda^* = 0 \tag{37b}$$

$$\nabla_\gamma c(x^*)\lambda^* = 0 \tag{37c}$$

$$c(x^*) = 0 \tag{37d}$$

As inequality constraints do not appear, problem (36) is a simplification of (1). Consequently, the set of bounded variables described in Sect. 4.1 is empty and the system (35) is simplified. Also, from (37a) to (37c) and from the LICQ assumption we define λ^* as:

$$\lambda^* = -(A^T A)^{-1} A \begin{bmatrix} \hat{V}_y^{-1}(y^* - \hat{y}) \\ \hat{V}_\theta^{-1}(\theta^* - \hat{\theta}) \\ 0 \end{bmatrix} \tag{38}$$

where $A = \nabla c(x^*)$. For suitable maximum likelihood assumptions on (36), $\mathbb{E}\left[y^* - \hat{y}\right] = 0$, $\mathbb{E}\left[\theta^* - \hat{\theta}\right] = 0$, Equation (38) leads to $\mathbb{E}\left[\lambda^*\right] = 0$.

Using the multiplier estimate $\lambda^* = 0$ we now consider the sensitivity of x^* to perturbations of the data, $\delta\hat{y}$ and $\delta\hat{\theta}$. Application of IFT to (37), as in (8) leads to the following sensitivity system:

$$\begin{bmatrix} \hat{V}_\theta^{-1} & 0 & 0 & A_\theta \\ 0 & 0 & 0 & A_\gamma \\ 0 & 0 & \hat{V}_y^{-1} & A_y \\ A_\theta^T & A_\gamma^T & A_y^T & 0 \end{bmatrix} \begin{bmatrix} \delta\theta \\ \delta\gamma \\ \delta y \\ \delta\lambda \end{bmatrix} = \begin{bmatrix} \hat{V}_\theta^{-1}\delta\hat{\theta} \\ 0 \\ \hat{V}_y^{-1}\delta\hat{y} \\ 0 \end{bmatrix}. \tag{39}$$

where $A^T = [A_\theta^T | A_\gamma^T | A_y^T | 0]$, and we assume that A_y is nonsingular. From

$$A^T \delta x = A_\theta^T \delta\theta + A_\gamma^T \delta\gamma + A_y^T \delta y = 0 \tag{40}$$

we can define $\delta_y = -A_y^{-T}(A_\theta^T \delta\theta + A_\gamma^T \gamma)$, and $\delta x = Z\begin{bmatrix} \delta\theta \\ \delta\gamma \end{bmatrix}$, with $A^T Z = 0$ and the choice:

$$Z = \begin{bmatrix} I & 0 \\ 0 & I \\ -A_y^{-T}A_\theta^T & -A_y^{-T}A_\gamma^T \end{bmatrix}.$$

Substituting δx into (39) leads to:

$$\left(Z^T \begin{bmatrix} \hat{V}_\theta^{-1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \hat{V}_y^{-1} \end{bmatrix} Z \right) \begin{bmatrix} \delta\theta \\ \delta\gamma \end{bmatrix} = Z^T \begin{bmatrix} \hat{V}_\theta^{-1}\delta\hat{\theta} \\ 0 \\ \hat{V}_y^{-1}\delta\hat{y} \end{bmatrix}, \tag{41}$$

where the matrix on the left hand side is the reduced Hessian, H_R. Solving for $\delta\theta$ and $\delta\gamma$ leads to:

$$\begin{bmatrix} \delta\theta \\ \delta\gamma \end{bmatrix} = H_R^{-1} \begin{bmatrix} \hat{V}_\theta^{-1}\delta\hat{\theta} - A_\theta A_y^{-1}\hat{V}_y^{-1}\delta\hat{y} \\ -A_\gamma A_y^{-1}\hat{V}_y^{-1}\delta\hat{y} \end{bmatrix}. \tag{42}$$

We can now recover the covariance for the parameters $\delta\theta$ and $\delta\gamma$ by noting the following expected values: $\hat{V}_y = \mathbb{E}\left[\delta\hat{y}\ \delta\hat{y}^T\right]$, $\hat{V}_\theta = \mathbb{E}\left[\delta\hat{\theta}\ \delta\hat{\theta}^T\right]$, and $\mathbb{E}\left[\delta\hat{y}\ \delta\hat{\theta}^T\right] = 0$. Taking expected values of $\begin{bmatrix} \delta\theta \\ \delta\gamma \end{bmatrix}[\delta\theta^T\ \ \delta\gamma^T]$, using (42), and simplifying the matrix algebra, leads to:

$$\mathbb{E}\left\{\begin{bmatrix} \delta\theta \\ \delta\gamma \end{bmatrix}\begin{bmatrix}\delta\theta^T & \delta\gamma^T\end{bmatrix}\right\} = V_{\theta,\gamma} = H_R^{-1}. \tag{43}$$

Consequently, finding the inverse of the reduced Hessian, as described in Sect. 4.1 directly determines an estimate of the covariance matrix for the parameter estimates.

4.3 Parameter estimation case study

To illustrate the approximation of the covariance using (35) and (43), we again consider the CSTR model described in equations (33a)–(33c). For this case study, we generated 14 datasets that were corrupted with Gaussian noise. These were then used to estimate two parameters, $\theta = [\Delta H_R, UA]$ the heat of reaction and the heat transfer coefficient multiplied by the area of the reactor, respectively. The values of these parameters used in the simulation to generate data were $-\Delta H_R = 5.9662 \times 10^5$ J/mol and $UA = 1.2 \times 10^6$ J/min $\cdot$ K. The parameters were estimated by solving the least squares problem, and gradually increasing the number of data sets used. The least squares problem has the following form:

$$\min_{\theta}\ \Phi = \frac{1}{2}\sum_{j=1}^{N_d}\sum_{l=0}^{N}\left(y_{l,j} - \hat{y}_{l,j}\right)^T V_y^{-1}\left(y_{l,j} - \hat{y}_{l,j}\right) \tag{44a}$$

$$\text{s.t.}\ \ z_{l+1,j} = f\left(z_{l,j}, u_{l,j}, \theta\right)\ \ \forall\ \ l = 0, \ldots, N-1,\ j = 1, \ldots, N_d \tag{44b}$$

$$y_{l,j} = g\left(z_{l,j}\right) \tag{44c}$$

$$z_{0,j} = z(0, j) \tag{44d}$$

$$z_{l,j} \in \mathbb{X},\quad u_{l,j} \in \mathbb{U},\quad y_{l,j} \in \mathbb{Y}, \tag{44e}$$

where N_d is the number of datasets, θ represents the parameters (in this case time independent), equation (44c) represents the model prediction of the measurement, $V_y = diag\left(10^{-4}, 10^{-2}, 10^{-2}\right)$ is the covariance matrix associated with the measurements, and $\hat{y}_l$ are the measurements. For this example we consider that $g\left(z_l\right) = z_l$, i.e., all three states were measured.

Having the covariance of the estimates allows us to quantify their reliability by analyzing the confidence region. Thus, following [29], we consider the Taylor series expansion of the objective function (44a)

$$\frac{1}{2}\left(\theta - \theta^*\right)^T V_\theta^{-1}\left(\theta - \theta^*\right) \approx \Phi(\theta) - \Phi(\theta^*) \leq c. \tag{45}$$

Table 3 Estimated parameters and the calculated standard deviation

Number of datasets	$-\Delta H_R$ (J/mol)	$\sigma_{-\Delta H_R}$ (J/mol)	UA (J/min K)	σ_{UA} (J/min K)
2	6.0480×10^5	8.4236×10^2	1.2248×10^6	1.1067×10^3
4	6.0851×10^5	5.9101×10^2	1.2297×10^6	1.1089×10^3
8	6.1762×10^5	4.1208×10^2	1.2529×10^6	1.1193×10^3
14	6.3153×10^5	3.0222×10^2	1.2835×10^6	1.1329×10^3

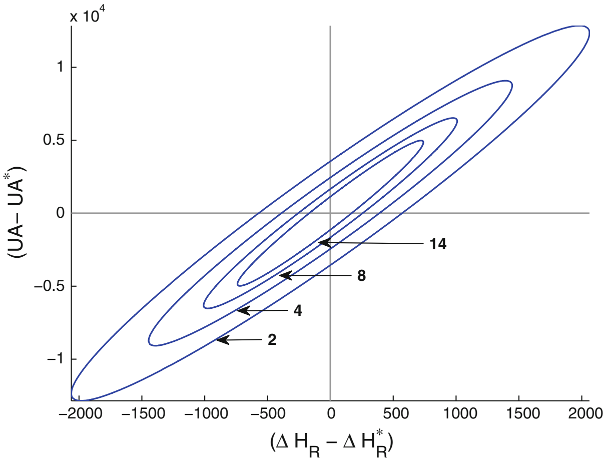

Fig. 3 Error ellipses for the estimated parameters, for when 2, 4, 8, and 14 datasets are used

This allows us to construct level sets of the objective function for increasing values of c. These ellipsoids are defined by the covariance of the estimated parameters. Moreover, it can be shown that, for large numbers of datasets with independent Gaussian measurement errors with known covariance V_y, then c is determined for the confidence level, β of a chi-square distribution χ^2 (n_θ, β), where n_θ is the number of degrees of freedom.

Equation (45) allows us to get order-of-magnitude estimates of the parameter confidence levels under different scenarios of practical interest. However, this inference analysis only applies to standard least squares formulations.

Table 3 shows the estimated parameters and the variance associated with them. Here we see that there is good agreement between the estimated parameters and the ones used for generating the datasets. Moreover, we can also note that the standard deviation of the estimates improves as more data are used. Figure 3 shows the 95 % confidence ellipsoids for the estimated parameters. The largest of these is for the case when only two datasets are used, the second largest for four datasets, and the innermost ellipse when 14 datasets are used. This figure shows clearly that as more data are used, the confidence region shrinks, and therefore our estimated parameters have less variance.

5 Conclusions

Sensitivity analysis is an essential component of any optimization study. It provides information on regularity and curvature conditions at KKT points and provides first-order estimates of the minimizer to parametric variations in nonlinear programs. With the availability of fast large-scale nonlinear programming solvers, it is also possible to integrate sensitivity capabilities to these solvers. This study discusses the development of *sIPOPT*, a program for NLP sensitivity that is paired with the IPOPT solver. Here we review properties of barrier methods and sensitivity analysis derived from the implicit function theorem. These require well-known regularity conditions including strict complementarity, strong second-order conditions and LICQ. We also relax the last condition and extend our method to include a fix-relax strategy, which is useful to handle active set changes; this is illustrated with a small example.

The *sIPOPT* program is also demonstrated on three large-scale case studies for real-time optimization and control. The first two case studies deal with the nonlinear control of chemical process units, where IPOPT solves a nonlinear program in background and *sIPOPT* provides an estimate of a neighboring solution, with initial states perturbed by noise or model mismatch. The third case study deals with parameter estimation of a dynamic system solved with IPOPT; this demonstrates how *sIPOPT* directly retrieves the covariance matrix of the estimated parameters (i.e., the reduced Hessian) in order to approximate the confidence region.

Written in C++, *sIPOPT* is an open source code that can be obtained directly from the contributed software section of the IPOPT website, and the simplest way to access it is through the AMPL interface. Future extensions to this code will deal with additional features that rely on Schur complement decompositions, including systematic treatment of active set changes. This will be especially useful for applications in nonlinear state estimation and control, as well as real-time optimization.

References

1. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
2. Fiacco, A.V.: Introduction to sensitivity and stability analysis in nonlinear programming. Mathematics in Science and Engineering, vol. 165. Academic Press, Dublin (1983)
3. Fiacco, A.V., Ishizuka, Y.: Sensitivity and stability analysis for nonlinear programming. Ann. Oper. Res. **27**, 215–236 (1990)
4. Büskens, H., Maurer, C. : Sensitivity analysis and real-time control of parametric control problems using nonlinear programming methods. In: Grötschel, M., Krumke, S., Rambau, J. (eds.) Online Optimization of Large-scale Systems, pp. 57–68. Springer, Berlin (2001)
5. Kyparisis, J.: Sensitivity analysis for nonlinear programs and variational inequalities with nonunique multipliers. Math. Oper. Res. **15**(2), 286–298 (1990)
6. Kojima, M.: Strongly stable stationary solutions in nonlinear programs. In: Robinson, S.M. (ed.) Analysis and Computation of Fixed Points, pp. 93–138. Academic Press, New York (1980)
7. Kojima, M., Hirabayashi, R.: Continuous deformation of nonlinear programs. Math. Program. Study **21**, 150–198 (1984)
8. Jongen, H.T., Jonker, P., Twilt, F.: Nonlinear Optimization in Finite Dimensions. Kluwer, Dordrecht (2000)
9. Jongen, H.T., Meer, K., Triesch, E.: Optimization Theory. Kluwer, Dordrecht (2004)

10. Fiacco, A.V. Ghaemi, A.: A user's manual for SENSUMT. A penalty function computer program for solution, sensitivity analysis and optimal bound value calculation in parametric nonlinear programs. Technical Report T-434, Management Science and Engineering, George Washington University (1980)
11. Ganesh, N., Biegler, L.T.: A reduced hessian strategy for sensitivity analysis of optimal flowsheets. AIChE **33**, 282–296 (1987)
12. Wolbert, D., Joulia, X., Koehret, B., Biegler, L.T.: Flowsheet optimization and optimal sensitivity analysis using exact derivatives. Comput. Chem. Eng. **18**, 1083 (1994)
13. Forbes, J., Marlin, T.E.: Design cost: a systematic approach to technology selection for model-based real-time optimization systems. Comput. Chem. Eng. **20**, 717–734 (1996)
14. Diehl, M., Findeisen, R., Allgöwer, F.: A stabilizing real-time implementation of nonlinear model predictive control. In: Biegler, L.T., Keyes, D., Ghattas, O., van Bloemen Waanders, B., Heinkenschloss, M. (eds.) Real-Time PDE-Constrained Optimization, pp. 25–52. SIAM, Philadelphia (2007)
15. Kadam, J. Marquardt, W.: Sensitivity-based solution updates in closed-loop dynamic optimization. In: Proceedings of the DYCOPS 7 Conference. Elsevier, Amsterdam (2004)
16. Zavala, V.M., Biegler, L.T.: The advanced-step NMPC controller: optimality, stability and robustness. Automatica **45**(1), 86–93 (2009)
17. Forsgren, A., Gill, P.E., Wright, M.H.: Interior point methods for nonlinear optimization. SIAM Rev. **44**(4), 525–597 (2002)
18. Guddat, J., Guerra Vazquez, F., Jongen, H.T.: Parametric Optimization: Singularities, Pathfollowing and Jumps. Teubner, Stuttgart (1990)
19. Robinson, S.M.: Generalized equations and their solutions, part II: applications to nonlinear programming. Math. Program. Study **19**, 200–221 (1982)
20. Diehl, M.: Real-Time Optimization for Large Scale Nonlinear Processes. Ph.D. thesis, Universität Heidelberg (2001). http://www.ub.uni-heidelberg.de/archiv/1659/
21. Zavala, V.M.: Computational Strategies for the Operation of Large-Scale Chemical Processes. Ph.D. thesis, Carnegie Mellon University (2008)
22. Bartlett, R.A., Biegler, L.T.: QPSchur: a dual, active-set, schur-complement method for large-scale and structured convex quadratic programming. Optim. Eng. **7**, 5–32 (2006)
23. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: a modeling language for mathematical programming. Duxbury Press, Pacific Grove (2002)
24. Pirnay, H., López-Negrete, R., Biegler, L.T.: *sIPOPT* Reference Manual. Carnegie Mellon University (2011). https://projects.coin-or.org/Ipopt
25. Rajaraman, S., Hahn, J., Mannan, M.S.: A methodology for fault detection, isolation, and identification for nonlinear processes with parametric uncertainties. Ind. Eng. Chem. Res. **43**(21), 6774–6786 (2004)
26. Biegler, L.T.: Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes. SIAM, Philadelphia (2010)
27. López-Negrete, R., Flores-Tlacuahuac, A.: Optimal start-up and product transition policies of a reactive distillation column. Ind. Eng. Chem. Res. **46**, 2092–2111 (2007)
28. Cervantes, A.M., Biegler, L.T.: Large-scale DAE optimization using a simultaneous NLP formulation. AIChE J. **44**(5), 1038–1050 (1998)
29. Bard, Y.: Nonlinear Parameter Estimation. Academic Press, New York (1974)