A new optimization algorithm with application to nonlinear MPC

Frode Martinsen [a] Lorenz T. Biegler [b] Bjarne A. Foss [a] *

[a]*Department of Engineering Cybernetics, NTNU, 7491 Trondheim, Norway.*

[b]*Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213.*

Abstract

This paper investigates application of SQP optimization algorithms to nonlinear model predictive control. It considers feasible vs. infeasible path methods, sequential vs. simultaneous methods and reduced vs. full space methods. A new optimization algorithm coined rFOPT which remains feasibile with respect to inequality constraints is introduced. The suitable choices between these various strategies are assessed informally through a small CSTR case study. The case study also considers the effect various discretization methods have on the optimization problem.

Key words: model predictive control, optimization, SQP, feasibility.

1 Introduction

Nonlinear model predictive control (NMPC) is a control strategy where application of nonlinear optimization methods is essential. This paper is application oriented, and contributes to the practical knowledge of implementation of NMPC. The paper focuses on application of sequential quadratic programming (SQP) optimization algorithms in NMPC, but also considers the interplay between choice of model discretization and optimization. The paper addresses the differences and similarities between feasible and infeasible path methods, sequential and simultaneous methods, and reduced and full space methods for solving the nonlinear programming (NLP) formulation for NMPC. Suitable choices between these various strategies are informally assessed by applying them to a case study, a CSTR with multiple steady states.

* Corresponding author. Tel +47-73594476 fax +47-73594399
E-mail: bjarne.a.foss@itk.ntnu.no

The theory of optimization algorithms is not dependent on how the equality constraints are formed. For instance in optimal control, and in particular in the special case of NMPC, much concern is put into discretization schemes for the nonlinear equality constraints. These equality constraints result from a continuous-time nonlinear dynamical system repeated over a time horizon P. Three major variants are usually considered: orthogonal collocation, multiple shooting and single shooting (possibly with a variable grid). [1] discuss the general benefits of these approaches, and [2] discusses this in conjunction with SQP algorithms. [3] presents a software packag, SOCS, for solution of open-loop optimal control problems using direct transcription. The SOCS package is available as a FORTRAN library or in Matlab through TOMLAB [4]. [5] addresses the dynamic optimization of general DAE systems. These approaches seek to find formulations of the equality constraints that are easier to satisfy, while simultaneously reducing the discretization error.

Nonlinear inequality constraints may be introduced for (nominal) stability purposes in NMPC ([6], [7], [8]). These references can be seen as extensions of earlier work ([9], [10]) where nominal stability (assuming an infinite prediction horizon for a stable system) was addressed as well as optimization algorithms for feasible path methods. In [7], example E conforms with the approach found in [10]. In essence, termination prior to convergence of the optimizer cannot guarantee nominal stability unless the equality constraints are satisfied.

The immediate answer to the need for early termination is single shooting, i.e., solve the model at each iteration with an initial value solver. Single shooting algorithms progress towards a solution by iterating between solving the model and solving a reduced size optimization problem. Due to this, single shooting is said to be a sequential method. Single shooting produces a reduced gradient problem in the free variables to be solved at each NMPC iteration. Maintaining feasibility of nonlinear inequalities involving dependent variables can then be obtained by use of RFSQP [11]. Single shooting may be costly if evaluation of the problem functions is costly, e.g. if an implicit discretization scheme must be applied. In addition single shooting lacks robustness when applied to unstable systems [1], section 4.1 and 4.6.2.

Simultaneous methods must be applied to solve optimization problems with stabilizing endpoint constraints. End-point constraints make the problem a two-point boundary value problem (TPBVP) which in general cannot be resolved with single shooting. Simultaneous methods do not solve the model at each iteration. Instead a simultaneous search for a model solution and optimal point is carried out. Multiple-shooting and orthogonal collocation, possibly on finite elements [12], are the most widely used simultaneous methods. Since simultaneous methods do not solve the model at each iteration, they cannot guarantee stability in the nominal stability setting of dual-mode or quasi-infinite horizon NMPC if terminated prior to convergence. A related multiple shooting, trust region strategy that maintains feasibility in NMPC problems was recently proposed in [13]. On the other hand, it was demonstrated by [14] that termination prior to convergence in multiple shooting may be viable for some applications. Decomposition strategies for orthogonal collocation on finite elements have been considered by [15] and [16].

We note that these algorithms can be applied to a number of NMPC problem formulations. For instance, the approach of [17] is meant only for systems with a flatness property. Here differential equations can be eliminated and the inputs and states of the problem can be represented through flat outputs and their derivatives. As a result, the system can be written as algebraic relations. The flat outputs can be written as polynomials on finite elements and all of the inputs and states can be recovered by optimization of these polynomial coefficients. The claim is that this leads to a much smaller optimization problem (but it is also completely dense). The algorithms developed in this paper can be applied equally well to NMPC formulations of problems that are represented by flat outputs.

The nonlinear MPC problem with $u^l \in \mathbb{R}^{n_u}$ and $x^l \in \mathbb{R}^{n_x}$

$$\begin{aligned} &\min_{x,u;l} \tfrac{1}{2}\left(\textstyle\sum_{l=0}^{P-1} \left\|x^{l+1}\right\|_Q^2 + \sum_{l=0}^{M-1} \left\|u^l\right\|_R^2\right) \\ &\textbf{s.t.} \qquad c_{\mathcal{E}}(x,u) = 0 \\ &\qquad\quad\ \ x \in \mathbb{X} \times \cdots \times \mathbb{X} \\ &\qquad\quad\ \ u \in \mathbb{U} \times \cdots \times \mathbb{U} \end{aligned} \tag{1}$$

with $M < P$ is considered in this paper. Provided that the system is given in continuous time, the variables (x, u) and the equality constraints in (1) are formed by assuming an appropriate discretization scheme of the continuous time DAE model $F(\dot{x}(t), x(t), u(t)) = 0$ repeated over the horizon P along with a suitable parameterization of the control profile. Endpoint constraints or augmentation of the objective may be included to guarantee nominal stability of the NMPC algorithm, see [8]. Nonlinear inequality constraints may be added for instance to provide stability or to model quality constraints. Observe that reference tracking and non-zero set-points can be handled in this framework with minor modifications. It is assumed that sufficiently smooth first principles state-space models with measured states and analytic first derivatives are used and that

$$(\mathcal{X}, \mathcal{U}) = (\mathbb{X} \times \cdots \times \mathbb{X}, \mathbb{U} \times \cdots \times \mathbb{U})$$

can be described by bounds.

The paper continues with a conceptual comparison between single shooting and reduced Hessian methods in section 2. In section 2.2 a new algorithm maintaining feasibility with respect to nonlinear inequalities is introduced. Simulation results follow in section 3. Discussion and conclusions follow in section 4.

2 Optimization methods

We first consider general SQP methods. The NMPC problem (1) can be restated as a general NLP problem [18]:

$$\begin{aligned} \min_z \quad & f(z) \\ \textbf{s.t.} \quad & c_{\mathcal{E}}(z) = 0 \\ & c_{\mathcal{I}}(z) \leq 0 \end{aligned} \tag{2}$$

where $z^T = [x^T \; u^T] \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $c_{\mathcal{E}} : \mathbb{R}^n \to \mathbb{R}^m$ and $c_{\mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^p$ where $n = n_x P + n_u M$, $m = n_x P$ and $p = 2n$ (assuming upper and lower bounds on (x^l, u^l) over the horizons P and M). The transpose of the Jacobian matrix of the equality constraints is denoted $A = A(z) = [\nabla c^1_{\mathcal{E}}(z), \nabla c^2_{\mathcal{E}}(z), \cdots, \nabla c^m_{\mathcal{E}}(z)]$ where $c^i_{\mathcal{E}}(z)$ is the i-th component of the vector $c_{\mathcal{E}}(z)$. The matrix $G(z)$ made up of A and the gradients of the active inequality constraints is assumed to have full column rank. The null-space of $G(z)^T$ defines the tangent space to the equality and active inequality constraints at z. We denote $H\mathcal{L}$ the Hessian of the Lagrangian function $\mathcal{L}(z, \lambda_{\mathcal{E}}, \lambda_{\mathcal{I}}) = f(z) + \lambda^T_{\mathcal{E}} c_{\mathcal{E}}(z) + \lambda^T_{\mathcal{I}} c_{\mathcal{I}}(z)$ where $\lambda_{\mathcal{E}}$ and $\lambda_{\mathcal{I}}$ are the multiplier vectors.

Please note that nonlinear constraints, e.g. due to quality constraints, will add nonlinear inequalities to c_I.

In SQP, a sequence of subproblems is solved, where for each subproblem the model is linearized and a quadratic model of the Hessian of the Lagrangian is formed about z_k. This gives the following quadratic programming (qp) problem to be solved at each iteration k of the SQP algorithm:

$$\begin{aligned} \min_{d_k} \quad & \nabla f(z_k)^T d_k + \tfrac{1}{2} d_k^T B_k d_k \\ \textbf{s.t.} \quad & \nabla c_{\mathcal{E}}(z_k)^T d_k + c_{\mathcal{E}}(z_k) = 0 \\ & \nabla c_{\mathcal{I}}(z_k)^T d_k + c_{\mathcal{I}}(z_k) \leq 0 \end{aligned} \tag{3}$$

where $B_k \succ 0$ is given by $H\mathcal{L}_k$, or its approximation. The solution d_k to (3) is a search direction and the SQP algorithm searches along d_k for a new iterate z_{k+1} that gives a reduction in a merit function. The merit function ϕ_ν is needed to give convergence to a point satisfying the strong second order assumptions, from any starting point under certain additional assumptions. Further details on SQP can be found in e.g. [19]. The basic SQP-algorithm is summarized in Algorithm 1.

```
(1) Guess $z_0$, set $B_0$.
(2) At $z_k$ evaluate $f(z_k)$, $c_{\mathcal{E}}(z_k)$, $c_{\mathcal{I}}(z_k)$, $\nabla f(z_k)$, $\nabla c_{\mathcal{E}}(z_k)$ and $\nabla c_{\mathcal{I}}(z_k)$.
(3) Update $B_k$, the Hessian of the Lagrange function $H\mathcal{L}_k$ or its approximation.
    (This approximation comes from a Gauss-Newton method or, if a certain curvature condition is satisfied, by a BFGS formula.)
(4) Solve (3) for $d_k$.
(5) Test for convergence.
(6) Find a step-size $\alpha_k$ s.t. $0 < \alpha_k \leq 1$ and $\phi_\nu(z_k + \alpha_k d_k) < \phi_\nu(z_k)$.
(7) Set $z_{k+1} = z_k + \alpha_k d_k$, $k = k + 1$ and go to 2.
```

Algorithm 1. *Basic SQP algorithm.*

2.1 Reduced gradient methods

In section 2.1, it is assumed that there are no nonlinear inequalities in problem (2). The qp (3) can be resolved in the full space of free and dependent variables, or in the reduced space of free variables by a suitable elimination of variables. Elimination of variables exploits that if $n_x P - n_u M \gg n_u M$ and that $n_u M$ is small, the reduced subproblem for the null-space step will be small (but dense). In the full space the sparsity of both the Hessian (which commonly requires analytic Hessians) and the Jacobian can be exploited to yield fast solutions [20]. This section shows that a reduced gradient approach can be derived by following two different strategies. The first uses a sequential approach, see e.g. [10], while the second follows the simultaneous null-space approach [21], [22].

2.1.1 Sequential approach (sOPT)

By iterating the model over the horizon P, the transformation $x = \Psi(x_0, u)$ allows the equivalent form [23]

$$\begin{aligned} \min_u \ & f_k^u(u) \\ \mathbf{s.t.} \ & \Psi(x_0, u) \in \mathcal{X} \\ & u \in \mathcal{U} \end{aligned} \tag{4}$$

The transformation $\Psi(\cdot)$ for linearized and discretized systems is essentially a projection onto the subspace $\mathcal{U}$. The sequential approach solves the model at each iteration, i.e. the qp-subproblem is solved following a feasible path strategy. Without active bounds, the KKT conditions for problem (1) are given by:

$$\begin{aligned} \nabla_x f_k + \nabla_x c_{\mathcal{E},k}^T \lambda_{\mathcal{E},k} &= 0 \\ \nabla_u f_k + \nabla_u c_{\mathcal{E},k}^T \lambda_{\mathcal{E},k} &= 0 \\ c_{\mathcal{E},k} &= 0 \end{aligned}$$

Details on the partial derivatives are given in [24]. We eliminate $\lambda_{\mathcal{E},k} = -\left(\nabla_x c_{\mathcal{E},k}\right)^{-T} \nabla_x f_k$, define $S_k^T = -\nabla_u c_{\mathcal{E},k}^T \left(\nabla_x c_{\mathcal{E},k}\right)^{-T}$ and get the reformulated KKT conditions

$$\nabla_u f_k + S_k^T \nabla_x f_k = 0$$
$$c_{\mathcal{E},k} = 0$$

Note that the above result also can be derived by considering the total differential of $c_{\mathcal{E},k} = 0$, which is $dc_{\mathcal{E}} = \nabla_x c_{\mathcal{E},k}^T dx + \nabla_u c_{\mathcal{E},k}^T du = 0$, and define $S_k^T = \left(\frac{dx}{du}\right)^T = -\nabla_u c_{\mathcal{E},k}^T \left(\nabla_x c_{\mathcal{E},k}\right)^{-T}$ (hence the notion of sensitivity [1]). In the view of problem (4), consider the objective and model derivatives with respect to u:

$$\begin{aligned} \frac{df_k}{du} &= \nabla_u f_k + \left(\tfrac{dx}{du}\right)^T \nabla_x f_k \\ &= \nabla_u f_k + S_k^T \nabla_x f_k \\ \frac{dc_{\mathcal{E},k}}{du} &= \nabla_u c_{\mathcal{E},k} + \nabla_x c_{\mathcal{E},k} S_k \end{aligned} \tag{5}$$

Note that the sensitivity matrix gives search directions $d_k^x = S_k d_k^u$, hence $\frac{dc_{\mathcal{E},k}}{du} d_k^u = (\nabla c_{\mathcal{E},k})^T d_k$ when $c_{\mathcal{E}}(x_k) = 0$. Also note that by defining $Z_k^T = [S_k^T \;\; I]$ we have $Z_k^T \nabla f_k^T = 0$ for the KKT conditions. Linearizing the reformulated KKT conditions with respect to u gives (with some abuse of notation) the Newton system:

$$(Z_k^T \nabla^2 f_k Z_k + \nabla Z_k^T \nabla f_k Z_k) d_k^u = -(\nabla_u f_k + S_k^T \nabla_x f_k)$$

Assuming $Z(z^*)^T \nabla f(z^*) = 0$ allows us to write the related bound constrained qp in $d_k^u \in \mathbb{R}^{n_u M}$:

$$\begin{aligned} \min_{d_k^u} &\left(\nabla_u f_k^T + S_k \nabla_x f_k^T\right) d_k^u \\ &+ \tfrac{1}{2} (d_k^u)^T \left(Z_k^T \nabla^2 f_k Z_k\right) d_k^u \\ \text{s.t.} &\begin{bmatrix} S_k \\ -S_k \end{bmatrix} d_k^u \leq \begin{bmatrix} x_U - x_k \\ x_k - x_L \end{bmatrix} \\ &u_k + d_k^u \in \mathcal{U} \end{aligned} \tag{6}$$

During the line search (as in step 7 of Algorithm 1), the algorithm evolves the model to get $x_{k+1} = \Psi(x_0, u_k + \alpha d_k^u)$. Then the algorithm solves the qp (6) for d_{k+1}^u which again is used to evolve the model giving x_{k+2} and so on.

[1] This definition of S gives the same result as that of [23], but is faster to compute in Matlab which benefits from vectorization. Note that the inversion is not necessary to numerically solve for S from $AS = B$.

The reduced Hessian term $Z^T\nabla^2 fZ$ in (6) is positive definite. This term is due to a simplification where the Hessian of the Lagrange function $H\mathcal{L}$ is replaced by $\nabla^2 f$, which is known as the Gauss-Newton assumption. This simplification can deteriorate to linear convergence if the second derivative contributions from the model are significant [18].

2.1.2 *Reduced Hessian approach (rOPT)*

This is a null-space method where a decomposition is applied to the KKT conditions to eliminate variables. Consider the SQP subproblem which at an iterate k generates a search direction $d_k \in \mathbb{R}^{(n_x P + n_u M)}$ by solving

$$\begin{aligned} \min\nolimits_{d_k} \ \ & g_k^T d_k + \tfrac{1}{2} d_k^T W\left(x_{k,} u_k\right) d_k \\ s.t. \ \ & c_{\mathcal{E}}(x,u;k) + A(x,u;k)^T d_k = 0 \\ & x^k + d_k^x \in \mathcal{X} \\ & u^k + d_k^u \in \mathcal{U} \end{aligned} \tag{7}$$

Here $\nabla f_k = g_{k,} W_k$ is given by $H\mathcal{L}_k$, the Hessian of the Lagrangian function, or an approximation. $A(x,u;k)$ denotes the constraint Jacobian. The next iterate is computed as $(x_{k+1}, u_{k+1}) = (x_k, u_k) + \alpha_k(d_k^x, d_k^u)$ where α_k is a step length parameter chosen to reduce a suitable merit function. We now partition $z = (x,u) \in \mathbb{R}^{(n_x P + n_u M)}$ into state and control variables through the basis given by a matrix $[Y_k \ \ Z_k]$. This allows the representation of the search vector as $d_k = Y_k p_{Y,k} + Z_k p_{Z,k}$. Assume that Z_k is a basis for $\mathcal{N}(A_k^T)$, i.e. $A_k^T Z_k = 0$, and that Y_k is chosen so that the matrix $[Y_k \ \ Z_k]$ is nonsingular. Hence, we decompose d_k into two components. The model constraint from problem (7) can now be rewritten as $c_{\mathcal{E},k} + A_k^T Y_k p_{Y,k} = 0$. Since $[Y_k \ \ Z_k]$ is nonsingular, assuming full column rank of A_k leads to $p_{Y,k} = -\left(A_k^T Y_k\right)^{-1} c_{\mathcal{E},k}$ which gives us $d_k = -Y_k\left(A_k^T Y_k\right)^{-1} c_{\mathcal{E},k} + Z_k p_{Z,k}$. We arrive at the reduced size qp subproblem in $p_{Z,k} \in \mathbb{R}^{n_u M}$ (considering $p_{Y,k}$ as a constant)

$$\begin{aligned} \min_{p_{Z,k}} \ & (Z_k^T g_k + w_k^T) p_{Z,k} + \\ & \frac{1}{2} p_{Z,k}^T Z_k^T W_k Z_k p_{Z,k} \\ \text{s.t. } & z_k + Y_k p_{Y,k} + Z_k p_{Z,k} \in \mathcal{X} \times \mathcal{U} \end{aligned} \tag{8}$$

where $w_k = Z_k^T W_k Y_k p_{Y,k}$. The choice of Y_k and Z_k is motivated by the partitioning into dependent and free variables. [25] argue that the partitioning $A_k^T = [C_k \quad N_k]$ with

$$Z_k = \begin{bmatrix} -C_k^{-1} N_k \\ I \end{bmatrix} \quad Y_k = \begin{bmatrix} I \\ 0 \end{bmatrix} \tag{9}$$

should be utilized, assuming a non-singular C_k and $A_k^T Z_k = 0$. If we choose the natural partitioning $[C_k \quad N_k] = \left[\left(\frac{\partial c_{\mathcal{E},k}}{\partial x}\right)^T \; \left(\frac{\partial c_{\mathcal{E},k}}{\partial u}\right)^T\right]$ arising from linearization with respect to (x, u), this leads to the following relations

$$Z_k = \begin{bmatrix} -C_k^{-1} N_k \\ I \end{bmatrix} = \begin{bmatrix} S_k \\ I \end{bmatrix}$$
$$d_k^x = p_{Y,k} + S_k p_{Z,k}$$
$$d_k^u = p_{Z,k}$$

Summarizing, the search direction in rOPT has an added component $p_{Y,k}$ which is not present in the sOPT method. Inserting $p_{Y,k} = 0$ into (8) and comparing with (6), we observe that the search directions for the sOPT approach coincide with the rOPT approach with $[C_k \; N_k]$ selected from the linearization. The step $p_{Z,k}$ is in the null space of A_k^T, i.e. it is tangential to the equality constraints, while $p_{Y,k}$ is in the range space of A_k. Then, $p_{Z,k}$ aims at reducing the objective while $p_{Y,k}$ searches for feasibility. Since the sOPT method is a feasible path method, $p_{Y,k} = 0$.

In comparing the two approaches we observe that the sequential approach maintains feasibility of all iterates, while rOPT searches for feasibility and optimality simultaneously. In addition the sequential method solves the model at each iteration, while rOPT only solves the model once, at the solution z^*. Recall that we neglected the cross-term involving second order model derivatives in equation (6), because $p_Y = 0$.

We also note that the sequential approach only handles initial value problems (IVP), i.e. end-state constraints like $x^P \in \Omega_x$ cannot be guaranteed since it implements a shooting strategy in evolving the model over the horizon. The endpoint constraint changes the problem into a boundary value problem (BVP) which must be handled by simultaneous strategies. Moreover the sequential approach can experience convergence problems on unstable systems. (See [23] and [7] for details.)

Finally, we observe that the reduction of the size of the qp to be solved, introduces an additional cost of calculating S_k (sOPT) and Z_k (rOPT) respectively. Here we implement decomposition strategies for sparse matrices from the Harwell subroutine libraries MA28/48. Note that $\nabla_x c_{\mathcal{E},k}$ and C_k are sparse block lower triangular matrices of order $n_x P$. Hence, solving for S_k (sOPT) and Z_k (rOPT) is cheap, of order $O\left((n_x P)^p\right)$, with $p \in [1, 2]$ instead of $p = 3$ due to sparsity. Recall that the efficiency of reduced gradient algorithms relies on the assumption that $n_x P - n_u M \gg n_u M$ and that $n_u M$ is small. Alternatives to computing the Jacobian by analytic partial derivatives are finite differences and automatic differentiation [26]. This will have a significant impact on the computational demands, as shown in section 3.2.

We now assume the presence of *nonlinear* inequality constraints in the `NLP` (2). In this section we propose an extension to the `rOPT` algorithm that maintains feasibility with respect to nonlinear inequality constraints. This feature has a number of advantages. Ensuring a feasible inequality path can lead to more reliable control performance with nonlinear constraints, especially if these are hard constraints. Also, maintaining this feasible path could ensure better conditioning of the nonlinear process model, particularly if these constraints enforce stability or other desirable behavior. Moreover, enforcing a feasible path for the inequality constraint also prevents the algorithm from taking poor steps due to poor linearizations of the model. This is the motivation for trust region methods as well (see [13]). Finally, with this approach, the penalty merit function is simplified as no weight need be selected for the inequality constraints. Otherwise, on ill-conditioned problems, these weights can become quite large in `rOPT`.

The proposed algorithm, coined `rFOPT`, is expected to be efficient for a moderate number of inequality constraints (e.g. that $m \gg p > 0$), due to the active set approach. If there are many inequality constraints, interior point methods should be considered. An alternative to the proposed algorithm is to introduce slack variables for the nonlinear inequalities, and to solve the resulting `NLP` (with nonlinear equality constraints and bounds). We associate two sets with the `NLP` (2):

$$\mathcal{Z} = \{z \in \mathbb{R}^n \mid c_{\mathcal{E}}(z) = 0,\ c_{\mathcal{I}}(z) \leq 0\}$$
$$\mathcal{Z}_I = \{z \in \mathbb{R}^n \mid c_{\mathcal{I}}(z) \leq 0\}$$

with $\mathcal{Z} \subseteq \mathcal{Z}_I$. A point $z \in \mathcal{Z}_I$ satisfies the inequality constraints but not necessarily the equality constraints. Points in $\mathcal{Z}$ are feasible, while points in $\mathcal{Z}_I$ are (at least) feasible with respect to inequalities. We will generate iterates $z_k \in \mathcal{Z}_I$ and a solution $z^* \in \mathcal{Z}$. Assume a known initial point $z_0 \in \mathcal{Z}_I$. The `qp` (3) produces the ordinary `SQP` direction d_k^0. Superscript 0 refers to ordinary `SQP` directions throughout this chapter.

A feasible direction for $z_k \in \mathcal{Z}_I$ is a direction $d_k \in \mathbb{R}^n$ such that there exists a scalar $\bar{\alpha} > 0$ satisfying

$$(z_k + \alpha d_k) \in \mathcal{Z}_I \ \ \forall \alpha \in [0, \bar{\alpha}] \tag{10}$$

Since d_k^0 lies in the null space of $G(z_k)^T$, it will not satisfy (10) in general. The `rFOPT` algorithm tilts the ordinary `SQP` direction into the feasible region and produces a direction d_k. The range-space step is given by

$$p_{Y,k} = -(A_k^T Y_k)^{-1} c_{\mathcal{E},k} = -C_k^{-1} c_{\mathcal{E},k} \tag{11}$$

In (11) the partitioning $A_k^T = [C_k \quad N_k]$ from Section 2.1.2 is used together with the definition of Y_k. `rFOPT` implements feasibility by modifying the subproblem (8) into:

$$\begin{aligned}\min_{p_{Z,k},\gamma_k} \quad & \tfrac{1}{2}p_{Z,k}^T B_k p_{Z,k} + \gamma_k \\ s.t. \quad & \left(Z_k^T g_k + \zeta_k w_k\right)^T p_{Z,k} \leq \gamma_k \\ & L - z_k - Y_k p_{Y,k} \leq Z_k p_{Z,k} \\ & \qquad \leq U - z_k - Y_k p_{Y,k} \\ & \left(Z_k^T \nabla c_{\mathcal{I},k}^T\right)^T p_{Z,k} \\ & \qquad \leq \gamma_k \eta_k - \rho(p_{Y,k})\end{aligned} \tag{12}$$

where, using $\hat{c}_{\mathcal{I}}(z_k) = c_{\mathcal{I}}(z_k) + \left(Y_k^T \nabla c_{\mathcal{I},k}^T\right)^T p_{Y,k}$,

$$\rho^i(p_{Y,k}) = \begin{cases} \hat{c}_{\mathcal{I}}^i(z_k) & \text{if } \hat{c}_{\mathcal{I}}^i(z_k) > ftol \\ ftol & \text{otherwise} \end{cases} \tag{13}$$

In (12), ζ_k is a scalar damping parameter and $w_k = T_k Y_k p_{Y,k}$, where $T_k \approx Z_k^T W_k$. T_k may be computed by Broyden's method. L and U are lower and upper bounds on z, and *ftol* is the feasibility tolerance for the nonlinear inequalities. The update rule for ζ_k is given by [22]. The reduced Hessian approximation B_k is updated by a `BFGS` scheme. Establishing that the scalar $\gamma_k \leq 0$, guarantees that

$$d_k = Y_k p_{Y,k} + Z_k p_{Z,k} \tag{14}$$

satisfies (10).

The vector parameter η_k controls the amount of tilting. There is one η_k^i for each nonlinear inequality. To prove superlinear convergence, it suffices to show that $d_k \to d_k^0$ as $k \to \infty$. This is accomplished by letting $\eta_k \to 0$ as the solution z^* is approached. Since z^* is unknown, updating η_k is done by computing d_k^0, and comparing this to d_k. The updating of η_k is revisited below.

The next iterate is computed as

$$z_{k+1} = z_k + \alpha_k d_k + \alpha_k^2 d_k^C \tag{15}$$

where α_k is a step-length parameter chosen to reduce the value of the constrained l_1-merit function

$$\begin{aligned}\phi_\nu(z_k) &= f(z_k) + \nu_k \left\|c_{\mathcal{E}}(z_k)\right\|_1 \\ s.t. \;\; & c_{\mathcal{I}}(z_k) \leq 0\end{aligned} \tag{16}$$

with penalty parameter ν_k updated by

$$\nu_k = \begin{cases} \nu_{k-1} & \text{if } \nu_{k-1} \geq \|\lambda_k\|_\infty + 2\rho \\ \|\lambda_k\|_\infty + 3\rho & \text{otherwise} \end{cases} \tag{17}$$

The correction term d_k^C in (15) is needed to guarantee that the discontinuity introduced by the constraint in the line search, will not disrupt superlinear convergence of the algorithm. d_k^C is decomposed into

$$d_k^C = Y_k p_{Y,k}^C + Z_k p_{Z,k}^C \tag{18}$$

The range space correction step is given by

$$p_{Y,k}^C = -C_k^{-1} c_{\mathcal{E}}(z_k + d_k) \tag{19}$$

Define

$$\begin{aligned} \hat{c}_{\mathcal{I},k}^C \quad &= c_{\mathcal{I}}(z_k + d_k) + \|d_k\|^\kappa \\ &\quad + \left(Y_k^T \nabla c_{\mathcal{I},k}^T\right)^T (p_{Y,k} + p_{Y,k}^C) \end{aligned}$$

($\kappa \in (2,3)$ is a user selected parameter) and

$$\rho^{C,i}(p_{Y,k}^C) = \begin{cases} \hat{c}_{\mathcal{I},k}^{C,i} & \text{if } \hat{c}_{\mathcal{I},k}^{C,i} > ftol \\ ftol & \text{otherwise} \end{cases}$$

The null space correction step is solved from the subproblem

$$\begin{aligned} \min_{p_{Z,k}^C} \ & \tfrac{1}{2}(p_{Z,k}^C)^T B_k p_{Z,k}^C + (Z_k^T g_k \\ & \qquad + \zeta_k(w_k + w_k^C) + B_k p_{Z,k})^T p_{Z,k}^C \\ \\ s.t. \ & L - (z_k + d_k) - Y_k(p_{Y,k} + p_{Y,k}^C) \\ & \qquad \leq Z_k(p_{Z,k} + p_{Z,k}^C) \\ & \qquad \leq U - (z_k + d_k) - Y_k(p_{Y,k} + p_{Y,k}^C) \\ & \left(Z_k^T \nabla c_{\mathcal{I},k}^T\right)^T (p_{Z,k} + p_{Z,k}^C) \\ & \qquad \leq -\rho^C(p_{Y,k}^C) \end{aligned} \tag{20}$$

where $w_k^C = T_k Y_k p_{Y,k}^C$.

Finally, the tilting parameter η is updated by computing the ordinary SQP direction. The range space tilting step is given by $p_{Y,k}^E = -C_k^{-1} c_{\mathcal{E}}(z_k) = p_{Y,k}$. The null space tilting step becomes

$$\begin{aligned} \min_{p_{Z,k}^E} \ & \tfrac{1}{2}(p_{Z,k}^E)^T B_k p_{Z,k}^E \\ & \qquad + \left(Z_k^T g_k + \zeta_k w_k\right)^T p_{Z,k}^E \\ \\ s.t. \ & L - z_k - Y_k p_{Y,k} \le Z_k p_{Z,k}^E \\ & \qquad \le U - z_k - Y_k p_{Y,k} \\ & \left(Z_k^T \nabla c_{\mathcal{I},k}^T\right)^T p_{Z,k}^E \\ & \qquad \le -\rho(p_{Y,k}) \end{aligned} \tag{21}$$

The update rule for η_k, based on

$$d_k^E = Y_k p_{Y,k} + Z_k p_{Z,k}^E \tag{22}$$

is given in Section 2.2.1. The Lagrange multipliers for the equality constraints are estimated from

$$\lambda_{\mathcal{E},k} = -(Y_k^T A_k)^{-1} Y_k^T g_k = -C_k^{-T} Y_k^T g_k \tag{23}$$

The multipliers for the inequalities are derived from the qp multipliers.

Hence, rFOPT solves the 3 subproblems (12), (20) and (21) at each iteration, one for each of $p_{Z,k}$, $p_{Z,k}^E$, $p_{Z,k}^C$. These subproblems consists of solving a small sized qp (due to the assumption $n_x P - n_u M \gg n_u M$), and two solves for $p_{Y,k}$ and $p_{Y,k}^C$ (with differing right-hand sides, but the same LU factorization). The price to pay for having $z_k \in \mathcal{Z}_I$ is an increased workload of the algorithm. This is shown in Table 4 in Section 3.

2.2.1 The rFOPT algorithm

(1) Choose constants $\theta \in (0, \frac{1}{2})$, $\kappa \in (2,3)$, $\epsilon_l > 0$, $0 < \underline{C}^\eta < \bar{C}^\eta$, $\bar{D} > 0$, $0 < \tau < \tau' < 1$, $\eta_0 > 0$ for (12) and $C_0^\eta \in \left[\underline{C}^\eta, \bar{C}^\eta\right]$. Set $k = 0$ and select a starting point $z_0 \in \mathcal{Z}_I$. Choose the initial penalty parameter ν_o, a $(n-m)\times(n-m)$ symmetric and positive definite matrix B_0 and a $(n-m)\times n$ starting matrix T_0 for the Broyden approximation.
(2) Evaluate f_0, g_0, c_0 and A_0. Compute Y_0 and Z_0 from (9).
(3) Solve for the range space step $p_{Y,0}$ from (11). Compute the approximation w_0 by Broyden's method.
(4) Compute the damping parameter $\zeta_0 \in (0,1]$ and compute the tilted null-space step $p_{Z,0}$ and γ_0 from (12).

```
(5)  MAIN LOOP: Define the tilted SQP direction by $d_k = Y_k p_{Y,k} + Z_k p_{Z,k}$ from (14). If $d_k = 0$ STOP.
(6)  Compute the Maratos correction term $d_k^C$ from (19,20,18) if it exists and satisfies $\left\| d_k^C \right\| \leq \|d_k\|$. Otherwise set $d_k^C = 0$. Set $\alpha_k = 1$.
(7)  Arc search. Test the constrained Armijo condition
```

$$
\begin{aligned}
&\phi_{\nu_k}(z_k + \alpha_k d_k + \alpha_k^2 d_k^C) \\
&\qquad \leq \phi_{\nu_k}(z_k) + \theta \alpha_k D\phi_{\nu_k}(z_k, d_k) \\
s.t. \quad & c_{\mathcal{I}}^i(z_k + \alpha_k d_k + \alpha_k^2 d_k^C) \leq 0
\end{aligned}
\tag{24}
$$

```
(8)  If (24) is not satisfied, choose a new $\alpha_k \in [\tau\alpha_k, \tau'\alpha_k]$ and go to step (7), otherwise set $z_{k+1} = z_k + \alpha_k d_k + \alpha_k^2 d_k^C$.
(9)  Evaluate $f_{k+1}$, $g_{k+1}$, $c_{k+1}$ and $A_{k+1}$. Compute $Y_{k+1}$ and $Z_{k+1}$ from (9).
(10) Compute the Lagrange multiplier estimates from (23). Update the weight $\nu_{k+1}$ of the merit function from (17).
(11) Update $T_{k+1}$ and $B_{k+1}$.
(12) Compute $p_{Y,k+1}$ from (11). Compute the approximation $w_{k+1} = T_{k+1} Y_{k+1} p_{Y,k+1}$, and $\zeta_{k+1} \in (0,1]$. Solve for $p_{Z,k}^E$ from (21).
(13) Select $C_{k+1}^\eta \in \left[ \underline{C}^\eta, \bar{C}^\eta \right]$.
     If $(\|d_k\| < \epsilon_l)$ then, compute $d_{k+1}^E$ from (22). Then
       if $\left\| d_{k+1}^E \right\| \leq \bar{D}$ then set $\eta_{k+1} \leftarrow C_{k+1}^\eta \left\| d_{k+1}^E \right\|^2$
       else set $\eta_{k+1} \leftarrow C_{k+1}^\eta \|d_k\|^2$
     else set $\eta_{k+1} \leftarrow C_{k+1}^\eta \epsilon_l^2$
(14) Solve for $p_{Z,k+1}$ and $\gamma_{k+1}$ from (12) with $w_{k+1}$ and $\zeta_{k+1}$ computed in step (12).
(15) Set $k \leftarrow k + 1$ and go to step (5)
```

A variant of the rFOPT algorithm has been shown to be 1-step superlinearly convergent under specified assumptions [24].

More details on item (13) are given in [11]. The main difference between rFOPT and the RFSQP algorithm [11] lies in the treatment of nonlinear equality constraints. RFSQP handles equality constraints by restating them as inequalities, which are forced to be asymptotically satisfied as equalities during the course of the optimization algorithm. rFOPT treats equalities as in rOPT, which may be more efficient for large-scale systems. rFOPT is implemented with sparse linear algebra in Matlab.

3 Simulations

NMPC was implemented on a simple case with four different optimization methods. The first is a basic full space SQP method. The second is the E04UCF solver from the NAG toolbox for Matlab (this a more robust version of basic SQP). The third is the reduced Hessian method rOPT, and the fourth is the sequential method sOPT. In the presence of nonlinear inequalities rOPT uses slack variables, while rFOPT from section 2.2 always uses the feasibility mechanism. The case is a CSTR with multiple steady states. The CSTR example was explored by application of various discretization methods and approximations of the Jacobian. Both sOPT and rOPT methods were implemented with Gauss-Newton options with analytic derivatives from the model, as well as with BFGS updates.

3.1 Implementation issues

The basic SQP full-space method and rOPT were implemented with a l_1-penalty function. sOPT implemented an l_1-penalty function without penalization of equality constraints, since sOPT always remain feasible with respect to equalities. The line search for all methods is backtracking line search.

The relaxed convergence criteria from [25], section 8.2.3, were implemented with tolerance 10^{-5} for the basic SQP method and sOPT. In rOPT the algorithm stops whenever a certain KKT measure is decreased below the tolerance 10^{-5}. The implementation of rOPT is generally more carefully performed than the basic SQP and sOPT methods. Hence, the relaxed termination criteria used in basic SQP and sOPT partly compensates for a rudimentary implementation. However, as the discussion in section 2 indicates, the sOPT method may show linear convergence in certain circumstances, and relaxed termination criteria can therefore be of crucial importance in production codes as well.

For the CSTR case the model was discretized with explicit and implicit Euler, Lobatto IIIC and ordinary 4th order Runge-Kutta. The Jacobian matrices associated with each discretization method has a specific structure (almost block-diagonal (ABD)) giving specific sparsity patterns. The CSTR case was implemented with both analytic Jacobian, automatic differentiation (AD) [26] and finite difference approximations of the Jacobian. The CSTR case was investigated with different sampling rates and prediction and move horizons.

The results are reported with the last trajectory as initial value for the optimization algorithms. However, for sOPT, the initial value had to be set to the original steady state value to avoid convergence problems. rOPT, rFOPT and E04UCF were insensitive to the initial value.

It is assumed that a disturbance should not produce a system without feasible solutions. A disturbance will affect the equality constraints (through the feedback), but not the initial guess in the present implementation. Note that the algorithms discussed in this paper can handle infeasible systems, if suitable precautions are taken. For example the algorithms can detect an infeasible optimization problem, and then it is up to the `NMPC`-implementor to take precautions (infeasibility recovery by constraint relaxation). Note that infeasibility may occur if the inequality constraints or variable bounds are set too tight. Infeasibility recovery is not considered in this paper.

3.2 Case: CSTR

The case is the following isothermal `CSTR` with multiple steady states from [27] also investigated by [23]

$$\begin{aligned} \tfrac{dx_1}{dt} &= u_1 + u_2 - k_1\sqrt{x_1} \\ \tfrac{dx_2}{dt} &= (C_{B_1} - x_2)\tfrac{u_1}{x_1} + (C_{B_2} - x_2)\tfrac{u_2}{x_1} \\ &\quad - \tfrac{k_2 x_2}{(1+x_2)^2} \end{aligned} \tag{25}$$

with parameter values $k_1 = 0.2$, $k_2 = 1$, $C_{B_1} = 24.9$ and $C_{B_2} = 0.1$. For $(u_1, u_2) = (1, 1)$ the `CSTR` has three equilibrium points at $x_1 = 100,\ x_2 \in (0.633, 2.72, 7.07)$, with the middle equilibrium point being unstable, and the others stable. The system (25) was discretized with a time step h, prediction horizon P, move horizon M and simulated for N_{MPC} samples, i.e. the `NMPC` problem is repeatedly solved N_{MPC} times. At time step 10 the process experiences a +50% step in C_{B_1} which is seen by the `NMPC` algorithm through the feedback only. The weights in problem (1) are $Q = 10I_{n_x}$ and $R = I_{n_u}$ and deviation from stationary values is penalized. Here the control objective is to keep the states and controls at their initial values $x_{eq} = (100, 2.72)$ and $u_{eq} = (1, 1)$, the unstable point.

The physical bounds $(x^l, u^l) \geq 0$ are imposed over the horizons. Note that with the given initial conditions and the given disturbance the active set is empty and unaltered throughout the prediction horizon P in this case. The `SQP`-algorithms were initialized with the output from the previous call for each `NMPC` iteration. Note that integral action is not implemented. This is justified because only a comparison of the optimization methods is investigated, and it is expected that introducing integral action will not influence this comparison. The `NMPC` problem was solved either with an end-point constraint $x^P = x_{eq}$, or with inequality constraints as in the next subsection. However, the reported results for `sOPT` were not produced with $x^P = x_{eq}$, since adding this constraint produced failure of the `sOPT` algorithm. Both Gauss-Newton and `BFGS` approximations were considered for all algorithms.

3.2.1 *Nonlinear inequality*

To test the algorithms a nonlinear inequality constraint was added:

$$(k_1\sqrt{x_1})x_2 \leq U_I \tag{26}$$

where $U_I = 5.65$ is selected so that the inequality is inactive at the set-point, but becomes active when the disturbance enters. This inequality can be considered as an economic or environmental constraint, limiting the flow-rate of a component. A representative simulation result with and without the inequality constraint is shown in Figure 1. The process was simulated by `Matlab`'s `ode45` in all cases. If the control inputs are kept at the equilibrium input $[1, 1]$, the system will settle at a new equilibrium point at $x = (100, 15.94)$. The ripple appearing with the inequality constraint is due to the fact that the optimizer only sees the disturbance through the feedback.

3.3 *Results*

The computations were implemented in `Matlab` with some routines available as `mex/dll`-files on a Dell Latitude C800 / Pentium III / 1GHz / 512Mb RAM running Windows 2000. Computational results are shown in Tables 2-4, where $h/P/M$ are the sampling time in seconds, prediction and input horizons. The # vars. tot/dep/free are the total, dependent and free number of variables.

The ratios free/tot for the two Euler methods, Lobatto and Runge-Kutta methods are 0.45, 0.29, 0.14 respectively. The inequality constraint (26) was applied at all stages in the prediction horizon P. The number of inequality constraints is equal to P for all discretization methods. That is, the sublevel variables in the Lobatto and Runge-Kutta methods are not constrained in this case study. Note that there are no active bound constraints in any of the cases studied.

Note that the controls are not identical for the case with and without inequality constraints (this does not show in Figure 1). The numerical values are given in Table 1. In Table 1, $u(i)$ are the controls, $x(i)$ are the predicted states and $x_p(i)$ are the measured states. It is verified that the predicted state values for the case without inequality constraints violate (26). Note that the *measured* states also violate (26) for the case with inequalities.

The Jacobian is either analytic, found by automatic differentiation (`AD`[2]) or approximated by finite differences. Finite differences were considered for both the full Jacobian (a dense matrix), denoted fd2, and the elements along the block diagonal (a sparse matrix), denoted fd1. In `sOPT` fd2 was implemented by finite difference

[2] Using the Tapenade tool available at: `http://tapenade.inria.fr:8080/`

perturbations of the simulator. That is, the sensitivity matrix $\mathcal{S}$ was approximated directly. CPU time is the time measured by `Matlab`'s `cputime` command from start to end of the main `NMPC` loop. The results are listed in Tables 2 (basic `SQP` and `E04UCF` from the `NAG` toolbox), 3 (`sOPT`) and 4 (`rOPT` and `rFOPT`). In Table 4, `rOPT` and `rFOPT` are identical when there are no inequalities (column 5 and 6). Column 7 shows results using `rOPT` with slacks, while column 8 shows results using `rFOPT`. Items marked with 'fail' denote failures, and items marked '-' were not attempted. The objective function has the shape of a narrow valley with a flat bottom, and the solutions are essentially the same.

For basic `SQP`, `E04UCF`, `rOPT` and `rFOPT`, the inequality constrained problems were solved with `BFGS` approximations. For `sOPT`, the inequality constrained problems were solved with the Gauss-Newton (listed as G-N in Tables 2-4) method.

Figure 2 highlights some of the tabulated results for the case without inequalities. In this figure the log_2 ratio, e.g., $\log_2(\frac{CPU_1}{CPU_2})$ of CPU times are sorted and compared. Whenever $\log_2 < 0$, we have $CPU_2 > CPU_1$ and vice versa. The left half of figure 2 shows that there is a tradeoff between basic `SQP` and reduced Hessian `SQP` when the number of variables is small (due to additional overhead in `rOPT`). The right half of Figure 2 shows that if explicit discretization schemes are applied, evaluation of the model constraint is cheap, and `sOPT` benefits from this.

The results in Tables 2-4 are summarized in the following conclusions:

- `sOPT` is sensitive to the choice between implicit and explicit integration methods, while basic `SQP`, `E04UCF`, `rOPT` and `rFOPT` are insensitive to this.
- Finite difference approximations of the full Jacobian in basic `SQP` , `E04UCF`, `rOPT` and `rFOPT` should be avoided.
- Reduced gradient methods perform better for a large number of variables with few degrees of freedom.
- All solvers benefit from analytic derivatives, and automatic differentiation [26] is a cheap way of achieving this.

We observe that `rOPT` and `rFOPT` does not work well with the Gauss-Newton method, and we attribute this to the projection $Z^T BZ$, which can be inaccurate. Table 4 shows that requiring feasibility of inequalities triples the computational load. This is as expected, since `rFOPT` solves three subproblems at each iteration. Hence, the tabulated results do not indicate any benefits from maintaining feasibility with `rFOPT` on this case. The results for `sOPT` were obtained by termination prior to convergence. In this case, this produced satisfactory results, with small violations of the inequality constraints. The importance and magnitude of such deviations is case dependent.

Note that a consequence of the first item in the list above is that simultaneous `SQP` methods can have fewer variables when implemented with implicit discretization methods. Here the step length h can be increased beyond the stability limit of explicit methods (but not beyond reasonable accuracy). Since the basic SQP and sOPT methods are similar in implementation complexity, sOPT should be chosen when explicit discretization schemes suffice. Attempts to use E04UCF in sequential mode (by simulating the model inside the call to the objective function) were not successful, since E04UCF never converged even for very loose tolerance settings.

In the face of more challenging processes, we anticipate that reduced Hessian methods are better provided that the assumption of few degrees of freedom continues to hold. This assumption commonly holds in NMPC. An example of rOPT applied to a problem with an increase in the number of discretization parameters for the states is given in [16], where hundreds of states were used.

4 Conclusions

In this paper different nonlinear MPC strategies were implemented on a CSTR case study and the computational load and quality of the results were investigated. From Tables 2-4 it is observed that among the different NMPC methods, the rOPT method is preferable in view of computational time if implicit discretization methods are required. In NMPC computational time is limited, and generally, SQP involves an adaptive subproblem, i.e. its computational time is not deterministic. Consequently, in NMPC feasible path SQP methods are preferred since they allow termination prior to convergence [28]. Such methods must solve the model constraints at each SQP iteration, which may be time consuming if the model is represented with an implicit discretization scheme. Hence, sOPT becomes computationally demanding if implicit discretization methods are applied, whereas simultaneous SQP methods perform equally well regardless of whether implicit or explicit discretization scheme are applied. Both methods require that the selected discretization scheme is appropriate, i.e. if the model cannot be simulated with a given method, it cannot be expected that the optimization algorithms perform well either.

Hence, nominal stability for unstable nonlinear systems can pose a challenge, where the need for implicit discretization schemes and feasibility of equality constraints at intermediate iterations can be hard to combine.

Feasibility with respect to inequality constraints is easier to achieve. RFSQP [11] and the present algorithm rFOPT maintain feasibility with respect to inequality constraints, and *asymptotic* feasibility with respect to nonlinear equality constraints by combining these with an exact penalty function and an arc search. The feasible path sOPT method with an implicit discretization is expected not to perform as well in the presence of strong nonlinearities [1]. This is contradictory to the needs for NMPC; e.g. problems with strong nonlinearities and trajectory tracking [29].

Finally, the practical considerations discussed in this paper explore the choices an engineer must take if (s)he wants to implement NMPC on a given process. The interplay between discretization methods and optimization algorithms has been investigated through a case study. For explicit discretization methods, sequential methods are easier to implement than simultaneous methods. This applies in particular to reduced Hessian methods which may be quite sophisticated. On the other hand, if implicit discretization methods must be applied, the performance of sOPT deteriorates while simultaneous SQP does not degrade.

Acknowledgment

This paper was financially supported by the Research Council of Norway (project no. 128726/420 - MODTEQ).

References

[1] U. M. Ascher, R. M. M. Mattheij, R. D. Russell, Numerical solution of boundary value problems for ordinary differential equations, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995, classics in Applied Mathematics.

[2] A. Barclay, P. E. Gill, J. B. Rosen, SQP methods and their application to numerical optimal control, in: Variational calculus, optimal control and applications (Trassenheide, 1996), Birkhäuser, Basel, 1998, pp. 207–222.

[3] J. T. Betts, Practical methods for optimal control using nonlinear programming, Advances in Design and Control, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001.

[4] K. Holmström, A. Göran, M. M. Edvall, User's guide for TOMLAB 4.2, Tech. rep., Tomlab Optimization, Sweden (2004).

[5] V. Vassiliadis, Computational solution of dynamic optimization problems with general differential-algebraic constraints, Ph.D. thesis, University of London, U.K. (1993).

[6] H. Chen, F. Allgöwer, A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability, Automatica J. IFAC 34 (10) (1998) 1205–1217.

[7] H. Chen, F. Allgöwer, A computationally attractive nonlinear model predictive control scheme with guaranteed stability for stable systems, J. Proc. Cont. 8 (5-6) (1998) 475–485.

[8] D. Q. Mayne, J. B. Rawlings, C. V. Rao, P. Scokaert, Constrained model predictive control: Stability and optimality, Automatica J. IFAC 36 (6) (2000) 789–814.

[9] W. C. Li, L. T. Biegler, C. G. Economou, M. Morari, A constrained pseudo-Newton control strategy for nonlinear systems, Computers Chem. Engng. 14 (4/5) (1990) 451–468.

[10] N. M. C. de Oliveira, L. T. Biegler, An extension of Newton-type algorithms for nonlinear process control, Automatica J. IFAC 31 (2) (1995) 281–286.

[11] C. T. Lawrence, A. L. Tits, A computationally effi cient feasible sequential quadratic programming algorithm, SIAM J. Optim. 11 (4) (2001) 1092–1118 (electronic).

[12] B. A. Finlayson, Nonlinear analysis in chemical engineering, Chemical engineering, McGraw-Hill, New York, NY, 1980.

[13] M. J. Tenny, S. J. Wright, J. B. Rawlings, Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming, Comp. Optim. Appl.Submitted.

[14] H. G. Bock, M. M. Diehl, J. P. Schlöder, F. Allgöwer, R. Findeisen, Z. Nagy, Real-time optimization and nonlinear predictive control of processes governed by differential-algebraic equations, in: L. T. Biegler, A. Brambilla, C. Scali (Eds.), Prepints: International Symposium on Advanced Control of Chemical Processes (ADCHEM 2000), Pisa, Italy, 2000, 2000, pp. 695–703.

[15] A. M. Cervantes, L. T. Biegler, A stable elemental decomposition for dynamic process optimization, J. Comput. Appl. Math. 120 (1-2) (2000) 41–57, sQP-based direct discretization methods for practical optimal control problems.

[16] L. T. Biegler, A. M. Cervantes, A. Wächter, Advances in simultaneous strategies for dynamic process optimization, Chem. Eng. Sci. 57 (4) (2002) 575–593.

[17] R. Mahedevan, S. Agrawal, F. J. Doyle, Differential flatness based nonlinear predictive control of fed-batch bioreactors, Control Engn. Practice 9 (8) (2001) 889–899.

[18] L. T. Biegler, Effi cient solution of dynamic optimization and NMPC problems, in: Nonlinear model predictive control (Ascona, 1998), Birkhäuser, Basel, 2000, pp. 219–243.

[19] P. T. Boggs, J. W. Tolle, Sequential quadratic programming, in: Acta numerica, Cambridge Univ. Press, Cambridge, 1995, pp. 1–51.

[20] C. V. Rao, S. J. Wright, J. B. Rawlings, Application of interior-point methods to model predictive control, J. Optim. Theory Appl. 99 (3) (1998) 723–757.

[21] J. Nocedal, S. J. Wright, Numerical optimization, Springer-Verlag, New York, 1999.

[22] L. T. Biegler, J. Nocedal, C. Schmid, A reduced Hessian method for large-scale constrained optimization, SIAM J. Optim. 5 (2) (1995) 314–347.

[23] N. M. C. de Oliveira, Newton-type algorithms for nonlinear constrained chemical process control, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA. (1994).

[24] F. Martinsen, The optimization algorithm rFSQP with application to nonlinear model predictive control of grate sintering, Ph.D. thesis, Norwegian University of Science and Technology, Norway (2001).

[25] P. E. Gill, W. Murray, M. H. Wright, Practical optimization, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1981.

[26] A. Griewank, Evaluating derivatives, Vol. 19 of Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000, principles and techniques of algorithmic differentiation.

[27] T. Matsuura, M. Kato, Concentration stability of the isothermal reactor, Chem. Eng. Sci. 22 (1967) 171–184.

[28] D. Q. Mayne, Nonlinear model predictive control : An assessment, in: J. C. Kantor, C. E. Garcia, B. Carnahan (Eds.), CPC-V : Proceedings of the Fifth International Conference on Chemical Process Control, Tahoe City, CA., 1996, AIChE symposium series ; no. 316, CACHE, 1997, pp. 217–231.

[29] S. J. Qin, T. A. Badgwell, An overview of nonlinear model predictive control applications, in: F. Allgöwer, A. Zheng (Eds.), Nonlinear model predictive control (Ascona, 1998), Birkhäuser, 2000, pp. 128–145.

List of Figures

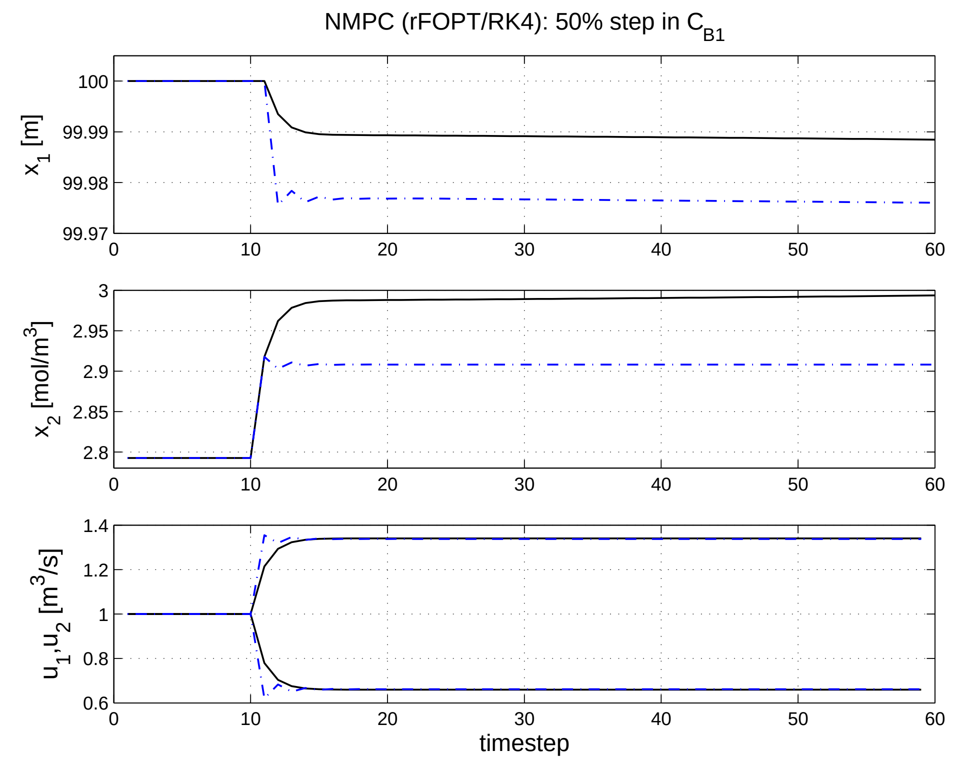

Fig. 1. *NMPC of CSTR*. The figure shows typical results for NMPC of the CSTR case with $h = 1$. The various discretization methods produced nearly identical results. The figure shows results with (solid) and without (dashed) the inequality constraint (26). In the lower subfigure the lower curves shows u_1 and the upper curves shows u_2. The step in C_{B_1} enters at time step 10, and this drives the state x_2 away from the equilibrium.

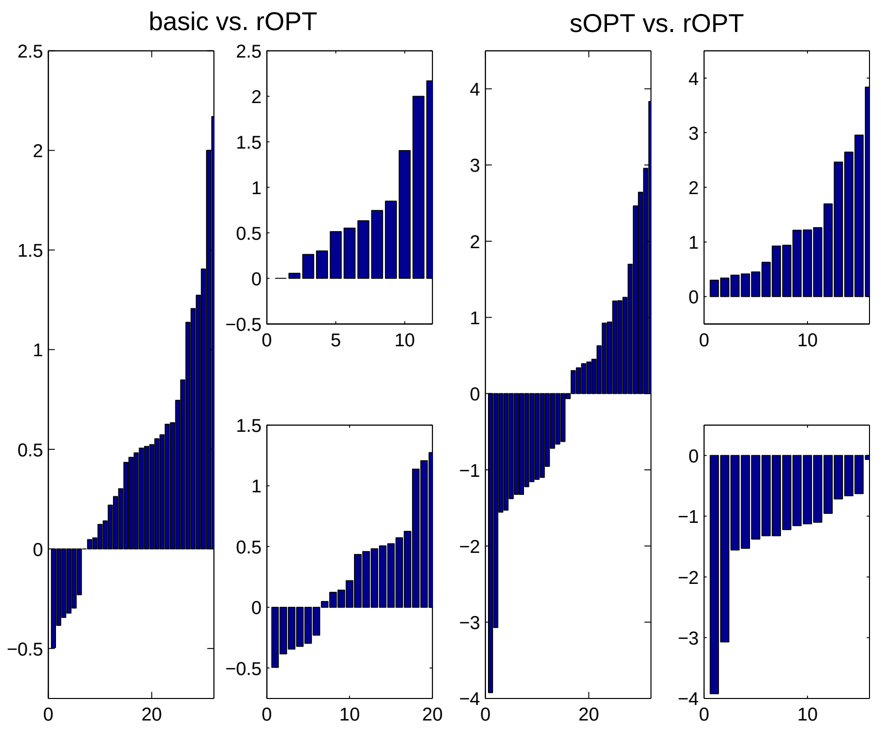

Fig. 2. *Logarithmic comparison of CPU times.* The figure shows log2 plots of the ratios between CPU times. The left half compares basic `SQP` and `rOPT` with `BFGS` updates and no inequalities. To the left is shown the $\log_2(\frac{basicSQP}{rOPT})$ ratios for all rows in the relevant columns in Tables 2 and 4. The two smaller plots shows: *Upper*: Larger cases (Lobatto with $h = 1$ and the two RK4 cases). *Lower*: Smaller cases (Lobatto with $h = 2$, explicit and implicit Euler). The right half compares `sOPT` (Gauss-Newton) with `rOPT` (`BFGS` updates) without inequalities. The left part of the right half shows the $\log_2(\frac{sOPT}{rOPT})$ ratios for all rows in the relevant columns in Tables 3 and 4. The two smaller plots shows: *Upper*: Implicit Euler and Lobatto IIIC cases. *Lower*: Explicit Euler and RK4 cases.

List of Tables

Table 1
State and control values from Figure 1

Variable	With INEQS	Without INEQS
$u(1)$	0.661801	0.659374
$u(2)$	1.337969	1.340521
$x(1)$	99.976941	99.989479
$x(2)$	2.825326	2.905540
$x_p(1)$	99.976941	99.989479
$x_p(2)$	2.907739	2.987641

Table 2
Nonlinear MPC on a CSTR: Basic SQP method and E04UCF.

Discretization method	Horizons $h/P/M$	# vars. tot/dep/free	Jacobian analytic/ fd1/fd2/AD	CPU time (s) basic SQP No ineqs. G-N	BFGS	Ineq. w/BFGS	E04UCF w/BFGS No ineqs.	Ineq.
Explicit Euler	2/6/5	22/12/10	analytic	1.6	2.2	4.4	4.6	4.0
			fd1	2.1	5.4	6.6	8.1	9.0
			fd2	2.1	5.8	-	9.1	-
			AD	1.8	2.8	5.4	6.0	6.8
	1/12/10	44/24/20	analytic	3.1	6.2	17	16	21
			fd1	5.1	22	25	44	51
			fd2	6.7	30	-	68	-
			AD	4.2	8.6	22	31	37
Implicit Euler	2/6/5	22/12/10	analytic	1.5	2.3	4.6	3.4	4.0
			fd1	2.1	5.0	8.3	8.0	9.1
			fd2	2.1	5.5	-	9.0	-
			AD	1.7	2.9	6.0	5.9	6.7
	1/12/10	44/24/20	analytic	3.0	6.1	17	16	20
			fd1	5.3	23	27	42	52
			fd2	6.7	29	-	65	-
			AD	4.0	8.5	22	30	37
Lobatto IIIC	2/6/5	34/24/10	analytic	1.8	2.6	4.8	5.2	6.2
			fd1	2.7	7.1	10	14	15
			fd2	3.3	8.1	-	20	-
			AD	2.2	3.5	6.4	9.6	11
	1/12/10	68/48/20	analytic	4.5	9.0	20	38	39
			fd1	10	27	49	89	89
			fd2	18	45	-	194	-
			AD	6.6	12	25	65	64
RK4	2/6/5	70/60/10	analytic	2.9	3.8	10	14	16
			fd1	4.3	11	19	24	28
			fd2	16	30	-	50	-
			AD	3.9	5.3	14	19	23
	1/12/10	140/120/20	analytic	12	36	67	253	255
			fd1	21	108	106	334	336
			fd2	116	280	-	791	-
			AD	17	45	77	297	299

Table 3
Nonlinear MPC on a CSTR: sOPT method.

Discretization method	Horizons $h/P/M$	# vars. tot/dep/free	Jacobian analytic/fd	CPU time (s) No ineqs. G-N	CPU time (s) No ineqs. BFGS	CPU time (s) Ineq. w/G-N
Explicit Euler	2/6/5	22/12/10	analytic	1.6	1.5	1.7
			fd1	1.4	1.7	1.8
			fd2	1.5	2.1	-
			AD	1.5	1.8	1.8
	1/12/10	44/24/20	analytic	2.8	3.4	3.3
			fd1	3.4	4.0	3.8
			fd2	4.5	6.2	-
			AD	3.5	4.3	4.2
Implicit Euler	2/6/5	22/12/10	analytic	4.1	10	6.4
			fd1	12	10	7.8
			fd2	25	43	-
			AD	4.3	11	6.6
	1/12/10	44/24/20	analytic	13	34	24
			fd1	21	38	24
			fd2	171	258	-
			AD	14	35	24
Lobatto IIIC	2/6/5	34/24/10	analytic	5.1	12	8.2
			fd1	12	12	13
			fd2	32	48	-
			AD	5.3	13	8.2
	1/12/10	68/48/20	analytic	17	44	35
			fd1	20	47	44
			fd2	225	346	-
			AD	19	47	32
RK4	2/6/5	70/60/10	analytic	2.4	2.2	3.2
			fd1	3.0	3.0	3.7
			fd2	2.5	3.2	-
			AD	3.1	3.1	4.5
	1/12/10	140/120/20	analytic	8.6	6.0	5.9
			fd1	11	9.4	7.9
			fd2	11	16	-
			AD	11	9.8	9.3
ode45	2/6/5	22/12/10	fd2	8.4	16	8.4
ode45	1/12/10	44/24/20	fd2	60	64	61

Table 4
Nonlinear `MPC` on a `CSTR`: `rOPT` and `rFOPT` method.

Discretization method	Horizons $h/P/M$	# vars. tot/dep/free	Jacobian analytic/fd1/fd2	CPU time (s) No ineqs. G-N	CPU time (s) No ineqs. BFGS	CPU time (s) Ineqs. w/`BFGS` w/slacks	CPU time (s) Ineqs. w/`BFGS` feasible
Explicit Euler	2/6/5	22/12/10	analytic	16	3.1	3.7	8.5
			fd1	28	3.5	5.4	7.5
			fd2	26	3.9	-	-
			AD	12	3.5	4.2	8.1
	1/12/10	44/24/10	analytic	36	6.0	6.2	14
			fd1	73	10	10	21
			fd2	92	13	-	-
			AD	52	7.8	7.8	17
Implicit Euler	2/6/5	22/12/10	analytic	9.7	3.0	3.5	5.7
			fd1	16	3.7	5.0	9.6
			fd2	19	4.0	-	-
			AD	12	3.4	4.1	6.5
	1/12/10	44/24/10	analytic	fail	5.6	6.7	16
			fd1	fail	16	11	25
			fd2	fail	12	-	-
			AD	fail	7.3	8.5	19
Lobatto IIIC	2/6/5	34/24/10	analytic	9.9	3.3	4.2	14
			fd1	113	5.0	7.1	20
			fd2	32	5.8	-	-
			AD	13	4.3	5.2	15
	1/12/10	68/48/20	analytic	269	7.3	7.7	14
			fd1	502	15	15	24
			fd2	-	29	-	-
			AD	131	10	11	17
RK4	2/6/5	70/60/10	analytic	14	3.8	5.1	13
			fd1	39	7.5	12	16
			fd2	501	21	-	-
			AD	26	5.1	7.5	10
	1/12/10	140/120/20	analytic	412	9.0	11	20
			fd1	fail	24	27	48
			fd2	3312	167	-	-
			AD	632	17	18	30