Real-Time Sequential Convex Programming for Nonlinear Model Predictive Control and Application to a Hydro-Power Plant

Quoc Tran Dinh*†, Carlo Savorgnan* and Moritz Diehl*

***Abstract*— In this paper we propose a new algorithm for solving nonlinear optimal control problems which is called real-time sequential convex programming (RTSCP). The main difference between this approach and the previous real-time iteration algorithms is that RTSCP linearizes only the nonconvex parts of the problem while it preserves all the convex structure which can be exploited by standard convex optimization techniques. The algorithm is applied to the control of a hydro power plant with 259 states and 10 controls. The numerical results show the benefits that the proposed method offers when compared to standard ones.**

1. Introduction

Model predictive control (MPC), or receding horizon control, is a powerful tool for many applications [1], [13], [15]. MPC requires the online solution of an optimization problem at every sampling time. The sequence of problems to be solved depends parametrically on the value of the state at the current time and maintains the same structure. This fact can be exploited numerically to obtain a solution efficiently.

When nonlinear model predictive control (NMPC) is considered, traditional optimization methods do not always meet the real-time requirements imposed by real-world applications. The nonlinear programming problems (NLPs) to be solved at every sampling time are usually solved with iterative methods. When the NLP problem is obtained by a direct single or multiple shooting method [4], sequential quadratic programming (SQP) is often used to solve the resulting optimization problems. SQP solves a sequence of quadratic approximations of the NLP problem to converge to a local solution. Another iterative technique which is often used in practice is the constrained Gauss-Newton method [3]. A major problem for the real-time application of these methods is that the computational cost corresponding to every iteration may be high, making the total time required to obtain a solution large compared to the sampling time. To overcome this problem, the *real-time iteration* (RTI) scheme was introduced in [9]. RTI does not solve the optimization problem until completely converged but uses a special transition between subsequent problems and performs only one iteration of the optimization method such as SQP or Gauss-Newton. If the subsequent problems to be solved online do not vary too much and the sampling time is sufficiently small, the approximated solution given by RTI tracks the exact solution of the optimization problem within a given accuracy. Proofs of nominal stability of RTI can be found in [8], [10].

One possible drawback in the application of standard RTI is that the local approximation used to characterize the NLP problem is always quadratic programming models (QP) and may not capture important features, e.g., convexity of the problem, resulting in a poor tracking of the optimal solution. This problem may occur in particular when some nonlinear convex constraints are linearized.

a) Paper contribution: In this paper, we present a new algorithm in which we combine RTI with sequential convex programming (SCP). Similarly to SQP, also SCP is an iterative method. However, in the SCP algorithm only the nonconvex parts of the problem are convexified while all the convex structures of the problem are preserved and exploited by using convex optimization techniques.

In the algorithm we propose *real-time sequential convex programming* (RTSCP), we solve only one convex optimization problem per sampling period. Like in standard SCP, we keep all the convex structure of the NLP problem in order to have a more faithful model compared to the one obtained by linearizing all the constraints.

To show the effectiveness of the method we apply it to the control of a hydro power plant with 259 state variables and 10 control inputs. The numerical simulation is implemented and compared to the conventional approach as well as the real-time Gauss-Newton approach.

b) Paper organization: Section 2 introduces sequential convex programming and real-time sequential convex programming. In Section 3 we describe the problem formulation we used in the simulations. Section 4 presents the numerical results. The conclusion and future work are discussed in Section 5.

c) Notation: For a given vector $x \in \mathbb{R}^n$, the norm $\|x\|_S$ is defined as $\|x\|_S = \sqrt{x^T S x}$ for any symmetric positive definite matrix S, $|||X|||$ is the Frobenius norm of a matrix X. For a vector valued function g from $\mathbb{R}^n$ to $\mathbb{R}^m$, $g'(x) \in \mathbb{R}^{m \times n}$ denotes its Jacobian matrix at x.

*Optimization in Engineering Center (OPTEC) and Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium, { quoc.trandinh, carlo.savorgnan, moritz.diehl}@esat.kuleuven.be

† Department of Mathematics-Mechanics-Informatics, Hanoi University of Science, Hanoi, Vietnam.

2. SEQUENTIAL CONVEX PROGRAMMING AND REAL-TIME IMPLEMENTATION

As mentioned in the introduction, the underlining optimization problem to be solved in NMPC is a parametric optimal control problem. By using the *initial valued embedding* technique [8], the parameter linearly enters into this problem as $x(t_0) - \xi = 0$, where $x(t_0)$ is an initial state at t_0 and ξ the parameter (see formula (11) in Section 3). Then, by applying a direct transcription, this optimal control problem can be transformed into a structured (and large-scale) parametric nonlinear optimization problem, where the nonlinear equality constraint originates from the dynamics by integrating it on a given discretized time grid.

This problem can be summarized in the following form:

$$\begin{cases} \min\limits_{w\in\mathbb{R}^n} & f(w) \\ \text{s.t.} & g(w) + M\xi = 0, \\ & w \in \Omega. \end{cases} \qquad (\mathrm{P}(\xi))$$

Here, without loss of generality, we assume that $f : \mathbb{R}^n \to \mathbb{R}$ is convex, $\Omega \in \mathbb{R}^n$ is a nonempty, closed convex set, $g : \mathbb{R}^n \to \mathbb{R}^m$ is nonlinear and continuously differentiable, $\xi \in \mathscr{P}$ is referred to as an input parameter, where $\mathscr{P}$ is a nonempty closed subset in $\mathbb{R}^p$, and $M \in \mathbb{R}^{m\times p}$ is a given matrix which embeds the parameter ξ into the nonlinear constraints. In other cases, slack variables can be used.

Let us denote by $S(\xi)$ the set of Karush-Kuhn-Tucker (KKT) points $\bar{z}(\xi) = (\bar{w}(\xi), \bar{\lambda}(\xi))$ of problem $\mathrm{P}(\xi)$. Throughout this section, we assume that $S(\xi)$ is nonempty for all $\xi \in \mathscr{P}$. As usual, $\bar{w}(\xi)$ is referred to as a stationary point and $\bar{\lambda}(\xi)$ is the Lagrange multiplier associated with the constraint $g(w) + M\xi = 0$. We also assume that f and g are twice continuously differentiable on their domains.

This section presents two algorithmic frameworks. First, we propose a local optimization method for solving nonlinear optimization problems, which is called sequential convex programming (SCP). The nonlinear equality constraint $g(w) + M\xi = 0$ is convexified by linearizing it around a given point w^j, while maintaining the convexity of the objective function and the constraint $w \in \Omega$. Then, we apply the SCP method to solve problem $\mathrm{P}(\xi)$ when parameter ξ varies on its domain. Such a method is called a conventional NMPC approach or a full-SCP algorithm. Finally, we combine the SCP algorithm and the real-time iteration scheme in [9] in order to obtain a real-time SCP algorithm for solving $\mathrm{P}(\xi)$.

A. Sequential convex programming

For a given $w^j \in \Omega$, we linearize the nonlinear equality constraint $g(w) + M\xi = 0$ around this point to obtain the following convex optimization subproblem:

$$\begin{cases} \min\limits_{w\in\mathbb{R}^n} & f(w) \\ \text{s.t.} & g(w^j) + g'(w^j)(w - w^j) + M\xi = 0, \\ & w \in \Omega, \end{cases} \qquad (\mathrm{P}(w^j, \xi))$$

Now, we fix the parameter ξ at $\xi = \bar{\xi}$. The SCP algorithm for solving $\mathrm{P}(\bar{\xi})$ is described as follows.

SCP ALGORITHM.

Initialization. Find an initial point $w^0 \in \Omega$ and set $j := 0$.

Iteration. For a given $w^j \in \Omega$, perform the following steps.

Step 1: Evaluate $g(w^j)$ and its Jacobian matrix $g'(w^j)$.

Step 2: Solve the convex subproblem $\mathrm{P}(w^j, \xi)$ with $\xi = \bar{\xi}$ to obtain a solution w_+^j. Set $\Delta w^j := w_+^j - w^j$ as a search direction.

Step 2: If $\|\Delta w^j\| \le \varepsilon$ and $\|g(w^j)\| \le \varepsilon$ then terminate. Otherwise, find an appropriate step size $t_j \in (0, 1]$. Set $w^{j+1} := w^j + t_j \Delta w^j$. Increase j by 1 and go back to Step 1.

The step size t_j in the SCP algorithm can either be fixed at a certain value in $(0, 1]$ or be dynamically updated using a line search procedure based on a merit function (see [14]). If we choose $t_j = 1$ for all j then the algorithm is called a full-step SCP method. For solving the convex subproblem $\mathrm{P}(w^j, \xi)$, one can implement an optimization algorithm such as an interior point method to exploit the problem structure or rely on available software. Further discussion on the local convergence of the SCP algorithm can be found in [20].

B. Real-time SCP algorithm

Now, we consider a real-time implementation of the SCP algorithm by combining it with the real-time iteration scheme [9]. Instead of solving completely the nonlinear optimization problem at each sampling time, we only perform *one step* of the *full-step SCP* algorithm, i.e. $j = 0$ to obtain an approximate solution. In other words, one convex subproblem of the form $\mathrm{P}(w^j, \xi)$ is required to be solved at each time interval. In summary, the algorithm is presented as follows.

REAL-TIME SCP ALGORITHM (RTSCP).

Initialization. Fix a starting parameter $\xi_0 \in \mathscr{P}$ and solve approximately problem $\mathrm{P}(\xi)$ for fixed $\xi = \xi_0$ to get a solution $w^0 \in \Omega$ as an initial point. Set $k := 0$.

Iteration. Perform the following steps:

Step 1: Evaluate $g(w^k)$ and the Jacobian matrix $g'(w^k)$.

Step 2: Obtain a new value of parameter $\xi_{k+1} \in \mathscr{P}$.

Step 3: Solve the convex subproblem $\mathrm{P}(w^k, \xi_{k+1})$ to get a solution w^{k+1}.

Step 4: Set $k := k+1$ and go back to Step 1.

In NMPC applications treated with shooting methods, evaluating the function g and its Jacobian matrix at a certain point is usually time consuming due to the integration of the dynamics. On the other hand, solving the convex subproblem $\mathrm{P}(w^j,\xi)$ requires less computational time by using an appropriate solver and exploiting the structure of the problem. In the application investigated in Section 4, the first task amounts up to $80\% - 90\%$ of the total computational time.

C. The stability of RTSCP.

Finally, we show that under certain assumptions, the RTSCP algorithm ensures the stability of the approximate solutions on the moving horizon. In other words, if the algorithm starts from z^0 close to the true KKT point $\bar{z}^0$ of $\mathrm{P}(\xi_0)$ then in the sampling time, the approximation z^1 is still close to the true KKT point $\bar{z}^1$ of $\mathrm{P}(\xi_1)$ provided that $\Delta\xi_0 := \xi_1 - \xi_0$ is sufficiently small.

To prove a theoretical result on the stability of the tracking error, two essential assumptions are required. For a given $\xi_k \in \mathscr{P}$, we denote $\bar{z}^k = (\bar{w}^k, \bar{\lambda}^k) := \bar{z}(\xi_k)$ and make the following assumptions.

Assumption A1. The following perturbed convex problem

$$\begin{cases} \min\limits_{w} & f(w) + \delta_f^T(w - \bar{w}^k) \\ \text{s.t.} & g(\bar{w}^k) + g'(\bar{w}^k)(w - \bar{w}^k) + M\xi_k = \delta_g, \\ & w \in \Omega, \end{cases} \tag{1}$$

has a unique KKT point $\bar{z}(\delta_f, \delta_g)$. Moreover, this KKT mapping is Lipschitz continuous with respect to $\delta = (\delta_f, \delta_g)$ with a Lipschitz constant $\gamma > 0$, i.e.:

$$\|\bar{z}(\delta) - \bar{z}(\delta')\| \leq \gamma \|\delta - \delta'\|,$$

for all δ and δ' in a neighborhood of the origin.

Assumption A2. The value of the second derivative $\nabla_w^2 L_g(w,\lambda)$ of $L_g(w,\lambda) := \lambda^T g(w)$ at $\bar{z}^k$ with respect to w satisfies: $|||\nabla_w^2 L_g(\bar{w}^k, \bar{\lambda}^k)||| \leq \kappa$ with $\kappa\gamma < 1$.

Discussion on Assumptions A1 and A2. Assumption A1 relates to the *strong regularity* of the KKT system of problem $\mathrm{P}_0(\xi)$ at $\bar{z}^k$. The concept of *strong regularity* was first introduced by Robinson in [16] and is a standard assumption in optimization as well as nonlinear analysis [17]. If the convex set Ω is polyhedral and the linear independence constraint qualification (LICQ) holds then the strong regularity is equivalent to the strong second order sufficient optimality condition in optimization [11]. Assumption A2 regards the second term in the Hessian matrix of the Lagrange function $L(w,\lambda) := f(x) + \lambda^T[g(x) + M\xi]$ at a KKT point, which requires it to be sufficiently small. This is similar to the κ-assumption in the analysis of constrained Gauss-Newton-type methods [5].

Theorem 2.1 (Contraction Theorem): Suppose that Assumptions **(A1)**-**(A2)** are satisfied. Then there exist neighborhoods $\mathcal{N}_\tau$ of ξ_k, $\mathcal{N}_\rho$ of $\bar{z}^k$ and a single-valued function $\bar{z}: \mathcal{N}_\tau \to \mathcal{N}_\rho$ such that for all $\xi_{k+1} \in \mathcal{N}_\tau$, $\bar{z}^{k+1} := \bar{z}(\xi_{k+1})$ is the unique KKT point of $\mathrm{P}(\xi_{k+1})$ in $\mathcal{N}_\rho$ with respect to parameter ξ_{k+1} (i.e. $S(\xi_{k+1}) \neq \emptyset$). Moreover, for any $\xi_{k+1} \in \mathcal{N}_\tau$, $z^k \in \mathcal{N}_\rho$ we have

$$\|z^{k+1} - \bar{z}^{k+1}\| \leq \omega \|z^k - \bar{z}^k\| + c_0 \|M(\xi_{k+1} - \xi_k)\|, \tag{2}$$

where $\omega \in (0,1)$, $c_0 > 0$ are two given constants, and z^{k+1} is a KKT point of $\mathrm{P}(w^k, \xi_{k+1})$.

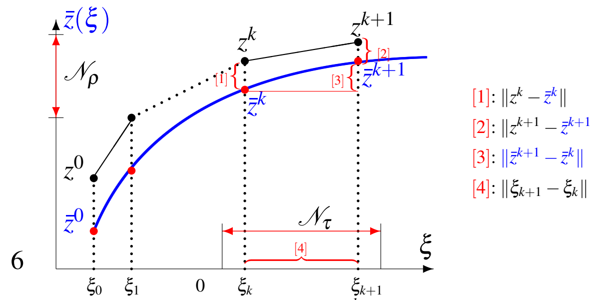

Fig. 1. The approximate sequence $\{z^k\}_k$ along the KKT trajectory $\bar{z}(\cdot)$.

Discussion of Theorem 2.1. As stated in the estimate (2), if the RTSCP algorithm starts from z^0 close to $\bar{z}^0$, i.e. $\|z^0 - \bar{z}^0\| \leq \varepsilon$ for a given $\varepsilon > 0$, then when $\Delta\xi_0 = \xi_1 - \xi_0$ is sufficiently small, the next approximation z^1 is still close to the true KKT point $\bar{z}^1$, i.e. $\|z^1 - \bar{z}^1\| \leq \varepsilon$. By induction, we can conclude that the whole approximate sequence $\{z^k\}$ generated by the *RTSCP algorithm* tracks the true KKT sequence $\{\bar{z}^k\}$ along the moving horizon provided that the initial point z^0 is sufficiently close to $\bar{z}^0$ and the parameter change $\Delta\xi^k$ is sufficiently small. This observation is illustrated in Fig. 1. A detailed discussion on the assumptions as well as the proof of Theorem 2.1 can be found in [21].

3. Control of a Hydro Power Plant: Problem Formulation

A. Dynamic model

We consider a hydro power plant composed of several subsystems connected together. The system includes six dams with turbines D_i $(i = 1, \ldots, 6)$ located along a river and three lakes L_1, L_2 and L_3 as visualized in Fig. 2. U_1 is a duct connecting lakes L_1 and $L2$. T_1 and T_2 are ducts equipped with turbines and C_1 and C_2 are ducts equipped with turbines and pumps. The flows through the turbines and pumps are the controlled variables. The complete model with all the parameters can be found in [18]. The dynamics of the lakes is given by

$$\frac{\partial h(t)}{\partial t} = \frac{q_{\text{in}}(t) - q_{\text{out}}(t)}{S}, \tag{3}$$

where $h(t)$ is the water level and S is the surface area of the lakes; q_{in} and q_{out} are the input and output flows, respectively. The dynamics of the reaches R_i $(i = 1, \ldots, 6)$ is described by the one-dimensional Saint-Venant partial

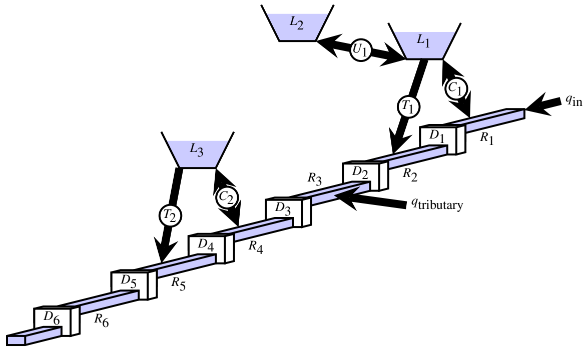

Fig. 2. Overview of the hydro power plant.

differential equation:

$$\begin{cases} \frac{\partial q(t,y)}{\partial y} + \frac{\partial s(t,y)}{\partial t} = 0, \\ \frac{1}{g}\frac{\partial}{\partial t}\left(\frac{q(t,y)}{s(t,y)}\right) + \frac{1}{2g}\frac{\partial}{\partial y}\left(\frac{q^2(t,y)}{s^2(t,y)}\right) + \frac{\partial h(t,y)}{\partial y} + I_f(t,y) - I_0(y) = 0. \end{cases} \tag{4}$$

Here, y is the spatial variable along the flow direction of the river, q is the river flow (or discharge), s is the wetted surface, h is the water level with respect to the river bed, g is the gravity, I_f is the friction slope and I_0 is the river bed slope. The partial differential equation (4) can be discretized by applying the method of lines in order to obtain a system of ordinary differential equations. Stacking all the equations together, we represent the dynamics of the system by

$$\dot{x}(t) = f(x,u), \tag{5}$$

where the state vector $x \in \mathbb{R}^{n_x}$ includes all the flows and the water levels and $u \in \mathbb{R}^{n_u}$ represents the input vector. The dynamic system consists of $n_x = 259$ states and $n_u = 10$ controls. The control inputs are the flows going in the turbines, the ducts and the reaches.

B. Nonlinear MPC formulation

Associated with the hydro power plant dynamic model (5), we are interested in the following NMPC setting:

$$\begin{array}{ll} \min\limits_{x,u} & J(x(\cdot),u(\cdot)) \\ \text{s.t.} & \dot{x} = f(x,u),\ x(t) = x_0(t), \\ & u(\tau) \in U,\ x(\tau) \in X,\ \tau \in [t, t+T] \\ & x(t+T) \in \mathscr{R}_T, \end{array} \tag{6}$$

where the objective function $J(x_0(t), u(\cdot)$ is given by

$$\begin{aligned} J(x(\cdot),u(\cdot)) := & \int_t^{t+T} \left[\|x(\tau)-x_s\|_P^2 + \|u(\tau)-u_s\|_Q^2\right] d\tau \\ & + \|x(t+T)-x_s\|_S^2. \end{aligned} \tag{7}$$

Here P, Q and S are given symmetric positive definite weighting matrices, and (x_s, u_s) is a steady state of the dynamics (5). The control variables are bounded by lower and upper bounds, while some state variables are also bounded and the others are unconstrained. Consequently, X and U are boxes in $\mathbb{R}^{n_x}$ and $\mathbb{R}^{n_u}$, respectively, but X is not necessarily bounded. The terminal region $\mathscr{R}_T$ is a control-invariant ellipsoidal set centered at x_s of radius $r > 0$ and scaling matrix S, i.e.:

$$\mathscr{R}_T := \left\{ x \in \mathbb{R}^{n_x} \mid (x-x_s)^T S (x-x_s) \leq r \right\}. \tag{8}$$

To compute matrix S and the radius r in (8) the procedure proposed in [7] can be used. In [12] it has been shown that the receding horizon control formulation (6) ensures the stability of the closed-loop system under mild assumptions. Therefore, the aim of this example is to track the steady state of the system and to ensure the stability of the system by satisfying the terminal constraint along the moving horizon. To have a more realistic simulation we added a disturbance to the input flow q_{in} at the beginning of the reach R_1 and the tributary flow $q_{\text{tributary}}$.

The matrices P and Q have been set to

$$P := \text{diag}\left(\frac{0.01}{(x_s)_i^2 + 1} \ : \ 1 \leq i \leq n_x\right), \tag{9}$$

$$Q := \text{diag}\left(\frac{4}{(u_l + u_b)_i^2 + 1} \ : \ 1 \leq i \leq n_u\right), \tag{10}$$

where u_l and u_b is the lower and upper bound of the control input u.

C. A short description of the multiple shooting method

In this subsection we briefly describe the multiple shooting formulation [4] which we use to discretize the continuous time problem (6). The time horizon $[t, t+T]$ of $T = 4$ hours is discretized into $N = 16$ shooting intervals with every $\Delta\tau = 15$ minutes such that $\tau_0 = t$ and $\tau_{i+1} := \tau_i + \Delta\tau$ $(i = 0, \ldots, N-1)$. The control $u(\cdot)$ is parametrized by using a piecewise constant function $u(\tau) = u_i$ for $\tau_i \leq \tau \leq \tau_i + \Delta\tau$ $(i = 0, \ldots, N-1)$.

Let us introduce $N+1$ shooting node variables s_i $(i = 0, \ldots, N)$. Then, by integrating the dynamic system $\dot{x} = f(x,u)$ in each interval $[\tau_i, \tau_i + \Delta\tau]$, the continuous dynamic (5) is transformed into nonlinear equality constraints of the form:

$$g(w) + M\xi := \begin{bmatrix} s_0 - \xi \\ x(s_0, u_0) - s_1 \\ \cdots \\ x(s_{N-1}, u_{N-1}) - s_N \end{bmatrix} = 0. \tag{11}$$

Here, vector w combines all the controls u_i and shooting node variables s_i as $w = (s_0^T, u_0^T, \ldots, s_{N-1}^T, u_{N-1}^T, s_N^T)^T$, ξ is the initial state $x_0(t)$ which is considered as a parameter, and $x(u_s, x_s)$ is the result of the integration of the dynamics from τ_i to $\tau_i + \Delta\tau$ where we set $u(\tau) = u_i$ and $x(\tau_i) = s_i$.

The objective function is approximated by

$$J(w) = \sum_{i=0}^{N-1} \left[\|s_i - x_s\|_P^2 + \|u_i - u_s\|_Q^2\right] + \|s_N - x_s\|_S^2 \tag{12}$$

while the constraints are imposed only at $\tau = \tau_i$ (the beginning of the intervals)

$$s_i \in X,\ u_i \in U,\ x_N \in \mathscr{R}_T, (i = 0, \ldots, N-1). \tag{13}$$

If we define $\Omega := U^N \times (X^N \times \mathscr{R}_T) \subset \mathbb{R}^{n_w}$ then Ω is convex. Moreover the objective function (12) is convex quadratic. Therefore, the resulting optimization problem is indeed of the form $\mathrm{P}(\xi)$. Note that Ω is not a box but a curved convex set due to $\mathscr{R}_T$.

The nonlinear program to be solved at every sampling time has 4563 decision variables and 4403 equality constraints.

4. Numerical Experiments

In this section we present the results of the simulation we performed and we give some details on the implementation. To evaluate the performance of the method proposed in this paper we implemented the following algorithms:

- Full-NMPC – the nonlinear program obtained by multiple shooting is solved at every sampling time until convergence by several SCP iterations.
- RTSCP – the solution of the nonlinear program is approximated by applying only one SCP iteration using the initial value embedding. The structure of Ω is preserved.
- RTGN – the solution of the nonlinear program is approximated by solving a quadratic program obtained by linearizing the dynamics and the terminal constraint $x_N \in \mathscr{R}_T$. This method can be referred to as a constrained Gauss-Newton method.

A. Implementation details

To compute the set $\mathscr{R}_T$ a mixed MATLAB and C++ code has been used. The computed value of r is 1.687836, while the matrix S is dense, symmetric and positive definite.

The quadratic programs (QPs) and the quadratically constrained quadratic programming problems (QCQPs) arising in the algorithms we implemented can be efficiently solved by means of interior point or other methods [6]. In our implementation, we used the commercial solver `CPLEX` which can deal with both types of problems.

All the tests have been implemented in C++ running on a 16 cores workstation with 2.7GHz Intel®Xeron CPUs and 12 GB of RAM. We used `CasADi`, an open source C++ package [2] which implements automatic differentiation to calculate the derivatives of the functions and offers an interface to `CVODES` from the `Sundials` package [19] to integrate ordinary differential equations and compute sensitivities. The integration has been parallized using `openmp`.

In the full-NMPC algorithm we perform at most 5 SCP iterations for each time interval. We terminate the SCP algorithm when the relative infinity-norm of the search direction as well as of the feasibility gap reached the tolerance $\varepsilon = 10^{-3}$. To have a fair comparison of the different methods, the starting point w^0 of the RTSCP and RTGN algorithms has been set to the solution of the first full-NMPC iteration.

The disturbance on the flows q_{in} and $q_{\text{tributary}}$ are generated randomly and varying from 0 to 25 and 0 to 8, respectively. All the simulations are perturbed at the same disturbance scenario.

B. Numerical results

We simulated the algorithms for $H_p = 30$ time intervals. The average time required by three methods is summarized in Table I. Here, `AvIntTime` is the average time in seconds needed to evaluate the function g and its Jacobian; `AvSolTime` is the average time for solving the QP or QCQP problems; `Total` corresponds to the sum of the previous terms and some preparation time. On average, the full-NMPC algorithm needed 3.7 iterations to converge to a solution.

TABLE I

The average time of three methods

`Methods`	`AvEvalTime[s]`	`AvSolTime[s]`	`Total[s]`
Full NMPC	240.84 (84.4%)	39.81 (14.0%)	285.43
RTSCP	79.42 (82.2%)	15.27 (15.8%)	96.56
RTGN	81.37 (92.2%)	5.07 (5.7%)	88.25

It can be seen from Table I that evaluating the function and its Jacobian matrix costs $80\% - 90\%$ of the total time. On the other hand, solving a QCQP problem is almost three times more expensive than solving a QP problem. The most time consuming procedure at every iteration is the integration of the dynamics and its linearization.

The control profiles of the simulation are illustrated in Figures 3 and 4. Here, the first figure shows the flows in the turbines and the ducts of lakes L_1 and L_2, while the second one plots the flows to be controlled in the reaches R_i $(i = 1, \ldots, 6)$. We can observe that the control profiles achieved by RTSCP are close to the profiles obtained by Full-NMPC, while the results from RTGN oscillate in the first intervals due to the violation of the terminal constraint. The terminal constraint in the RTSCP algorithm is active in many iterations.

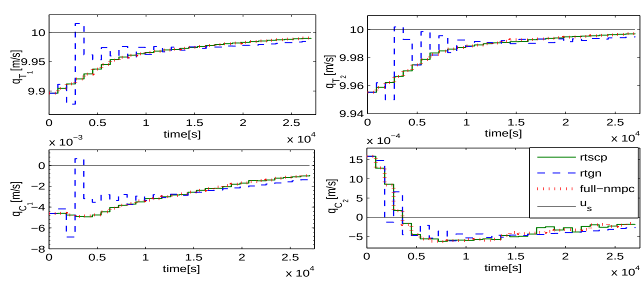

Fig. 3. The controller profiles q_{T_1}, q_{C_1}, q_{T_2} and q_{C_1}.

Figure 5 shows the relative tracking error of the solution of the nonlinear programming problem of the RTSCP and

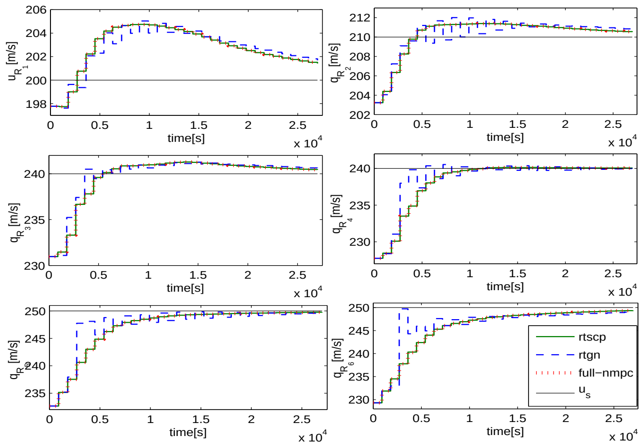

Fig. 4. The controller profiles of $q_{R_1}, \ldots, q_{R_6}$.

RTGN algorithms when compared to the full-NMPC one. The error is quite small in RTSCP while it is higher in the

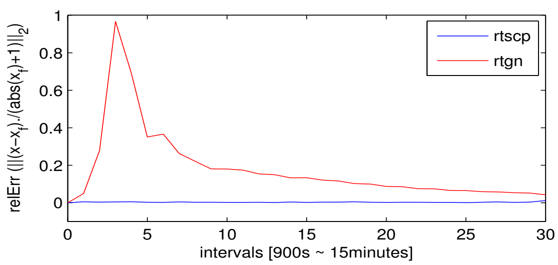

Fig. 5. The relative errors of RTSCP and RTGN compared to Full-NMPC.

RTGN algorithm. This happens because the linearization of the quadratic constraint can not adequately capture the shape of the terminal constraint $x_N \in \mathscr{R}_T$.

5. Conclusion

A new method called *real-time sequential convex programming* for solving NMPC is proposed. This method is suitable for problems that possess convex substructures which can be efficiently handled by using convex optimization techniques.

Our future work is to develop a complete theory for this approach and apply it to new problems. For example, in some robust control problem formulations as well as robust optimization formulations, where we consider worst-case performance within robust counterparts, a nonlinear programming problem with second order cone and semidefinite constraints needs to be solved.

Acknowledgment. Research supported by Research Council KUL: CoE EF/05/006 Optimization in Engineering(OPTEC), IOF-SCORES4CHEM, GOA/10/009 (MaNet), GOA/10/11, several PhD/postdoc and fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects G.0452.04, G.0499.04, G.0211.05, G.0226.06, G.0321.06, G.0302.07, G.0320.08, G.0558.08, G.0557.08, G.0588.09,G.0377.09, research communities (ICCoS, ANMMM, MLDM); IWT: PhD Grants, Belgian Federal Science Policy Office: IUAP P6/04; EU: ERNSI; FP7-HDMPC, FP7-EMBOCON, Contract Research: AMINAL. Other: Helmholtz-viCERP, COMET-ACCM, ERC-HIGHWIND, ITN-SADCO.

References

[1] Allgöwer, F. and Zheng, A. (eds), *Nonlinear Model Predictive Control*, Birkhuser Verlag, Basel, (2000).

[2] Andersson, J., Houska, B. and Diehl, M., Towards a Computer Algebra System with Automatic Differentiation for use with Object-Oriented modelling languages, *In Proc. of the 3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Oslo, Norway, October 3, (2010).

[3] Bock, H.G., Deuflhard, P. and Hairer, E. (eds.), *Recent advances in parameter identification techniques for ODE*, Numerical Treatment of Inverse Problems in Differential and Integral Equations, Birkhäuser, (1983).

[4] Bock, H.G. and Plitt, K.J., A multiple shooting algorithm for direct solution of optimal control problems, Proceedings 9th IFAC World Congress Budapest, 243–247, Pergamon Press, (1984).

[5] Bock, H.G., Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen, Universität Bonn, PhD Thesis, 183 pages (1987).

[6] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[7] Chen, H. and Allgöwer, F., A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability, *Automatica*, 34(10), 1205–1217 (1998).

[8] Diehl, M., Real-Time Optimization for Large Scale Nonlinear Processes, Universität Heidelberg, (2001), `http://www.ub.uni-heidelberg.de/archiv/1659/`.

[9] Diehl, M., Bock, H.G., Schlöder, J.P., Findeisen, R., Nagy, Z. and Allgöwer, F., Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations, *J. Proc. Contr.*, 12(4):577–585, (2002).

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

[11] Dontchev, A.L. and Rockafellar, T.R., Characterizations of Strong Regularity for Variational Inequalities over Polyhedral Convex Sets, *SIAM J. Optim.*, 6(4), 1087–1105, (1996).

[12] Jadbabaie, A. and Hauser, J., On the stability of receding horizon control with a general terminal cost, *IEEE Trans. Automatic Control*, 50(5), 674–678 (2005).

[13] Kouvaritakis B. and Cannon, M. (eds), *Model Predictive Control: Theory and Practice*, The Institution of Engineering and Technology, London, US (2001).

[14] Nocedal, J. and Wright, S., *Numerical Optimization*, Springer-Verlag, Berlin (2007).

[15] Rawlings, J.B., E.S. Meadows and K.R. Muske, Nonlinear model predictive control: A tutorial and survey, *ADCHEM94 Proceedings*, Kyoto, Japan (1994).

[16] Robinson, S.M., Strongly regular generalized equations, *Mathematics of Operations Research*, 5(1):43-62, (1980).

[17] Rockafellar, T.R. and Wets, R.J-B., *Variational Analysis*, Springer-Verlag, New York, (1997).

[18] Savorgnan, C. and Diehl, M., Control benchmark of a hydro power plant, *Tech. Report*. (2010), URL: `http:/homes.esat.kuleuven.be/ mdiehl/`.

[19] Serban, R. and Hindmarsh, A.C., CVODES: the sensitivity-enabled ODE solver in SUNDIALS, *In Proceedings of IDETC/CIE* 2005, (2005).

[20] Tran Dinh, Q. and Diehl, M., Local convergence of sequential convex programming for nonlinear optimization, In M. Diehl, F. Glineur, E. Jarlebring, and W. Michiels (eds.), *Recent Advances in Optimization and its Applications in Engineering*, pages 93–102. Springer-Verlag, Berlin, Heidelberg, (2010).

[21] Tran Dinh, Q., Savorgnan, C. and Diehl, M., Real-Time Sequential Convex Programming for Optimal Control Applications In: Bock, H. et al (eds), *Modeling, Simulation and Optimization of Complex Processes*, Springer-Verlag, (2009) (accepted). URL: `http://arxiv.org/abs/1105.3427`.