sciendo

Fast nonlinear model predictive control of a chemical reactor: a random shooting approach

Peter Bakaráč, Michal Kvasnica

Department of Information Engineering and Process Control Institute of Information Engineering, Automation, and Mathematics FCFT STU in Bratislava, Radlinského 9, 812 37 Bratislava, Slovakia
peter.bakarac@stuba.sk

Abstract: This paper presents a fast way of implementing nonlinear model predictive control (NMPC) using the random shooting approach. Instead of calculating the optimal control sequence by solving the NMPC problem as a nonlinear programming (NLP) problem, which is time consuming, a sub-optimal, but feasible, sequence of control inputs is determined randomly. To minimize the induced sub-optimality, numerous random control sequences are selected and the one that yields the smallest cost is selected. By means of a motivating case study we demonstrate that the random shooting-based approach is superior, from a computational point of view, to state-of-the-art NLP solvers, and features a low level of sub-optimality. The case study involves a continuous stirred tank reactor where a fast multi-component chemical reaction takes place.

Keywords: nonlinear model predictive control, random shooting, continuous stirred tank reactor

1. Introduction

Controlling processes with a nonlinear dynamics, such as chemical reactors, is a challenging tasks especially if a safe operation of the process is of paramount importance. Among all available control strategies, model predictive control (MPC) (Maciejowski, 2002) is among the most successful ones as it enables to select control inputs such that constraints can be enforced while optimizing the overall process behavior. In fact, the survey of (Qinand Badgwell, 2003) shows that MPC is indeed the most popular and preferred control approach adopted by the process industry. A plethora of MPC success stories is reported in the literature especially for controlling chemical reactors (Oravec and Bakošová, 2012; Bakošová and Oravec, 2014; Oravec and Bakošová, 2015; Bakošová et al., 2013), distillation columns (Martin et al., 2013), and heat exchangers (Oravec et al., 2016, 2018), to name just a few.

MPC optimizes the control inputs by employing a mathematical model of the controlled process to predict its future behavior. Conventionally, MPC is based on linear prediction models. The advantage of such models is that the selection of optimal control inputs is relatively simple and can be done quickly. However, linear models have only limited validity and often do not sufficiently capture all nuances of the controlled process. Therefore, one would prefer to solve the MPC problem using a nonlinear model of the processes. Such nonlinear MPC (NMPC), however, is often non-convex, renderingthe subsequent optimization a tedious and time-consuming task. The increased computational requirements of NMPC often prohibit its implementation in real time as the optimal control actions need to be determined within the duration of one sampling period.

Nonlinear model predictive control is therefore a vibrant research field (Allgöwer and Zheng, 2012) with the primary aim of reducing the computational load associated with solving a non-convex optimization problem. Numerous approaches are available in the literature for solving such problems either to local or to the global optimality. Local methods (Wright and Nocedal, 1999), such as steepest descent or sequential quadratic programming approaches, are typically based on moving along the gradient of objective function and the constraints and therefore require certain regularity assumptions, such as all functions (including the prediction model) to be once or twice continuously differentiable. Global methods are either deterministic or stochastic. Deterministic methods often employ convex relaxations coupled with a branch-and-bound exploration of the search space (Čižniar et al., 2009), but they are limited, from a practical point of view, to problems of a small size. Stochastic approaches, such as simulated annealing (Kirkpatrick et al., 1983) or particle swarm optimization (Poli et al., 2007) are based on investigating a large number of random solutions, each evaluated using the NMPC cost function by selecting the candidate with the lowest cost. However, they are unable to exploit the structure of the NMPC problem and therefore feature a slow performance.

In this paper we propose to solve non-convex NMPC problems using the random shooting approach, which is popular in the robotics (Piovesan and Tanner, 2009) and machine learning (Sahoo

et al., 2018) communities. The method is based on generating a large number of random control sequences, followed by selecting the random sequence that is feasible (i.e., satisfies all constraints over the whole prediction window) and features the best value of the performance index. Even though the resulting control sequence is sub-optimal, it is still feasible, thus guarantees a safe operation of the controlled processes. Moreover, sub-optimality can be reduced by increasing the number of random scenarios. Despite its naivety, the approach performs well in real-life situations due to its very simple and fast implementation. In fact, if the underlying optimization problem is convex, even hard bounds on the runtime and the sub-optimality of the random shooting approach can be given (Dyer et al., 2014). Most importantly, the random shooting approach assumes no regularity conditions on the performance index and the constraint sets, and is therefore very versatile. As an example, it can be applied to solve NMPC where the prediction model and/or the performance index are discontinuous/non-differentiable, or if the constraint sets are non-convex and even discontinuous, such as finite sets. Moreover, the algorithm only requires the *membership oracle* of the constraints and an *evaluation oracle* of the performance index (Dyer et al., 2014). Therefore, black-box versions of the performance index and the constraints can be handled by the random-shooting approach without difficulties.

In this paper we apply random shooting to solve an NMPC problem of controlling a continuous stirred tank reactor (CSTR) in which a fast multi-component chemical reaction $A \leftrightarrows 2C \rightarrow B$ takes place (Fissore, 2008). By a simulation study we show that the random shooting approach features a 100-times faster computation of control inputs compared to a global solver, and features just a small level of sub-optimality.

2. Theoretical

We consider the control of nonlinear systems in the discrete-time domain represented by

$$x(t+\Delta) = f(x(t),\, u(t)), \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state vector at time t, $u(t)$ is the vector of control inputs, $x(t+\Delta)$ is the successor state at time $t+\Delta$ with Δ denoting the sampling time, and $f: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is the (possibly nonlinear) state update function.

The objective of nonlinear model predictive control (NMPC) is to determine the sequence $\{u^*_0, \ldots, u^*_{N-1}\}$ of optimal control inputs over a fixed prediction horizon $N \in \mathbb{N}$ that bring the system in (1) from any admissible initial state $x(t)$ to a desired final state while minimizing a given performance index, and making sure that the control inputs, as well as the system states remain bounded by $x \in \mathcal{X}, u \in \mathcal{U}$ where $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$. Specifically, the optimal control sequence can be obtained by solving the following NMPC problem:

$$\min_{u_0,\ldots,u_{N-1}} \ell_N(x_N) + \sum_{j=0}^{N-1} \ell(x_j, u_j), \tag{2a}$$

$$\text{s.t.} \quad x_{j+1} = f(x_j, u_j),\ j = 0, \ldots, N-1, \tag{2b}$$

$$x_j \in \mathcal{X},\ j = 0, \ldots, N-1, \tag{2c}$$

$$u_j \in \mathcal{U},\ j = 0, \ldots, N-1, \tag{2d}$$

$$x_N \in \mathcal{T}, \tag{2e}$$

$$x_0 = x(t). \tag{2f}$$

Here, x_j and u_j represent, respectively, the state and input predictions at time step j of the prediction window, whose length is N (the prediction horizon). The predictions are initialized from the currently known measurement of the state $x(t)$, or its estimate. The performance index in (2a) involves two penalty functions: ℓ_N as the final penalty, and ℓ as the stage cost. A particular form of these two functions depends on the control objective. If, for instance, one aims at regulating the nonlinear system in (1) to a desired steady state, represented by the tuple (x_S, u_S), one can choose

$$\ell_N(x_N) = \| x_N - x_S \|^2_P, \tag{3}$$

and

$$\ell(x_j, u_j) = \| x_j - x_S \|^2_Q + \| u_j - u_S \|^2_R, \tag{4}$$

where $\| z \|^2_W = z^T W z$ is the weighted squared two-norm of a vector. Alternatively, the performance index (2) can account for economic aspects, such as the maximization of the yield of a chemical reaction, or to the minimization of input costs. Additional control objectives can be added to the stage cost (4). Frequently, one wants to make the control action smooth and not to fluctuate in time too much. This requirement can be accomplished by penalizing the slew rate of control inputs by adding the term $\| u_j - u_{j-1} \|^2_D$ to (4) with a suitable penalty matrix $\mathcal{D}$ and, for $j = 0$, $u_{-1} = u(t - \Delta)$ representing the control input from the previous optimization. The constraint sets in (2c)–(2e) typically involve min/max bounds on the states and inputs, but they can also entail additional requirements. For instance, one can enforce that the slew rate of control actions is bounded by adding the constraint $u_j - u_{j-1} \in \mathcal{D}$ to (2) for all steps of the prediction horizon, where the set consists of lower/upper bounds on the slew rate.

If the state-update equation $f(x, u)$ in (2b) is nonlinear, or if the constraint sets $\mathcal{X}, \mathcal{U}, \mathcal{T}, \mathcal{D}$ are non-convex, the NMPC problem (2) is a nonlinear optimization problem. Solving such a problem to its global optimum is, however, challenging and is likely to take a substantial amount of time. For a safe feedback implementation, however, we require that the optimal control sequence $\{u^*_0, ..., u^*_{N-1}\}$ is obtained within the duration of one sampling instant, i.e., Δ. The reason being that NMPC is implemented in a feedback arrangement using the receding horizon principle where only the first element of the sequence, i.e., u^*_0 is actually implemented to the system. Then, at the next sampling instant, new state measurement is conducted and the NMPC problem (2) is re-solved to obtain a new input sequence, from which again only the first element is used for control. The NMPC feedback law is thus $u(t) = u^*_0(x(t))$. If the time required to solve (2) is longer than the sampling time Δ, the feedback implementation can compromise safety of the system as it may lead to violation of constraints and even to instability. Therefore it is of imminent practical importance to be able to solve NMPC problems as in (2) fast enough. In practice, one even often sacrifices optimality in favor of obtaining a sub-optimal control sequence as long as such a sequence satisfies constraints in (2c)–(2e).

In this paper we propose to solve the NMPC problem (2) using the random shooting method, which belongs to the class of global stochastic optimization techniques. It is based on first generating a (possibly large) number of independent and identically distributed random control actions $u_0, ..., u_{N-1}$, followed by retaining only the sequences that satisfy the constraints in (2c)–(2e). To do that, (2b) is first used to obtain the state predictions $\{x_0, ..., x_N\}$ by simulating the system starting from $x_0 = x(t)$ and using the random control sequence $\{u_0, ..., u_{N-1}\}$. Once the predicted state trajectory is available, feasibility can be straight forwardly checked via (2c)–(2e). Finally, among the feasible sequences, the one that yields the smallest value of the performance index in (2a) is selected. The reason random shooting is well suited to solve NMPC problems (2) stems from its ability to exploit the structure of the problem and to check the satisfaction of the constraints step-by-step instead of having to check the whole sequence at once. Therefore infeasible candidates (i.e., those that do not satisfy the constraints in (2b)–(2e)) can be ruled out quickly with only low computational effort. This significantly reduces the overall runtime of the algorithm. Moreover, the random shooting method assumes no regularity conditions on the dynamics, constraints, and/or the performance index in (2a). The dynamics in (1) can therefore be arbitrary, e.g., discontinuous, non-differentiable, etc. Moreover, the constraints can be non-convex sets, including finite sets where the control inputs are either binary or integer. A price to be paid for such a generality is the sub-optimality of the method.

A pseudo-code of the proposed random shooting NMPC method is reported as Alg. 1. The algorithm performs at most N_f selections of the random input sequence using a while-loop. Each pass of the loop then selects a random control sequence, checks its feasibility over the whole prediction horizon using a for-loop, and updates the performance index using the stage cost (4). This is done step-by-step, aborting the for-loop prematurely if the control sequence is infeasible at some prediction step $j \in \{0, ..., N - 1\}$. Finally, if even the terminal set constraint in (2e) is satisfied, the performance index is updated with the terminal penalty in (3), and the cost of the current random input sequence is compared to the so-far best known random solution. The main advantage of Alg. 1 is twofold. First, each iteration only involves simple calculations, such as checking of the constraints in Step 6, updating the value of the performance index in Step 7, and the evaluation of the state-update equation $f(x, u)$ in Step 8; thus no costly optimization is involved. Second, infeasible random sequences can be ruled out quickly (cf. Steps 6 and 10) as constraints are checked step-by-step over the prediction window, further mitigating the runtime. Therefore the implementation of Alg. 1 is very simple and it runs fast.

Alg. 1 investigates a total of N_f random control sequences, returning the sequence with the lowest value of the performance index in (2a). We remark that, due to Steps 5, 6 and 13, the output generated by the algorithm is always feasible for the NMPC problem (2), despite being suboptimal. As a consequence, if the terminal set $\mathcal{T}$ in (2e) is chosen as a positively invariant set for the system in (1), Alg. 1 provides recursive feasibility guarantees. One such choice is to take $\mathcal{T} = \{x_s\}$, i.e., to use the desired steady-state as a terminal point constraint. Closed-loop stability can be achieved, for instance, by adding a contraction constraint of the form $\| x_{j+1} - x_s \| < \| x_j - x_s \|$ where $\| \cdot \|$ is any vector norm.

The procedure for selecting the random control input $u_j \in \mathcal{U}$ in Step 5 depends on the type of the input constraint set $\mathcal{U}$:

Case 1: If $\mathcal{U}$ is a finite set, i.e., $\mathcal{U} = \{u^{(1)}, ..., u^{(q)}\}$, then $u_j = u^{(r)}$ where r is a random integer from the interval $[1, q]$.

Case 2: If $\mathcal{U}$ is a hyperbox defined by its min/max bounds, i.e., $\mathcal{U} = \{u \mid u_{min} \le u \le u_{min}\}$, then u_j is generated by $u_{j,i} = u_{min,i} + \alpha_i(u_{max,i} - u_{min,i})$ for $i = 1, ..., m$ where $\alpha_i \sim U(0, 1)$ are uniformly distributed

random numbers from the interval [0, 1], and $u_{j,i}$ denotes the i-th coordinate of the vector $u_j \in \mathbb{R}^m$. Notice that hyperbox-type of input constraints are predominantly used in practice.

Case 3: If $\mathcal{U}$ is a convex polytope, two avenues can be taken – an iterative and a non-iterative one:

Case 3.1: The iterative procedure first constructs, off-line, the tightest outer hyperbox approximation $\overline{\mathcal{U}}$ of $\mathcal{U}$ by solving $2m$ linear programs (Suard et al., 2004). Then, in Step 5, random values $u_j \in \overline{\mathcal{U}}$ are generated by randomly sampling the hyperbox $\overline{\mathcal{U}}$ as discussed in Case 2 above until $u_j \in \mathcal{U}$ is verified to hold. Therefore multiple random selections might be required to guarantee that $u_j \in \mathcal{U}$.

Case 3.2: The non-iterative procedure constructs, off-line, the largest ball $\mathcal{B} = \{u \mid \| u - u_c \|_2 \leq r\}$ inscribed into $\mathcal{U}$, i.e., $\mathcal{B} \subseteq \mathcal{U}$. Note that such a ball can be constructed by solving a single linear program (Suard et al., 2004). Then $u_j = \alpha v + u_c$ where $\alpha_i \sim U(0, r)$ with r denoting the radius of the ball, and $n = [v_1, \ldots, v_m]^T$ with $v_i \sim U(-1, 1)$ for $i = 1, \ldots, m$ being a vector with components randomly distributed over the interval [–1, 1]. Clearly, $\alpha v + u_c \in \mathcal{B}$ and, since $\mathcal{B} \subseteq \mathcal{U}$, $u_j \in \mathcal{U}$ as a consequence.

Case 4: If $\mathcal{U}$ is a generic (possibly non-convex) set then one can proceed as in Case 3.1 by first constructing, off-line, the outer hyperbox approximation of $\mathcal{U}$ by solving $2m$ (possibly non-convex) optimization problems, followed by sampling the outer approximation and checking if $u_j \in \mathcal{U}$.

Naturally, the quality of the best feasible control sequence generated by the random shooting algorithm depends on the number N_f of random scenarios investigated. The more samples, the greater the chance of finding the global optimum is. If the NMPC (2) problem is convex, then one can provide a bound on the minimal number of feasible samples that provides a probabilistic bound on the suboptimality of the sequence $\{u_0, \ldots, u_{N-1}\}$ found by Alg. 1. The bound is due to (Vidyasagar, 2001; Tempo et al., 2012) and applies to random samples that are selected in Step 5 independently and identically distributed:

Lemma 2.1 *If the number of feasible samples satisfies*

$$N_f \geq \frac{log\left(1/\left(1-\delta/100\right)\right)}{log\left(1/\left(1-\alpha/100\right)\right)},$$

then the control sequence $\{u_0, \ldots, u_{N-1}\}$ *generated by Alg. 1 is by* α *per cent suboptimal with a confidence of* δ *per cent.*

As an example, if the sequence is to be, at most, $\alpha = 1$ % suboptimal with a confidence of $\delta = 99.9$ %, one would need to generate $N_f \geq 688$ random input sequences. Unfortunately, for generic non-convex problems, no such hard bound on the number of samples can be given. Nevertheless, Lemma 2.1 provides at least an indicator on the required number of samples.

It is also worth noting that random shooting in particular, and stochastic methods in general, are ill-suited to cope with NMPC problems with equality constraints, with two exceptions. First, linear equality constraints, such as $u_j = u_{j+1}$ which are often used as move-blocking constraints, can be easily eliminated from the problem by projecting them onto their respective null spaces (Boyd and

Algorithm 1 NMPC via Random Shooting

INPUT: Current state measurement $x(t)$, maximal number N_f of feasible sequences to check.

OUTPUT: Sub-optimal, but feasible control sequence $\{u_0, \ldots, u_{N-1}\}$.

1. Initialization: $J_{best} = \infty$, $U_{best} = \varnothing$, $n_f = 0$
2. **while** $n_f < N_f$ do
3. Set $x_0 = x(t)$, J = 0, and feasible = true.
4. **for** $j = 0, \ldots, N - 1$ **do**
5. Pick a random control input $u_j \in \mathcal{U}$.
6. **if** $x_j \in \mathcal{X}$ (and possibly $u_j - u_{j-1} \in \mathcal{D}$) **then**
7. Update $J = J + \ell(x_j, u_j)$.
8. Calculate $x_{j+1} = f(x_j, u_j)$.
9. **else**
10. Set feasible = false and **break**.
11. **end if**
12. **end for**
13. **if** *feasible* = *true* and $x_N \in \mathcal{T}$ **then**
14. Update $J = J + \ell_N(x_N)$.
15. Increment $n_f = n_f + 1$.
16. **if** $J < J_{best}$ **then**
17. Update $J_{best} = J$ and store $U_{best} = \{u_0, \ldots, u_{N-1}\}$ as the so-far best solution.
18. **end if**
19. **end if**
20. **end while**
21. **return** U_{best} as the best feasible control sequence.

Vandenberghe, 2004). The second exception are nonlinear equality constraints due to the prediction equation in (2b). Note that in the proposed random shooting method (2b) is not treated as a constraint. Instead, it is used to directly calculate the state predictions $\{x_0, ..., x_N\}$ in Step 8. Therefore the only practical limitation for Alg. 1 to work properly is that the constraint sets $\mathcal{X}, \mathcal{U}, \mathcal{T}$ are either fully dimensional, or they contain just linear equalities.

3. Experimental

We consider a continuous stirred tank reactor (CSTR) in which a fast, isothermal, liquid phase, multi-component chemical reaction $A \leftrightarrows 2C \rightarrow B$ takes place (Fissore, 2008). The continous-time nonlinear dynamical model of the reactor is given by

$$\dot{c}_{\mathrm{A}} = -k_1 c_{\mathrm{A}} + \frac{F}{V}\left(c_{A,\mathrm{feed}} - c_{\mathrm{A}}\right) + k_2 c_{\mathrm{C}}^2, \tag{10}$$

$$\dot{c}_{\mathrm{B}} = -\frac{F}{V} c_{\mathrm{B}} + k_3 c_{\mathrm{C}}^2, \tag{11}$$

$$\dot{c}_{\mathrm{C}} = k_1 c_{\mathrm{A}} - \frac{F}{V} c_{\mathrm{C}} - \left(k_2 + k_3\right) c_{\mathrm{C}}^2 + u, \tag{12}$$

with the state vector $x = [c_{\mathrm{A}}, c_{\mathrm{B}}, c_{\mathrm{C}}]^{\mathrm{T}}$ consisting of the concentrations. The objective is to keep the concentration of all compounds on their respective steady-state levels by manipulating the molar feed rate of component *C*, represented by the input signal *u*. The model parameters, as reported in (Fissore, 2008), are: $k_1 = 1.0$ m^3 mol^{-1} s^{-1}, $k_2 = 3.0$ m^3 mol^{-1} s^{-1}, $k_3 = 5.0$ m^3 mol^{-1} s^{-1}, $F = 3$ m^3 s^{-1}, $V = 3$ m^3 and $c_{\mathrm{A,feed}} = 2$ mol m^{-3}. The desired steady state is given by $c_{\mathrm{A,S}} = 2.18$ mol m^{-3}, $c_{\mathrm{B,S}} = 3.93$ mol m^{-3}, $c_{\mathrm{C,S}} = 0.87$ mol m^{-3} and $u_{\mathrm{S}} = 5$ mol s^{-1}.

First, the system in (5) was discretized using a sampling time of $\Delta = 0.1$ s using a forward Euler discretization. Subsequently, the NMPC problem (2) was formulated with prediction horizon $N = 5$, and penalty matrices $P = 100 \cdot I_{3\times3}$, $Q = 10 \cdot I_{3\times3}$, $R = 0.05$ in (3) and (4). Moreover, a slew-rate penalty of the form $\| u_j - u_{j-1} \|_D^2$ with $D = 2$ was also added to the stage cost to achieve a smooth control action that avoids abrupt changes. Control inputs were constrained by $3.5 \leq u \leq 6.5$ mol s^{-1}. To amplify the ability of NMPC to deal with constraints, we have also added a slew-rate constraint of the form $-1 \leq u(t) - u(t-\Delta) \leq 1$, which can be considered as a rate limit of the actuator. Finally, all states are required to be non-negative, i.e., $c_{\mathrm{A}} \geq 0$, $c_{\mathrm{B}} \geq 0$, $c_{\mathrm{C}} \geq 0$. To make the problem more challenging, we have also included an artificial constraint $0.75 \leq c_{\mathrm{C}} \leq 1.00$ mol m^{-3}.

4. Results and discussion

To illustrate the performance of the random shooting method, represented by Alg. 1, we have conducted a closed-loop simulation where the states were initialized to their respective steady-state inputs. Then, at time instants $t = 0.5$ and $t = 4$ seconds, disturbances of 2 mol m^3 and –1.5 mol m^3 were added to the second state, respectively. Two simulations were performed, one where the control inputs were selected by solving the NMPC problem (2) to its global optimum at every sampling instant using the global solver BARON (which is one of the best commercially available nonlinear solver), and the other one where the inputs were determined by Alg. 1 with $N_{\mathrm{f}} = 1000$ random control sequences.

The time profiles of the CSTR are shown in Fig. 1. As can be observed, the differences between the optimal solution and the sub-optimal control actions calculated by Alg. 1 are negligible. Moreover, as can be seen from Figs. 1(c) and 1(d), even the random shooting-based NMPC controller provides satisfaction of hard constraints on system states, control inputs, as well as on its slew rate, defined as $u(t) - u(t-\Delta)$. We remark that the state constraints in Fig. 1(c) were not hit because the control action is already saturated, as can be seen in Fig. 1(d).

It took 270 milliseconds on average to solve the NMPC problem using BARON to global optimality, exceeding the sampling time $\Delta = 0.1$ seconds. Therefore, optimal NMPC would not be suitable for a real-time implementation. Alg. 1, on the other hand, only took 0.4 milliseconds, on average. Therefore, the proposed random shooting approach is 675 times faster than optimal NMPC and is thus well suited for real-time NMPC. All computations were done on a 3.5 GHz CPU with 16 GB of RAM using Matlab 2017b.

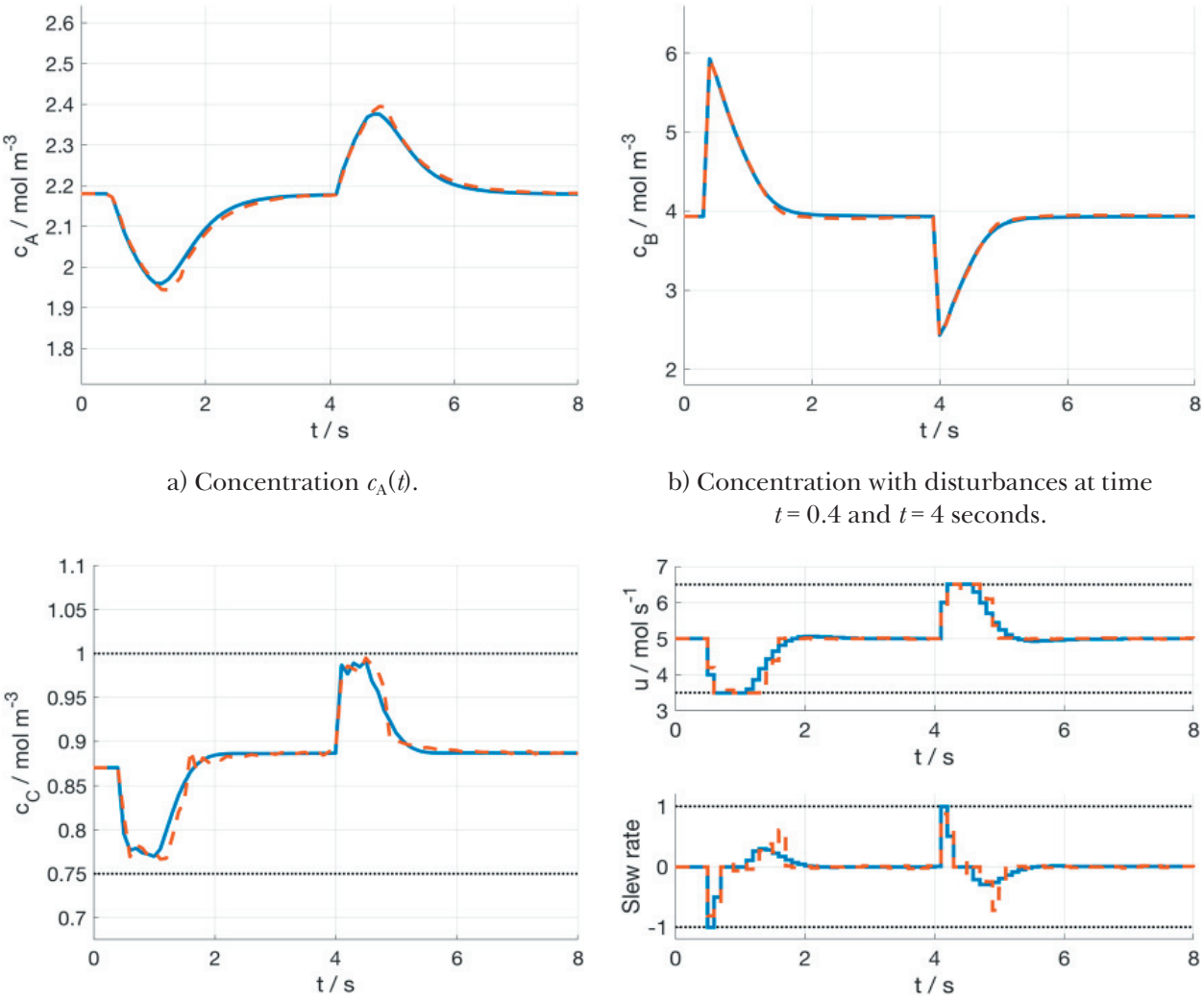
Naturally, the control inputs generated by Alg. 1 are sub-optimal. However, as can be seen in Fig. 1, the sub-optimality is low. To quantify the loss of optimality, we have calculated the discrete-time version of the integrated squared error (ISE) criterion defined as

$$\sum_{t=0}^{t_{\mathrm{sim}}} \| x(t) - x_{\mathrm{S}} \|_2^2,$$

where $t_{\mathrm{sim}} = 8$ s is the length of the simulation window, $x(t)$ is the state vector at time instant t, and x_{S} is the desired steady-state of the reactor. Moreover, the overal consumption of the input feed, given in mol, was computed by

$$\sum_{t=0}^{t_{\mathrm{sim}}} \Delta\, u(t),$$

where Δ is the sampling time. Judging by the ISE



a) Concentration $c_A(t)$.

b) Concentration with disturbances at time $t = 0.4$ and $t = 4$ seconds.

c) Concentration $c_C(t)$ with associated constraints.

d) Control input $u(t)$ and its slew rate $u(t) - u(t - \Delta)$, together with respective constraints.

Fig. 1. Time profiles of concentrations c_A, c_B, c_C and the control input u for the CSTR example. Solid lines represent profiles under the optimal NMPC feedback law. Dashed lines are profiles under the sub-optimal random-shooting approach of Alg. 1. Dotted black lines represent constraints. We remind that the steady-state values to which the CSTR should be controlled are $c_{A,S} = 2.18$ mol m^{-3}, $c_{B,S} = 3.39$ mol m^{-3}, $c_{C,S} = 0.87$ mol m^{-3}, $u_S = 5$ mol s^{-1}.

criterion, the random-shooting NMPC controller was worse just by 1.5 % compared to the globally optimal NMPC solution. The total consumption of the input stream was 40.2456 mol for the optimal NMPC controller versus 40.3412 mol for the random-shooting approach, a sub-optimality of just 0.3 %. We remark that sub-optimality can be further reduced by increasing the number of random samples N_f in Alg. 1 at the expense of an increased runtime. In our experience, the runtime scales linearly with N_f.

5. Conclusions

We have proposed to solve non-convex nonlinear model predictive control problems using a random shooting method where control inputs are selected at random. Despite the control inputs being sub-optimal, the algorithm guarantees that they are at least feasible, thus maintain recursive feasibility and closed-loop stability. The procedure explicitly exploits the specific structure of the NMPC problem in (2) as to simplify the calculations. Specifically, the algorithm only involves forward-in-time simulations of the nonlinear dynamics, followed by checking constraints and calculating the value of the performance index. Therefore the procedure is not just fast, but very versatile as well as it assumes no regularity conditions. By means of a motivating case study we have illustrated that the random shooting approach to NMPC indeed offers a significant reduction of the calculation time while

featuring just a small sub-optimality. Therefore it is well suited to implement nonlinear MPC in real time.

Acknowledgements

The Authors gratefully acknowledge the contribution of the Slovak Research and Development Agency under the project APVV 15-0007 and the contribution of the Scientific Grant Agency of the Slovak Republic under the grants 1/0112/16 and 1/0403/15, and the Research & Development Operational Programme for the project University Scientific Park STU in Bratislava, ITMS 26240220084, supported by the Research 7 Development Operational Programme funded by the ERDF.

6. References

Allgöwer F, Zheng A (2012) Nonlinear model predictive control. Vol. 26. Birkhäuser.

Bakošová M, Oravec J, Matejičková K (2013) Model Predictive Control-Based Robust Stabilization of a Chemical Reactor. Chemical Papers, vol. 67, no. 9, pp. 1146—1156.

Bakošová M, Oravec J (2014) Robust MPC of an Unstable Chemical Reactor Using the Nominal System Optimization. Acta Chimica Slovaca, vol. 7, no. 2, pp. 87—93.

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge university press.

Čižniar M, Podmajerský M, Hirmajer T, Fikar M, Latifi MA (2009) Global optimization for parameter estimation of differential-algebraic systems. Chemical Papers, vol. 63, no. 3, pp. 274—283.

Dyer M, Kannan R, Stougie L (2014) A simple randomised algorithm for convex optimisation. Mathematical Programming, vol. 147, no. 1—2, pp. 207—229.

Fissore D (2008) Robust control in presence of parametric uncertainties: observer-based feedback controller design. Chemical Engineering Science, vol. 63, no. 7, pp. 1890—1900.

Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. Science, vol. 220, no. 4598, pp. 671—680.

Maciejowski J (2002) Predictive control: with constraints. Pearson education.

Martin P, Odloak D, Kassab F (2013) Robust model predictive control of a pilot plant distillation column. Control Engineering Practice, vol. 21, no. 3, pp. 231—241.

Oravec J, Bakošová M, Mészáros A, Míková N (2016) Experimental Investigation of Alternative Robust Model Predictive Control of a Heat Exchanger. Applied Thermal Engineering, vol. 105, pp. 774—782.

Oravec J, Bakošová M, Trafczynski M, Vasičkaninová A, Mészáros A, Markowski M (2018) Robust model predictive control and PID control of shell-and-tube heat exchangers. Energy, vol. 159, pp. 1—10.

Oravec J, Bakošová M (2012) Robust Constrained MPC Stabilization of a CSTR. Acta Chimica Slovaca, vol. 5, no. 2, pp. 153—158.

Oravec J, Bakošová M (2015) Robust Model-Based Predictive Control of Exothermic Chemical Reactor. Chemical Papers, vol. 69, no. 7.

Piovesan J, Tanner H (2009) Randomized model predictive control for robot navigation. In: IEEE International Conference on Robotics and Automation, pp. 94—99.

Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. Swarm intelligence, vol. 1, no. 1, pp. 33—57.

Qin S, Badgwell T (2003) A survey of industrial model predictive control technology. Control engineering practice, vol. 11, no. 7, pp. 733—764.

Sahoo S, Lampert C, Martius G (2018) Learning Equations for Extrapolation and Control. arXiv preprint arXiv:1806.07259.

Suard R, Lofberg J, Grieder P, Kvasnica M, Morari M (2004) Efficient computation of controller partitions in multi-parametric programming. In: Decision and Control, 2004. CDC. 43rd IEEE Conference on. Vol. 4, Citeseer, pp. 3643—3648.

Tempo R, Calafiore G, Dabbene F (2012) Randomized algorithms for analysis and control of uncertain systems: with applications. Springer Science & Business Media.

Vidyasagar M (2001) Randomized algorithms for robust controller synthesis using statistical learning theory. Automatica, vol. 37, no. 10, pp. 1515—1528.

Wright S, Nocedal J (1999) Numerical optimization. Springer Science, vol. 35, no. 67—68, p. 7.