École polytechnique fédérale de Lausanne

Bachelor Project

Bachelor in Mathematics

# Study of Relaxed Recentered Log-Barrier function based Nonlinear Model Predictive Control

*Author:*
Tudor Oancea

*Supervisors:*
Nicolas Boumal
Colin Jones

**Abstract**

In this project, we investigate the use of relaxed recentered logarithmic barrier functions in the context of Nonlinear Model Predictive Control (Nonlinear MPC or NMPC). These functions are a variation of the regular log-barrier functions that are introduced in the objective function of an optimization problem as a penalty on the deviation from the constraints set. The resulting new MPC scheme has been studied in the case of linear dynamics, and several interesting results on the global nominal asymptotic stability of the corresponding closed-loop system and on constraint satisfaction guarantees have been obtained. Extending them to the case of nonlinear dynamics is a non trivial task and we show in this project that these properties can still hold, but only locally. The theoretical results are complemented by numerical illustrations based on the Real Time Iteration method.

# Contents

# 1 Introduction

*Control theory* can simply be defined as the study of a dynamical system whose state evolution can be influenced by the means of external parameters called *controls*, or *inputs*. This theory is of major importance in numerous fields of engineering such as robotics, mechanical or chemical engineering where problems such as stabilizing a chemical reaction at a certain temperature or making an autonomous car follow the road are common.

An important subfield of control theory is *optimal control*, which attempts to find a *control law*, i.e. a policy used to control a system in order to accomplish a certain goal, by formulating and solving an optimization problem. For example, if one wants to stabilize the state around a reference point that is constant in time, one will want to choose a control law that will minimize the distance between the future state (that depends on the current state and the chosen control) and the target state. This problem of *stabilization* will be the one we will consider in this paper, but it is interesting to know there are other problems for which optimal control is applicable, such as path tracking or optimal trajectory generation.

One of the most important algorithm of optimal control is *Model Predictive Control*, or MPC. This procedure uses the knowledge of the system's *model* (the equation governing its evolution) to compute a control law that also takes the future into account. More precisely, the control law is computed for an interval in time called the *horizon*, and the cost along this whole horizon is minimized. In some sense, the controller tries not to take decisions it will regret afterwards. MPC has many advantages over other classical control strategies, such as the ability to handle several control parameters and to take into consideration constraints on the states and the controls (by solving a *constrained* optimization problem). These are very important in practice because any real-life system has physical limitations (e.g. the maximum speed of a car, the maximum torque of a motor, etc.). We will properly introduce the mathematical formulation of MPC and the theoretical tools used to study it in section 2.

In section 3 we will introduce a different formulation in which (most of) the state and control constraints are replaced with a penalty added to the cost function in the form of a log-barrier function, which are common in Interior Point methods. These functions have several drawbacks, such as the fact that they are only defined in the interior of the feasible set. In this paper we will introduce a relaxed version of these functions called *Relaxed Recentered Log-Barrier functions*, or *RRLB functions*. They extend regular log-barrier functions to the whole space and yield an equality-only-constrained optimization problem, which should in general be easier to solve.

In the theoretical study of this new formulation that we will call *RRLB MPC*, we will prove two major results:

1. When using the control law computed by the RRLB MPC, the corresponding closed-loop system asymptotically stabilizes at the target state (see subsection 4.1). This property will be called *nominal stability*, to be differentiated of *real-time stability* (see subsection 5.2).

2. In this formulation, the optimal solution of the optimization problem might yield states or controls that are not feasible in the original problem, since the constraints are not strictly enforced. However, if we start sufficiently close to the target state, this will never happen (see subsection 4.2).

Finally, in section 5 we examine numerical aspects of the RRLB MPC. We will introduce a common strategy to solve NMPC problems called *Real Time Iteration* (or RTI) and discuss its advantages and drawbacks as for the tradeoff between computational efficiency and accuracy. A simple Python implementation of this method will be used to verify our theoretical results and to compare the performance of RRLB MPC and regular MPC (eg in terms of runtime and in terms of convergence speed). The system of choice will be a popular benchmark problem called *Continuously Stirred Tank Reactor* (or CSTR).

# 2 Background material

This whole section constitutes classical knowledge on MPC and dynamical system and can references can be found in textbooks such as [RMD17] (chapters 1 and 2 on MPC and Appendix B on stability theory) or [LG17] (chapter 3 on NMPC and chapter 4 on infinite horizon MPC).

## 2.1 What is MPC ?

We consider a discrete-time controlled nonlinear dynamical system of the form $x(k+1) = f(x(k), u(k))$ (which we will usually write as $x^+ = f(x,u)$) with state and control constraints $x(k) \in \mathcal{X} \subseteq \mathbb{R}^{n_x},\ u(k) \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$.

In practice, the systems of interest are almost always continuous. We therefore assume that we will only change the control variables at instants evenly spaced in time, and we discretize the dynamics using ODE integrators such as RK4. The time between each two consecutive control updates is called the *sampling time*.

Our goal is to stabilize the system around a target state $x^*$ and a target control $u^*$, which have to be a fixed point of the system: $x^* = f(x^*, u^*)$. To do that we will try to construct a *state-feedback control law*, i.e. a function $\mu : \mathcal{X} \to \mathcal{U}$ such that $x^*$ is *asymptotically stable* for the dynamical system $x^+ = f(x, \mu(x))$ (which only depends on the state). The exact definition of asymptotical stability and how to prove it is discussed in subsection 2.2.

An important simplification that is often made (and that we will use in this paper) is that we consider $x^* = 0$ and $u^* = 0$. It is easy to verify that when this is not the case, we can just define new translated states $\hat{x} = x - x^*$ and new control $\hat{u} = u - u^*$ and consequently new dynamics, new constraints, etc.
To construct this control law $\mu$, we will solve the following optimization problem:

$$\begin{aligned} V_N(x) = \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} l(x_k, u_k) \ + \ F(x_N) \\ \text{s.t.} \quad & x_0 = x \text{ and } x_{k+1} = f(x_k, u_k),\ k = 0, \dots, N-1 \\ & x_k \in \mathcal{X},\ k = 0, \dots, N \\ & u_k \in \mathcal{U},\ k = 0, \dots, N-1 \end{aligned} \tag{1}$$

where $N \in \mathbb{N}$ is called the *horizon length*, the function $l : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the *stage cost*, the function $F : \mathcal{X} \to \mathbb{R}$ is the *terminal cost* (its role will become clearer in next section) and the function $J_N : \mathcal{X}^{N+1} \times \mathcal{U}^N \to \mathbb{R}, (\mathbf{x}, \mathbf{u}) \mapsto \sum_{k=0}^{N-1} l(x_k, u_k) \ + \ F(x_N)$ is called the *cost function*.

These costs have to be defined by the user as a measure of the deviations from the target state and control (we will always assume that $l(x^*, u^*) = l(0,0) = 0$). Their fine tuning is of capital importance for the feasibility, robustness and performance of the MPC scheme, and it is a whole area of interest of control theory (which is unfortunately out of the scope of this paper). Then, during the experiment, we observe at a time $k$ the state $x(k)$, we solve the MPC 1 and find the optimal sequences of states and controls

$$\mathbf{x}^*(x(k)) = \{x_0^*(x(k)) = x(k), \dots, x_N^*(x(k))\}, \quad \mathbf{u}^*(x(k)) = \left\{u_0^*(x(k)), \dots, u_{N-1}^*(x(k))\right\}$$

They represent the controls to apply at time $k, k+1, \dots, k+N-1$ and the states the system will be in after these controls are applied. In a sense, we 'over-solve' the problem since we are actually only interested in $u_0^*(x(k))$ that will define our control law:

$$\mu_{\text{MPC}}(x(k)) := u_0^*(x(k))$$

Since at time $k+1$ the horizon will have shifted, we say we solve the control problem in a *receding horizon* fashion.

Other variants of MPC exist in which different state constraint, called *terminal constraints*, are enforced on $x_N$ . We are also going to consider very specific forms of costs and constraints: the costs will always be quadratic ($l(x,u) = x^TQx + u^TRu$ and $F(x) = x^TPx$ for some symmetric and positive definite matrices $Q, R$ and $P$) and the (inequality) constraints will always be polytopic (of the form $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid C_x x \leq d_x\}$ , $\mathcal{U} = \{u \in \mathbb{R}^{n_u} \mid C_u u \leq d_u\}$ for matrices $C_x, C_u$ and vectors $d_x, d_u$ of appropriate dimension such that the sets are bounded). These are very common in real-world applications.

Other variants of MPC are:

- the *infinite horizon MPC*:

$$\begin{aligned} V_\infty(x) = \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{\infty} l(x_k, u_k) \\ \text{s.t.} \quad & x_0 = x \text{ and } x_{k+1} = f(x_k, u_k),\ k = 0, \dots, N-1 \\ & x_k \in \mathcal{X},\ k = 0, \dots, N \\ & u_k \in \mathcal{U},\ k = 0, \dots, N-1 \end{aligned}$$

  In real life applications, we consider that the duration of the experiment is virtually infinite, so this variant may seem more natural. It is usually not used in practice as the optimization problem becomes infinite-dimensional, which is in general computationally intractable.

  However, it enjoys several theoretical properties such as *Bellman's principle of optimality*(see [Bel57]) that can be defined in the following way:

  > *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

  or in mathematical terms:

$$V_\infty(x) = l(x, u_0^*(x)) + V_\infty(x, f(x, u_0^*(x))) = \min_{u \in \mathcal{U}} \ l(x,u) + V_\infty(x, f(x,u)) \tag{2}$$

  This recursive minimization problem forms the basis of what we call *Dynamic Programming*, and relies on the fact that the horizon is infinite. In the finite horizon case (without terminal costs) a weaker version reads:

$$V_N(x) = \min_{u \in \mathcal{U}} \ l(x,u) + V_{N-1}(x, f(x,u))$$

  The terminal costs in regular MPC can actually be interpreted as a proxy for the part of the infinite horizon that is not considered and are present to bridge the gap between finite horizon performance and infinite horizon optimality. As it is illustrated in the next point, in some cases $V_\infty$ can actually be computed exactly.

- the *(infinite-horizon) Linear Quadratic Regulator*, or *LQR*:
  which is basically an MPC with linear dynamics ($x^+ = Ax + Bu$) and without state and

control constraints. These problem is the most basic one we can find and is deeply linked to the *(Discrete) Algebraic Riccati Equations*, or *DARE*:

$$P = Q + A^T PA - A^T PB(R + B^T PB)^{-1} B^T PA \tag{3}$$

This equation derived from the Bellman's principle of optimality evoked above allows us to solve the dynamic programming problem 2 explicitly: the unique positive definite solution $P$ of this equation verifies $V_\infty(x) = x^T Px$. The proof of the existence of this unique solution belongs to the classical control theory, is not strictly necessary to understand this project and is therefore omitted.

The DARE theory also provides the state-feedback control law of the infinite horizon LQR: $\mu_{\text{LQR}}(x) = Kx$ where $K = -(R + B^T PB)^{-1} B^T PA$. It is interesting to notice that in this case, the solution $P$ to the DARE also verifies the following equation called *Lyapunov equation*:

$$P = A_K^T PA_K + Q_K$$

where $A_K := A + BK$ corresponds the evolution of the controlled system since $x^+ = Ax + B\mu(x) = (A + BK)x$, and $Q_K := Q + K^T RK$ corresponds to cost of a stage $(x, Kx)$ since $l(x, Kx) = x^T Q + (Kx)^T R(Kx) = x^T Q_K x$.

All the derivation of the previous facts can be found in the appendix 7.2. In most cases, the terminal costs of a finite horizon MPC are constructed by solving a variation of the DARE that stems from a LQR approximating the original M¨PC.

## 2.2 Lyapunov stability theory

In this subsection we present the necessary background in dynamical systems theory and in particular give more precise definitions of the stability properties we want to prove. We will consider a general nonlinear dynamical system $x^+ = g(x)$, without any control, since we remember that our goal is to construct the control as only depending on the state.
A state $x^*$ is said to be:

- *locally stable* if $\forall \epsilon > 0, \exists \delta > 0$ s.t. $\|x(0) - x^*\| \leq \delta \implies \|x(k) - x^*\| \leq \epsilon, \ \forall k \geq 0$ or in other words if when the system starts close to $x^*$, it always stays close to it.
- *locally attractive* if $\|x(k) - x^*\| \underset{k\to\infty}{\longrightarrow} 0$ for all $x(0)$ in a neighborhood of $x^*$ or in other words if when the system starts close to $x^*$, it will converge to it.
- *globally attractive* if $\|x(k) - x^*\| \underset{k\to\infty}{\longrightarrow} 0$ for all $x(0) \in \mathcal{X}$ or in other words that the system will converge to it no matter the initial state.
- *locally asymptotically stable*, or *LAS*, if it is both locally stable and locally attractive.
- *globally asymptotically stable*, or *GAS*, if it is both stable and globally attractive.

It can be showed that a state $x^*$ is GAS (respectively LAS), if we can find a function $V : \mathcal{X} \to \mathbb{R}$ and constants $c_1, c_2, c_3 > 0$ such that $\forall x \in \mathcal{X}$ (respectively in a neighborhood of $x^*$):

$$\begin{aligned} V(x) &\leq c_1 \|x - x^*\|^2 \\ V(x) &\geq c_2 \|x - x^*\|^2 \\ V(g(x)) &\leq V(x) - c_3 \|x - x^*\|^2 \end{aligned}$$

Such a function is called a *Lyapunov function*, and the last equation is called the *descent property*, which is usually the one that is the harder to prove for a given candidate function.

When considering a system controlled by MPC as described in section 2.1, the candidate function of choice will be the objective value function $V_N$ defined in 1. General conditions that ensure this property can be found in chapter 2 section 4 of [RMD17].

# 3 The RRLB MPC

In this section we will introduce the RRLB functions, formulate the RRLB MPC and recall the results proved in the linear case in [FE17]. As mentioned in the last section, we will suppose that $x^* = 0 \in \mathcal{X}^\circ$ and $u^* = 0 \in \mathcal{U}^\circ$.

Before introducing the RRLB MPC, let's first state the principal problem it solves. In practice, it might happen than the constraints are violated because of external disturbances or of the presence of *model mismatch*, i.e. the system's evolution is not perfectly described by the dynamics used in the MPC. In this case, the NLP defining the MPC 1 becomes unfeasible and the algorithm implementing it will either crash or return solutions that are not optimal at all. A common way to fix this problem was to use soft constraints on the state and control constraints, by introducing slack parameters that are penalized in the objective function. Such approaches were reported for example in [KM00]. The problem with this soft-constrained formulation is the incapacity of providing closed-loop properties like asymptotic stability or constraints satisfaction, unless nonsmooth penalty functions are used, which dramatically impacts the algorithms in charge of solving the NLP. RRLB functions are designed to overcome this limitations while still preserving the recursive feasibility properties.

Let's first take a step back and recall what regular log-barrier functions are. We will only give the definitions for the states $x \in \mathbb{R}^{n_x}$, since they only depend on the constraint set and not on the actual quantity represented.

For a simple constraint of the form $c^T x \leq d$, the associated *log-barrier function* is defined as $-\log(d - c^T x)$. Such a function is defined on the interior of feasible set of the constraint and becomes infinity near its boundary. For a set of polytopic constraints $\{x \mid C_x x \leq d_x\}$, we can define the log-barrier as the sum of the log-barriers for each constraint:

$$B_x(x) = \sum_{i=1}^{q_x} -\log(d_{x,i} - \mathrm{row}_i(C_x)x)$$

where $q_x$ is the number of constraints, or equivalently the numbers of rows in $C_x$ .

A first interesting property we would like to add to our log-barrier function is the one of *positive definiteness*, i.e. $B_x(x) > 0,\ \forall x \neq 0$ in a neighborhood of the origin. To this end we introduce the notion of *weight recentered log-barrier function* as defined in [FE15]:

$$B_x(x) = \sum_{i=1}^{q_x} (1 + w_{x,i}) \left[\log(d_{x,i}) - \log(d_{x,i} - \mathrm{row}_i(C_x)x)\right]$$

where the weights $w_{x,i}$ are defined as chosen such that $B_x(0) = 0$ and $\nabla B_x(0) = 0$. It is shown in [FE15] that such weights always exist when the constraint set is polytopic (i.e. polyhedral and bounded) and that this wight-recentered log-barrier function is also positive definite and upper-bounded by a quadratic.

The second important property we have to add is the definiteness on the whole space. For a given *relaxation parameter* $\delta > 0$, we can define the *relaxed recentered log-barrier function*, or

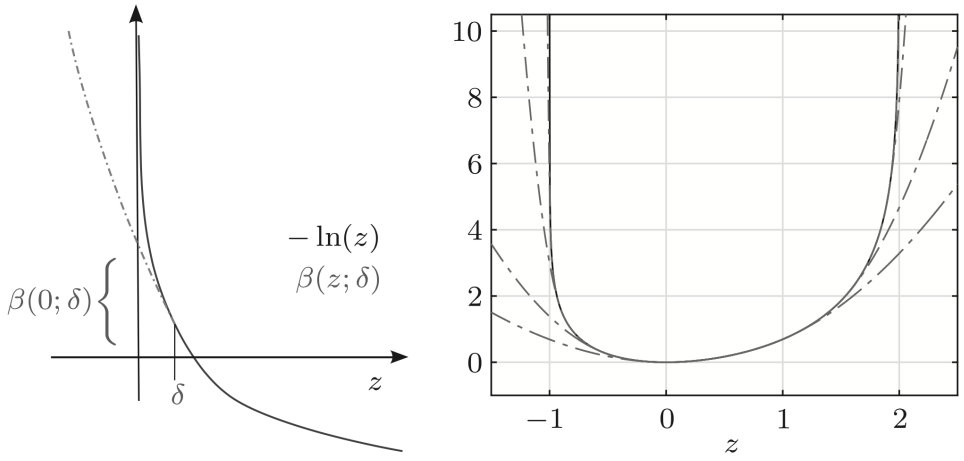



Figure 1: *(Left)*: Principle of relaxed log barrier function based on quadratic relaxation. (*Right*): Regular weight recentered log barrier function (solid line) and RRLB function for $\delta \in \{0.01, 0.1, 0.5, 1\}$ for the constraint $z \in \mathbb{R},\ -1 \leq z \leq 2$

*RRLB* function as follows:

$$\begin{aligned} B_x(x) &= \sum_{i=1}^{q_x} (1 + w_{x,i}) B_{x,i}(x) \\ \text{with } B_{x,i}(x) &= \begin{cases} \log(d_{x,i}) - \log(d_{x,i} - \text{row}_i(C_x)x) & \text{if } d_{x,i} - \text{row}_i(C_x)x > \delta \\ \beta(d_{x,i} - \text{row}_i(C_x)x; \delta) & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

where $\beta$ is a twice continuously function that extends the log-barrier to a twice continuously differentiable function on $\mathbb{R}$. The simplest example of such a function is

$$\beta(z;\delta) = \frac{1}{2}\left[\left(\frac{z - 2\delta}{\delta}\right)^2 - 1\right] - \log(\delta)$$

To ensure that this function still verifies $B_x(0) = 0$ and $\nabla B_x(0) = 0$, we need to choose $0 < \delta \leq \min\{d_{x,1}, \ldots, d_{x,q_x}, d_{u,1}, \ldots, d_{u,q_u}\}$, which is always possible if $0 \in \mathcal{X}$ and $0 \in \mathcal{U}$. An illustration of this relaxation procedure can be found in figure 1 found in [DBS05].

With all these definitions we can finally define the *RRLB MPC*:

$$\begin{aligned} \tilde{V}_N(x) = \min_{\mathbf{x},\mathbf{u}} \quad & \tilde{J}_N(\mathbf{x}, \mathbf{u}) \\ \text{s.t.} \quad & x_0 = x \text{ and } x_{k+1} = f(x_k, u_k),\ k = 0, \ldots, N-1 \end{aligned} \tag{5}$$

where the new objective function $\tilde{J}_N(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{N-1} \tilde{l}(x_k, u_k) + \tilde{F}(x_N)$ is defined using the new stage costs $\tilde{l}(x,u) = l(x,u) + \epsilon B_x(x) + \epsilon B_u(u)$ and the new terminal cost $\tilde{F}(x) = x^T P x$ for another matrix $P$ that will be determined in such a way that the control law given by the MPC yields an asymptotically stable system. The barrier parameter $\epsilon$ has in theory the following interpretation: when it goes to zero, the solution of problem 5 converges to the one of 1. However, we will here consider it as fixed for simplicity.

When we consider the RRLB MPC 5 in the case of linear dynamics $x^+ = Ax + Bu$, we know the two following theorems.

**Theorem 3.1** (Theorem 5 in [FE17]). *Suppose that the pair $(A, B)$ is stabilizable (the definition can be found in appendix 7.1) and that the matrix $P$ is chosen as the unique positive definite solution to the following modified DARE:*

$$P = A^T PA - A^T PB(R + \epsilon M_u + B^T PB)^{-1} B^T PA + Q + \epsilon M_x$$

*where $M_x = \nabla^2 B_x(0)$ and $M_u = \nabla^2 B_u(0)$ are the hessian of the RRLB functions at the origin.*

*Then for any initial state $x(0) \in \mathbb{R}^{n_x}$, the control law given by the MPC yields an asymptotically stable system.*

**Theorem 3.2** (Lemma 5 in [FE17]). *In the same conditions as in the previous theorem, there is a neighborhood $\mathcal{X}_N(\delta)$ of the origin such that for any initial state $x(0) \in \mathcal{X}_N(\delta)$, all the constraint will be satisfied along the closed-loop trajectories. Furthermore, the set $\mathcal{X}_N(\delta)$ is given explicitly by:*

$$\mathcal{X}_N(\delta) = \left\{ x \in \mathcal{X} \mid \tilde{V}_N(x(0)) - x(0)^T P_{\text{LQR}} x(0) \leq \min\{\beta_x(\delta), \beta_u(\delta)\} \right\}$$

*where $P_{\text{LQR}}$ is the solution to the classical DARE 3 and*

$$\beta_x(\delta) = \min_{i,x} \{B_x(x) \mid \text{row}_i(C_x x) = d_{x,i}\}, \quad \beta_u(\delta) = \min_{i,u} \{B_u(u) \mid \text{row}_i(C_u u) = d_{u,i}\}$$

Additional results were proven in the linear case, such as the existence of a procedure to find $\delta$ such that the maximum constraint violation along the closed-loop trajectories is bounded by a pre-defined tolerance. In the nonlinear case, however, we will not be able to prove such powerful results because of the lack of knowledge on the error terms that appear in the dynamics (see subsection 4.2).

# 4 Theoretical properties of RRLB Nonlinear MPC

In this section, we consider the RRLB MPC 5 with nonlinear dynamics and show that the results of the previous section hold locally, in a neighborhood of the origin.

From this point on, for an initial state $x \in \mathbb{R}^{n_x}$ we will denote by $\tilde{\mathbf{x}}(x) = \{\tilde{x}_0(x) = x, \tilde{x}_1(x), \ldots, \tilde{x}_N(x)\}$ and $\tilde{\mathbf{u}}(x) = \{\tilde{u}_0(x), \tilde{u}_1(x), \ldots, \tilde{u}_{N-1}(x)\}$ the optimal sequences of states and controls found by the RRLB MPC. When no confusion is possible we will drop the "$(x)$".

## 4.1 Nominal asymptotic stability

**Lemma 4.1.** *Consider the RRLB MPC 5 and re-write it in a simpler way as*

$$\tilde{V}_N(x) = \min_{\mathbf{u}} \quad J(x, \mathbf{u})$$

*where $J(x, \mathbf{u}) = \tilde{l}(x, u_0) + \tilde{l}(f(x, u_0), u_1) + \cdots + \tilde{F}(f(f(...), u_{N-1}))$. If for a certain initial state $\bar{x}$ we suppose that $D_{\mathbf{u}} J(\bar{x}, \tilde{\mathbf{u}}) = 0$ and $\nabla^2_{\mathbf{uu}} J(\bar{x}, \tilde{\mathbf{u}}) \succ 0$ (the matrix is positive definite) then in a neighborhood of $\bar{x}$ we have:*

- $\forall k = 0, \ldots, N-1, \quad \|\tilde{u}_k(x)\| = O(\|x\|)$
- $\forall k = 1, \ldots, N, \quad \|\tilde{x}_k(x)\| = f(f(\ldots, u_{k-2}), u_{k-1}) = O(\|x\|)$

*Proof.* We can use the proof of the Theorem 4.2 in [Sti18] to argue that each $\tilde{x}_k$ and $\tilde{u}_k$ is continuously differentiable in a closed neighborhood of the origin. Therefore, by continuity their gradient is bounded on this neighborhood and they are Lipschitz. □

Now our main result:

**Theorem 4.2.** *Let's consider the problem 5 and assume the following:*

1. *If we denote the objective function as $J(x, \mathbf{u})$ as in Lemma 4.1, we have $D_{\mathbf{u}}J(0, \mathbf{u}(0)) = D_{\mathbf{u}}J(0,0) = 0$ and $\nabla^2_{\mathbf{uu}}J(0, \mathbf{u}(0)) = \nabla^2_{\mathbf{uu}}J(0,0) \succ 0$*

2. *When linearizing the system dynamics around the equilibrium and letting $A = D_x f(0,0)$, $B = D_u f(0,0)$, we suppose that the pair $(A, B)$ is stabilizable (the definition can be found in appendix 7.1). This implies in particular that there exists a stabilizing cost $K$, i.e. a matrix such that $A_K := A + BK$ only has eigenvalues in the unit disk.*

3. *The matrix $P$ defining the terminal costs is the unique positive definite solution to the following Lyapunov equation:*

$$P = A_K^T P A_K + \mu Q_K$$

*where $\mu > 1$ and $Q_K = Q + \epsilon M_x + K^T(R + \epsilon M_u)K$ .*

*Then the dynamical system $x^+ = f(x, \tilde{u}_0(x))$ is LAS, i.e. for all initial state in a neighborhood of the origin, the control law given by the RRLB MPC yields an asymptotically stable system.*

*Remark.* The matrix $K$ can be constructed using the DARE for the following modified infinite horizon LQR problem:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{\infty} x_k^T(Q + \epsilon M_x)x_k + u_k^T(R + \epsilon M_u)u_k \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k,\ k = 0, \dots, N-1 \end{aligned} \tag{6}$$

where $M_x = \nabla^2 B_x(0)$ and $M_u = \nabla^2 B_u(0)$, i.e.

$$\begin{cases} K = -(R + B^T \Pi B)^{-1} B^T \Pi A \\ \Pi = A_K^T \Pi A_K + Q_K \end{cases}$$

The only difference between this last equation and the Lyapunov equation in the statement of the theorem is the $\mu$ term. In theory it is very important, but in practice, we can actually take it very close to 1 and find $P$ and $K$ by solving

$$\begin{cases} K = -(R + B^T P B)^{-1} B^T P A \\ P = A_K^T P A_K + Q_K \end{cases}$$

More on that in 5.3.

*Proof.* To prove the result we will show that $\tilde{V}_N$ is a Lyapunov function in a neighborhood of the origin. First, by writing the Taylor expansion of the stage costs $\tilde{l}$ we can see that

$$\begin{aligned} \tilde{l}(x,u) &= x^T[\nabla^2_{xx}\tilde{l}(0,0)]x + u^T[\nabla^2_{uu}\tilde{l}(0,0)]u + O(\|x\|^3 + \|u\|^3) \\ &= x^T[Q + \epsilon M_x]x + u^T[R + \epsilon M_u]u + O(\|x\|^3 + \|u\|^3) \end{aligned}$$

so $\tilde{l}(x, Kx) = x^T Q_K x + O(\|x\|^3)$ . By the second assumption, we also have $\forall x \in \mathbb{R}^{n_x}$:

$$\tilde{F}(A_K x) + \mu x^T Q_K x - \tilde{F}(x) = 0$$

We can use the same reasoning as in the paragraph 2.5.5 of [RMD17] to show that $\tilde{F}$ is a local Lyapunov function, i.e. there exists a neighborhood of the origin such that $\forall x$ in it:

$$\tilde{F}(f(x,Kx)) + x^T Q_K x - \tilde{F}(x) \leq 0 \Longrightarrow \tilde{F}(f(x,Kx)) - \tilde{F}(x) + \tilde{l}(x,Kx) = O(\|x\|^3)$$

What's more, by previous lemma, the predicted states are all Lipschitz with respect to the initial state, so we can choose an even smaller neighborhood such that $\tilde{x}_N(x)$ is also in it.
Now using the optimal sequence of states and controls of the RRLB MPC starting at $x$, we can construct new feasible sequences for the problem starting at $\tilde{x}_1(x)$ as
$\mathbf{x}' = (\tilde{x}_1, \dots, \tilde{x}_N, f(\tilde{x}_N, K\tilde{x}_N))$ and $\mathbf{u}' = (\tilde{u}_1, \dots, \tilde{u}_{N-1}, K\tilde{x}_N)$. Then we have:

$$\tilde{V}_N(\tilde{x}_1) \leq \tilde{J}_N(\mathbf{x}', \mathbf{u}') = \underbrace{\tilde{J}_N(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})}_{=\tilde{V}_N(x)} \underbrace{\underbrace{-\tilde{l}(x,\tilde{u}_0)}_{=O(\|x\|^2)} + \underbrace{\tilde{F}(f(\tilde{x}_N, K\tilde{x}_N)) - \tilde{F}(\tilde{x}_N) + \tilde{l}(\tilde{x}_N, K\tilde{x}_N)}_{=O(\|\tilde{x}_N\|^3)=O(\|x\|^3)}}_{\leq -c\|x\|^2 \text{ in a smaller nbh}} \checkmark$$

We have therefore shown the descent property of our candidate Lyapunov function $\tilde{V}_N$. Finally, it is easy to show that $\tilde{V}_N$ is lower and upper bounded by coercive quadratic functions (since $Q, R$ are positive definite and $B_x, B_u$ are positive definite and upper bounded by quadratics), so that $\tilde{V}_N$ is indeed a Lyapunov function in a small neighborhood around the origin. □

## 4.2 Constraints satisfaction guarantees

This section follows closely the section IV.D of [FE17] but gives more loose results that cannot take us as far. In particular, will show that we can ensure the existence of a neighborhood around the origin such that if we start in it, the states and controls along the closed-loop trajectories will never violate the constraints. However, this neighborhood cannot be easily computed because of the generality of the considered dynamics, and will therefore not be verified experimentally in section 5.3. In the whole section, we assume Theorem 4.2 to hold.

**Lemma 4.3.** *Consider the RRLB MPC 5 and an initial state $x(0)$ in the neighborhood given by Theorem 4.2. Let's denote by $\{x(0), x(1), \dots\}$ and $\{u(0), u(1), \dots\}$ the closed-loop state and control trajectories (given by $u(k) = \tilde{u}_0(x(k))$, $x(k+1) = f(x(k), u(k))$). Then $\forall k \geq 0$:*

$$B_x(x(k)) \leq \frac{1}{\epsilon}\left(\tilde{V}_N(x(0)) - x(0)^T P_{\text{LQR}} x(0) - \sum_{k=0}^{\infty} \eta(x(k))\right)$$
$$B_u(u(k)) \leq \frac{1}{\epsilon}\left(\tilde{V}_N(x(0)) - x(0)^T P_{\text{LQR}} x(0) - \sum_{k=0}^{\infty} \eta(x(k))\right)$$

*where $P_{\text{LQR}}$ is the solution to the DARE associated to 6 and $\eta(x) = \tilde{l}(\tilde{x}_N(x), K\tilde{x}_N(x)) + \tilde{F}(\tilde{x}_N(x)) - \tilde{F}(f(\tilde{x}_N(x), K\tilde{x}_N(x)))$.*

*Proof.* In the proof of theorem 4.2 we showed that $\forall k \geq 0$:

$$\tilde{V}_N(x(k+1)) - \tilde{V}_N(x(k)) \leq -\tilde{l}(x(k), u(k)) + \tilde{l}(\tilde{x}_N(x(k)), K\tilde{x}_N(x(k))) - \tilde{F}(\tilde{x}_N(x(k))) + \tilde{F}(f(\tilde{x}_N(x(k)), \tilde{x}_N(x(k))))$$

so by summing everything we get a telescopic sum that we can compute using the fact that the

system is asymptotically stable so $\lim_{k\to\infty} \tilde{V}_N(x(k)) = 0$, we get:

$$\begin{aligned}
\tilde{V}_N(x(0)) &\geq \sum_{k=0}^{\infty} \tilde{l}(x(k), u(k)) - \tilde{l}(\tilde{x}_N(x(k)), K\tilde{x}_N(x(k))) + \tilde{F}(\tilde{x}_N(x(k))) - \tilde{F}(f(\tilde{x}_N(x(k)), \tilde{x}_N(x(k)))) \\
&= \sum_{k=0}^{\infty} l(x(k), u(k)) + \epsilon B_x(x(k)) + \epsilon B_u(u(k)) + \eta(x(k)) \\
&\geq x(0)^T P_{\mathrm{LQR}} x(0) + \sum_{k=0}^{\infty} \eta(x(k)) + \epsilon \sum_{k=0}^{\infty} B_x(x(k)) + \epsilon \sum_{k=0}^{\infty} B_u(u(k))
\end{aligned}$$

Even if we don't know a closed form for $\eta(x(k))$, we know that $\sum_{k=0}^{\infty} \eta(x(k))$ must be finite because it is bounded by $\tilde{V}_N(x(0))$. Now since the RRLB functions are all positive definite, we can easily conclude. □

We define for ease of notation the following quantities

$$\begin{aligned}
\alpha(x(0)) &= \tilde{V}_N(x(0)) - x(0)^T P_{\mathrm{LQR}} x(0) - \sum_{k=0}^{\infty} \eta(x(k)) \\
\beta_x &= \min_{i,x} \{B_x(x) \mid \mathrm{row}_i(C_x)x = d_{x,i}\} \\
\beta_u &= \min_{i,u} \{B_u(u) \mid \mathrm{row}_i(C_u)u = d_{u,i}\}
\end{aligned}$$

Even if not written explicitly, the bounds $\beta_x, \beta_u$ depend on the relaxation parameter $\delta$.

**Lemma 4.4.** *For all relaxation parameter $\delta$:*

$$\{x \in \mathbb{R}^{n_x} \mid B_x(x) \leq \beta_x\} \subseteq \mathcal{X}, \quad \{u \in \mathbb{R}^{n_u} \mid B_u(u) \leq \beta_u\} \subseteq \mathcal{U}$$

*Proof.* See Lemma 1 in [FE17]. □

**Theorem 4.5.** *In the same setting as lemma 4.3, for any initial state $x(0)$ in the set*

$$\mathcal{X}_N(\delta) := \{x \in \mathbb{R}^{n_x} \mid \alpha(x(0)) \leq \epsilon \min\{\beta_x, \beta_u\}\}$$

*there is no state or control constraint violation along the closed loop trajectories.*

*Proof.* For any $x(0) \in \mathcal{X}_N(\delta)$ it holds due to 4.3 that for all $k \geq 0$, $\epsilon B_x(x(k)) \leq \alpha(x(0)) \leq \epsilon\beta_x$ so $B_x(x(k)) \leq \beta_x$ and $x(k) \in \mathcal{X}$. The same reasoning applies on the controls. □

# 5 Numerical aspects of RRLB

In this section we take a more concrete look at the numerical methods employed to solve MPC problems in practice. One of the more important challenges of such implementations resides in the large number of variables and more generally the complexity of the NLP to be solved, that may not permit finding an accurate solution when the sampling time is very short. This issue is deeply aggravated on embedded platforms where the computational power is often limited. This is why, historically, the first successful implementations of MPC in industrial applications were found in chemical engineering, where the sampling time (duration between two state measurement, and therefore time available to compute a new control) are usually long enough to find an accurate solutions. For instance, in the chemical system studied in

subsection 5.3, the sampling time is of $200.0s$, while in robotics or autonomous driving the sampling time is rather in the order of the $10.0ms$.

A great deal of effort has been put in the past decade to design algorithms that make best use of scarce computational power to deliver usable results on embedded systems that necessitate very fast response times. One that rapidly gained interest is a variant of *Sequential Quadratic Programming* (or SQP) called *Real Time Iteration* (or RTI). Invented by Moritz Diehl and published in [DBS05], it can roughly be defined as SQP with a fixed number of iterations, usually only one, that makes use of the solutions found at the previous MPC prediction to warm-start the solver and to ensure both computational efficiency and accuracy.

In subsection 5.1 we formally introduce the RTI scheme for general MPC and the specialize it for RRLB MPC, then briefly discuss it stability properties in subsection 5.2 and finally present numerical results verifying the theoretical properties of RRLB MPC and compare its performance with regular MPC in subsection 5.3.

## 5.1 The RTI scheme

We assume in this section familiarity with the SQP framework for general NLPs, as described for example in [NW06], and specialize it in the context of MPC. We son't mention here the necessary assumptions necessary for SQP convergence and just assume them to be satisfied to better focus on our object of study : MPC.

### 5.1.1 SQP for regular MPC

Just as in sections 3 and 4, we are going to consider the target state and controls to be the origin, which will simplify our notations inspired from [Gro+16]. With the following general MPC:

$$\begin{aligned}
\mathrm{MPC}(x) = \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} l(x_k, u_k) \; + \; F(x_N) \\
\text{s.t.} \quad & x_0 = x \text{ and } x_{k+1} = f(x_k, u_k), \; k = 0, \dots, N-1 \\
& h_x(x_k) \leq 0, \; k = 0, \dots, N \\
& h_u(u_k) \leq 0, \; k = 0, \dots, N-1
\end{aligned}$$

we can define the following QP subproblem:

$$\begin{aligned}
\mathrm{QP}(x, \mathbf{x}_i, \mathbf{u}_i) = \operatorname*{argmin}_{\Delta\mathbf{x}_i, \Delta\mathbf{u}_i} \quad & \sum_{k=0}^{N-1} \begin{pmatrix} \Delta x_{i,k} \\ \Delta u_{i,k} \end{pmatrix}^T H_{i,k} \begin{pmatrix} \Delta x_{i,k} \\ \Delta u_{i,k} \end{pmatrix} + J_{i,k}^T \begin{pmatrix} \Delta x_{i,k} \\ \Delta u_{i,k} \end{pmatrix} \\
& + \Delta x_{i,N}^T H_{i,N} \Delta x_{i,N} + J_{i,N}^T \begin{pmatrix} \Delta x_{i,N} \\ \Delta u_{i,N} \end{pmatrix} \\
\text{s.t.} \quad & \Delta x_0 = x - x_{i,0} \\
& \Delta x_{i,k+1} = A_{i,k} \Delta x_{i,k} + B_{i,k} \Delta u_{i,k} + r_{i,k}, \; k = 0, \dots, N-1 \\
& C_{i,k} \Delta x_{i,k} + h_{x,i,k} \leq 0, \; k = 0, \dots, N \\
& D_{i,k} \Delta x_{i,k} + h_{u,i,k} \leq 0, \; k = 0, \dots, N-1
\end{aligned}$$

with the sensitivities:

$$A_{i,k} = D_x f(x_{i,k}, u_{i,k}),\ B_{i,k} = D_u f(x_{i,k}, u_{i,k}),\ C_{i,k} = D_x h(x_{i,k}, u_{i,k}),\ D_{i,k} = D_u h(x_{i,k}, u_{i,k})$$
$$J_{i,k} = \nabla l(\Delta x_{i,k}, \Delta u_{i,k}),\ J_{i,N} = \nabla F(\Delta x_N)$$
$$r_{i,k} = f(x_{i,k}, u_{i,k}) - x_{i,k+1}, \quad h_{x,i,k} = h_x(x_{i,k}), \quad h_{u,i,k}(u_{i,k})$$

and $H_{i,k} \approx \nabla^2_{(x_k,u_k),(x_k,u_k)} \mathcal{L}(\mathbf{x}_i, \mathbf{u}_i),\ H_{i,N} \approx \nabla^2_{x_N,x_N,u_N} \mathcal{L}(\mathbf{x}_i, \mathbf{u}_i)$, where $\mathcal{L}$ is the Lagrangian of the MPC problem.

When using quadratic stage and terminal costs and when we choose to use Gauss-Newton approximation for the Hessian, we get:

$$H_{i,k} = \begin{pmatrix} Q & 0 \\ 0 & R \end{pmatrix}, \quad H_{i,N} = P, \quad J_{i,k} = \begin{pmatrix} Q & 0 \\ 0 & R \end{pmatrix} \begin{pmatrix} \Delta x_{i,N} \\ \Delta u_{i,N} \end{pmatrix}, \quad J_{i,N} = P \begin{pmatrix} \Delta x_{i,N} \\ \Delta u_{i,N} \end{pmatrix}$$

What's more, when the constraints are polytopic, the sensitivities read:

$$C_{i,k} = C_x,\ D_{i,k} = C_u,\ h_{x,i,k} = C_x x_{i,k} - d_x,\ h_{u,i,k} = C_u u_{i,k} - d_u$$

These two particular cases make up the MPC we have been studying in this whole paper, that we will call in this section the *regular MPC*.
With all these definitions, the complete SQP method reads:

**Algorithm 1** SQP method for regular MPC

**procedure** SQP(x, $\mathbf{x}^{init}$, $\mathbf{u}^{init}$)
  $\mathbf{x}_0 \leftarrow \mathbf{x}^{init}$ ▷ initial guess for the states specified by the user
  $\mathbf{u}_0 \leftarrow \mathbf{u}^{init}$ ▷ initial guess for the controls specified by the user
  $k \leftarrow 0$
  **while** a stopping cirterion is not met **do**
    Compute $r_{i,k}, h_{x,i,k}, h_{u,i,k}$ and the sensitivities $A_{i,k}, B_{i,k}, C_{i,k}, D_{i,k}$ as above
    $(\Delta \mathbf{x}_i, \Delta \mathbf{u}_i) \leftarrow \mathrm{QP}(x, \mathbf{x}_i, \mathbf{u}_i)$ ▷ Find descent direction as above
    $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \Delta \mathbf{x}_i$
    $k \leftarrow k + 1$
  **end while**
**end procedure**

### 5.1.2 SQP for RRLB MPC

Using the same notations as above, the only things that change are the fact that the inequality constraints are ignore and that the sensitivities are computed in the following way:

$$H_{i,k} = 2 \begin{pmatrix} Q & 0 \\ 0 & R \end{pmatrix} + \epsilon \begin{pmatrix} \nabla^2 B_x(x_{i,k}) & 0 \\ 0 & \nabla^2 B_u(u_{i,k}) \end{pmatrix}, \quad H_{i,N} = P + \epsilon \nabla B_x(x_{i,N})$$
$$J_{i,k} = 2 \begin{pmatrix} Q & 0 \\ 0 & R \end{pmatrix} \begin{pmatrix} x_{i,k} \\ u_{i,k} \end{pmatrix} + \epsilon \begin{pmatrix} \nabla B_x(x_{i,k}) \\ \nabla B_u(u_{i,k}) \end{pmatrix}, \quad J_{i,N} = 2P x_{i,N} + \epsilon \nabla B_x(x_{i,N})$$

### 5.1.3 Remarks on SQP

In [Gro+16] the presented SQP procedure includes a line-search step that computes a step size $\alpha \in (0, 1]$ such that enough progress is made at each iteration. In the case of general SQP, this is particularly important to ensure convergence. However, if the initial guesses $\mathbf{x}^{init}$, $\mathbf{u}^{init}$ are well chosen, we can take full steps at each iteration and therefore attain faster convergence.

The most natural way to construct this initial guess would be to *shift* the solution found at the previous control update (or equivalently, at the previous call to the solver of algorithm 1). More precisely, if at time $k$ we found $\mathbf{x} = (x_0, \dots, x_N), \mathbf{u} = (u_0, \dots, u_{N-1})$ then our initial guess at time $k+1$ will be $\mathbf{x}^{init} = (x_1, \dots, x_N, f(x_N, u_{new})), \mathbf{u}^{init} = (u_1, \dots, u_{N-1}, u_{new})$ where $u_new$ might be chosen using different strategies:

- whenever possible, setting $u_{new} = \kappa(x_N)$ where $\kappa : \mathcal{X} \to \mathcal{U}$ is a *locally stabilizing control law* that would make converge to the origin all the states in a neighborhood, without taking into account the constraints. An example of such locally stabilizing control law can be the control law of the infinite horizon LQR mentioned in subsection 2.1.
- using the reference control $u^*$ (which again we consider as 0 for simplicity).
- using the same control as in the previous stage: $u_{new} = u_{N-1}$ .

This shifting heuristic is based on the fact that the dynamics $f$ should either exactly equal or at least close to the actual dynamics of the system so that this initial guess is very close to the actual solution. When it is the case, and if the solution found at time $k$ is feasible, then so will be the initial guess for time $k+1$, and the very first iteration of algorithm 1 provides a very accurate estimate of the solution at time $k+1$. Numerical illustrations of these statements can be found in [Gro+16]. This method supposes that the very first prediction of a closed-loop experiment has to be computed exactly, but this can be done offline (i.e. before starting the simulation) with an arbitrary accuracy.

### 5.1.4 The RTI scheme

RTI relies on previous remarks to find an accurate solution to the MPC problem and combines them with another important idea trying to solve the *real-time dilemma* which can be summarized in the following way:

> Upon obtaining a new state estimate $x$, a new SQP procedure can be started to find the solution of the MPC. During the execution, the systems continues to evolve and by the time it finishes, the current state of the system won't be the one fed to the algorithm, making the prediction obsolete.

To overcome this issue, RTI separates the SQP procedure into:

- a *preparation phase* comprising the computation of the sensitivities and the condensation (i.e. the creation of the sparse matrices defining the QP problem by concatenating the terms of each stage) and
- a *feedback phase* that solves the QP using the last available state estimate as initial state.

This makes sense, since the preparation phase only actually needs information from the previous call to the solver, and can therefore be started right after it, before the next state estimate is even available.

Instead of presenting RTI with pseudocode as in algorithm 1, we will rather draw up a timeline resuming how the different steps of the algorithm are used in real situation:

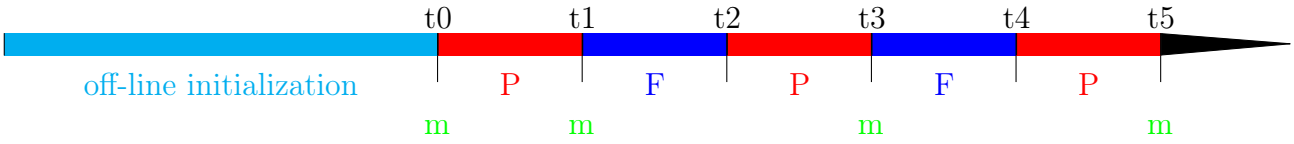



Figure 2: RTI timeline

Several important points should be highlighted:

- Before the start of the experiment, we solve the problem with an arbitrary precision **offline**, when there is no limit on the running time of the solver. This supposes that we know in advance the initial state of the system, but this is a common and realistic assumption. In this phase, we could use the general SQP with as many steps as necessary for full convergence (remember that we can't properly use RTI yet since we don't have a good initial guess), or any other algorithm to solver constrained NLPs.
- The "m" labels indicate the state measurements of the system. In practice, we suppose they are done at a constant rate given by the sampling time $dt$, and we can thus say that $\mathrm{t1} - \mathrm{t0} = dt$ and $\mathrm{t}(2k+3) - \mathrm{t}(2k+1) = dt,\ \forall k \geq 0$.
- The phases denoted by "P" and "F" are the preparation and feedback phases of the RTI scheme as described above.
- During the experiment, the state evolves continuously, but the control parameters are kept constant on all intervals of the form $[\mathrm{t}(2k), \mathrm{t}(2k+2))$ because we update the control only when a new one is computed.

More practical details on RTI will be given in subsection 5.3 where we apply this scheme to the simulation of a real system. In particular, we will give explicit details on the form of the matrices to condense.

## 5.2 Asymptotic stability of RTI scheme

Section 4 proved several important theoretical properties of RRLB MPC, and the next subsection presents some experimental verifications of these theorems. However, it should be noted that all the solution to problem 5 found in these experiments are by essence inexact, and we can therefore not easily claim that the asymptotic stability and the constraint satisfaction still hold in practice. That's why we make the distinction between *nominal stability* and *real-time stability*.

The determination of assumptions and results guaranteeing these properties in the real-time framework have been studied for example by Moritz Diehl or Andrea Zanelli, although in slightly different contexts than ours. The managed to derive powerful theorems but their assumptions are hard or even impossible to verify in practice and they are not really verified in practice. Instead, the desired properties are verified experimentally by means of a simple simulation of the system at hand. This is the scope of next subsection, in which we will not worry about this distinction between nominal and real-time stability.

## 5.3 Numerical experiments

In this section, we verify the stability properties of the RRLB MPC using a concrete system that is often used to benchmark different MPC schemes or solvers: the Continuously Stirred Tank Reactor (CSTR). We will not however determine explicitly the neighborhood given by theorem 4.2 because of the lack of theoretical description. For the same reason, we will not compute the neighborhood given by theorem 4.3, and just see if the constraints are satisfied or not along the closed-loop trajectories. When they are, this will allow us to explicitly compare RRLB to regular MPC.

| Ref point | $c_A^*$ | $c_B^*$ | $\theta^*$ | $\theta_K^*$ | $u_1^*$ | $u_2^*$ |
|---|---|---|---|---|---|---|
| 1 | 2.140 | 1.090 | 114.191 | 112.907 | 14.19 | -1113.50 |
| 2 | 2.715 | 1.023 | 105.506 | 100.892 | 13.666 | -3999.589 |
| 3 | 2.975 | 0.954 | 101.150 | 95.194 | 12.980 | -5162.957 |

Table 1: Reference points for the CSTR system

### 5.3.1 Experimental setup

The CSTR system describes an exothermic chemical reaction with 4 states ($c_A$ the concentration of cyclopentadiene (reactant), $c_B$ the concentration of cyclopentenol (product), $\vartheta$ the temperature in the reactor and $\vartheta_K$ the temperature in the cooling jacket of the tank reactor) and 2 controls ($u_1$ the scaled feed inflow rate of cyclopentadiene and $u_2$ the heat removal rate of the external heat exchanger designed to cool down the reactor). The system of ODEs describing the model as well as the costs and constraints can be found in the appendix 7.4, and more details about this system can be found in [Die01].

By a simple steady state analysis procedure [1], we determined 3 viable reference points for states and controls that can be found (at $10^{-3}$ precision and in SI units) in table 1.
Some general remarks on the implementation:

- None of the reference points in table 1 are 0, therefore we had to translate everything as briefly described at the beginning of section 2.1.
- We numerically checked the assumptions of theorem 4.2 for each reference point.
- The discrete-time dynamics have been obtained from the model's ODE by using a 6-step RK4 scheme.
- We chose a sampling time of $20.0s$, and variable horizon lengths between 25 and 250 time steps. In all our experiments we set the maximum number of control feedbacks to 350 and ran the simulation in closed loop (i.e. applying to the system the control computed by the MPC) until the state converged to the reference state within a $10^{-3}$ relative tolerance in norm (i.e. $\|x - x^*\|_\infty / \|x^*\|_\infty < 10^{-3}$).
- For RRLB MPC, we determined the terminal cost by solving the modified DARE with $\mu = 1$ as mentioned in 4.1. We will see in the results that this does not impact convergence when the horizon size is big enough.
- For RRLB MPC, the relaxation parameter $\delta$ was chosen as $\frac{1}{2}\min\{d_{x,1}, \dots, d_{x,q_x}, d_{u,1}, \dots, d_{u,q_u}\}$ and the barrier parameter $\epsilon$ was chosen as $1.0e-3$ after a bit of tuning in order to ensure that the MPC would converge.
- The initial prediction (as described in 5.1) is computed using the interior point method solver IPOPT (see [WB06]) through the algorithmic differentiation package CasADi (see [And+19]).
- RTI was implemented from scratch using OSQP (see [Ste+20]) to solve the quadratic subproblems. This library solves general quadratic problems of the form:

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \quad & \frac{1}{2} z^T P z + q^T z \\ \text{s.t.} \quad & l \leq Az \leq u \end{aligned}$$

[1] This procedure was applied to the discrete-time dynamics obtained via discretization with RK4.

We therefore have to translate the QPs found in section 5.1 for regular MPC and RRLB MPC into this form, which corresponds to the condensation part of the preparation phase. The exact form of the matrices and vectors involved in this formulation can be found in 7.3.

- The whole implementation for this project was done in Python and can be found in the folder `cstr` attached to the project report. To test the code, we recommend creating a virtual environment using the `requirements.txt` file provided. The code is heavily documented and general indications on the structure of the code are given in the `README.md` file.

### 5.3.2 Experimental procedure

We generated a list of 5 random initial states (that can be found in the columns of the file `exp_all_initial_states.csv` in the directory `results`) using a uniform distribution between the following bounds:

$$0.0 \leq c_A, c_B \leq 5.0,\ 98.0 \leq \vartheta \leq 110.0,\ 92.0 \leq \vartheta_K \leq 110.0$$

and with the 3 reference points in table 1, this generated 15 different situations in which we ran the following experiment:

We first ran an open loop simulation of the infinite horizon MPC (approximated by a regular MPC with horizon length of 5000) to determine an 'optimal' behavior[2], and then ran closed-loop simulations with RRLB MPC and regular MPC and variable horizon lengths for both of them (in the list $\{25, 50, 100, 250\}$). For all of them we measured:

- the *performance measure*, i.e. the sum of the stage costs over all the time steps up until convergence, or the maximum number of time feedbacks, is reached.

  For the the closed loop simulations, RTI was used and the sum was only done over the time instants where a *measurement is made* (see 2).

- the number of iterations taken to reach the convergence.

- whether the constraints were violated or not.

Additionally, for the closed-loop simulations where RTI was used, we also measured the times (average and standard error) in milliseconds of the sensitivity computation, the condensation, and the QP solve. All these measures can be found in the file `exp_all_results.csv` in the folder `results`.

### 5.3.3 Results and discussion

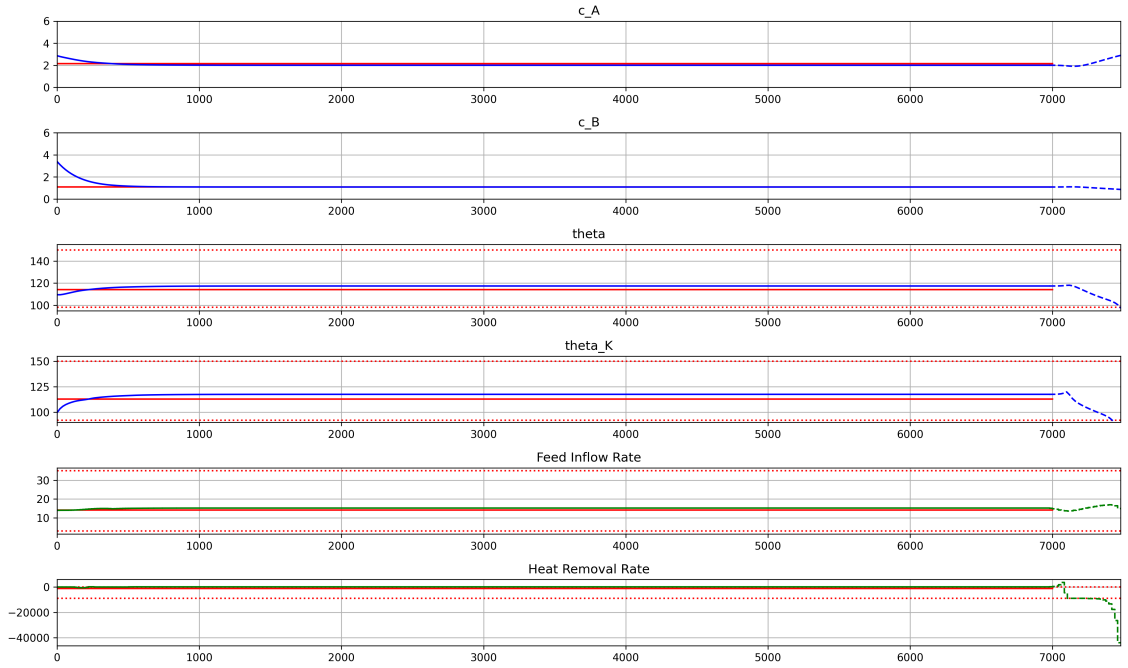
From all the measures described above, we can make several observations:

- In every situation, when the horizon size is 25, the RRLB MPC does not manage to converge to the desired reference point and instead converges to another steady state. For instance, for initial state n°1 and reference point n°1, it converges to (values are given at $10^{-4}$ precision):

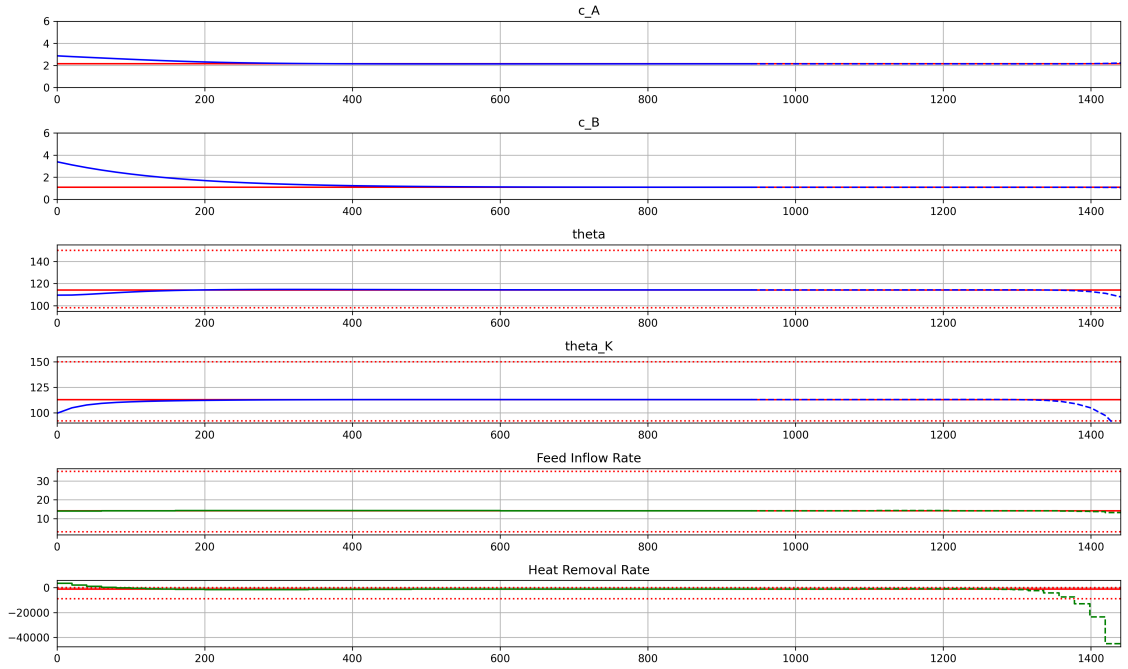
  $$x^* = (2.0005, 1.086, 117.3737, 117.5026), \quad u^* = (15.2094, 111.7617)$$

  Nonetheless, when decreasing the barrier parameter $\epsilon$, we can produce convergence at the expense of more constraint violation due to the fact that the constraints are somehow less enforced. The corresponding closed loop trajectories can be found in figure 3.

[2] As described in section 2.1




(a) $\epsilon = 1.0e - 3$




(b) $\epsilon = 1.0e - 8$

Figure 3: Closed-loop trajectories for the RRLB MPC with initial state n°1, reference point n°1 and horizon size 25

- In every situation, whe the horizon size was 50, the RRLB MPC did not converge before the 350 iterations either, however we see in the results that the performance measure is much closer to the one achieved with horizon sizes 100 and 250 (when convergence was observed). This could indicate that the system actually stabilizes to the desired reference point, but just slower, and has not yet reached the desired tolerance at the 350th iteration. By inspecting the closed-loop trajectory in figure 4 we can see that this is indeed the case. The final (relative) discrepancy on the state is actually $6.71e-3$, which is quite close to the $1.0e-3$ tolerance.

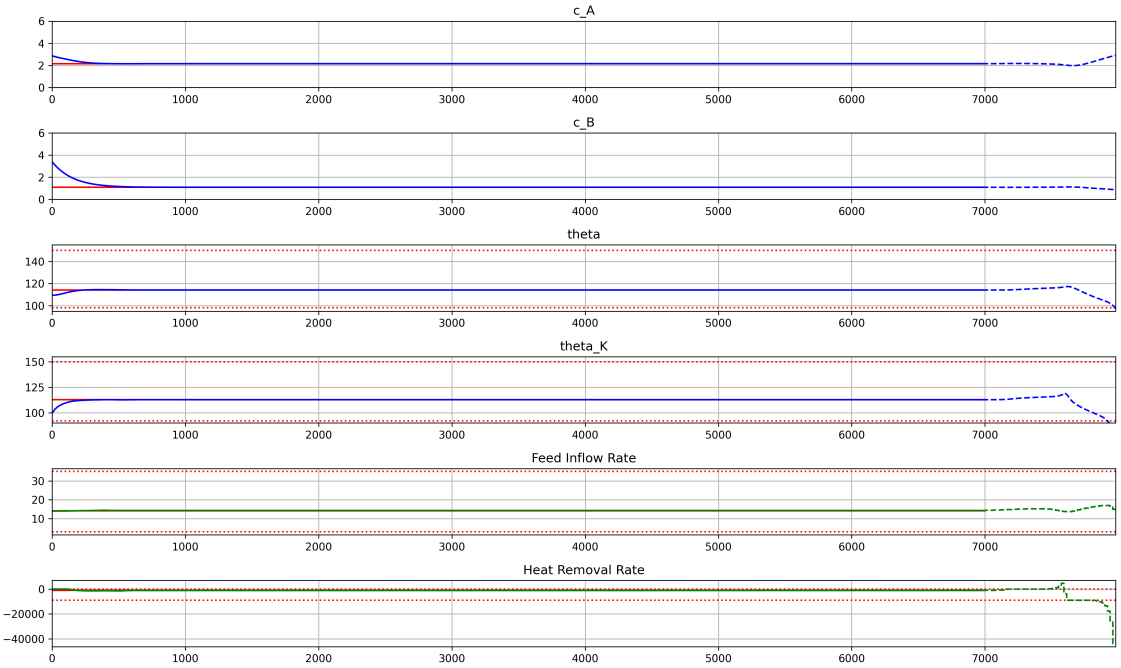



Figure 4: Closed-loop trajectory for the RRLB MPC with initial state n°1, reference point n°1 and horizon size 50

- For horizon sizes 100 and 250, the RRLB MPC always converged to the desired objective and what ca be observed is that it always converged faster than regular MPC, in fewer iterations, and with smaller performance measure. In half of the situations this is caused by some constraint violation that allows the MPC to take "shortcuts", i.e. some trajectories that might be have a lower cost, but that are do not verify all the constraints. We used our script `exp_analysis.py` to check case by case that this was indeed the case. In the other half, the improvements are quite substantial and their average (aggregated along the situations) can be found in figure 5.

- In general, we can see that augmenting the horizon size always improves the number of iterations needed to converge and the performance measure, for RRLB as for regular MPC. This general property of MPC is linked to the Bellman principle of optimality cited in 2.1, which suggests that the infinite horizon MPC is optimal, and that this optimum is better and better approximated as $N \to \infty$.

- The fact that the RRLB MPC only converges for a big enough horizon could be explained by the proof of theorem 4.2 in which the actual condition we needed to prove the descent property was that $\tilde{x}_N$ had to belong to the neighborhood given by the paragraph 2.5.5 of [RMD17], let's call it $\mathcal{N}_0$. Consequently, we could suppose the existence of a larger neighborhood $\mathcal{N}_N$ such that $x \in \mathcal{N}_N \implies \tilde{x}_N(x) \in \mathcal{N}_1$, and imagine that $\mathcal{N}_N$ gets

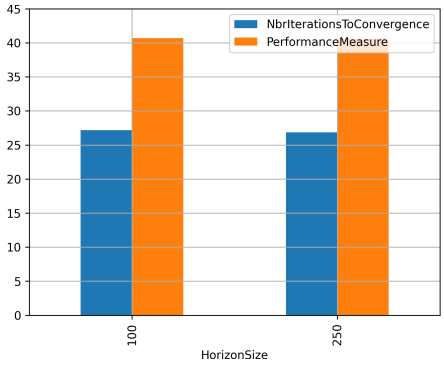
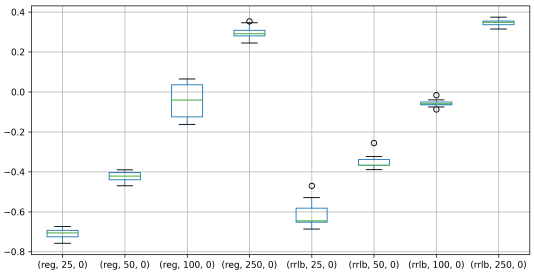


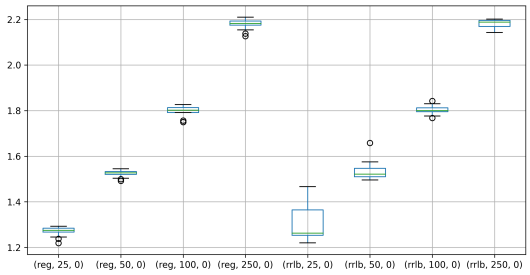

Figure 5: Average improvement in percentage of the Number of iterations to convergence and the Performance measure of RRLB MPC compared to regular MPC

monotonically larger as $N \to \infty$. Intuitively, this last property could be motivated by the fact that as the horizon size increases, the MPC has more time to reach $\mathcal{N}_0$. However, these properties are out of the scope of this paper and could be the object of further research on the subject of RRLB MPC.

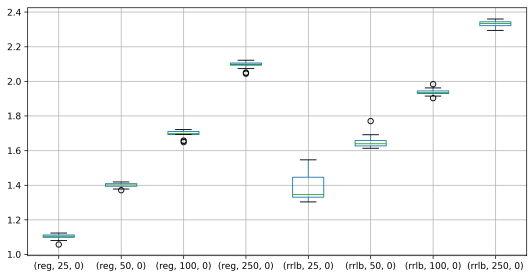
- As suggested by the figure 6, RRLB seems i average a bit more computationally demanding. The main difference appears to be in the sensitivity computation time, which is understandable given their more complex form involving the gradients and hessians of RRLB functions shown in appendix 7.3.




(a) Average sensitivities computation times (in log scale) throughout the experiments




(b) Average condensation times (in log scale) throughout the experiments




(c) Average solving times (in log scale) throughout the experiments

Figure 6: Boxplots of runtimes for different phases of RTI, averaged by scheme and horizon length

# 6 Conclusion and outlook

Throughout this report, we have explored the theory of MPC and RRLB MPC, and we have managed to show two important results. However, because of the generality of the considered dynamics, they are not as powerful as the ones in the linear case that were recalled in Theorems 3.1 and 3.2. In particular, their lack of precise description of the neighborhoods of convergence and of constraint satisfaction makes them of little interest in real-life applications. They are nevertheless essential theoretical building blocks and should serve as motivation for further research.

You can find below a non-exhaustive list of potential research questions that came up during the semester:

- Could we improve the description of the neighborhood provided by theorem 4.2 by proving the kind of monotonicity property evoked in section 5.3.3?
- How do the parameters $\epsilon$ and $\delta$ really influence the behavior of the scheme? In particular, are there ways to determine values for those parameters that always ensure asymptotic stability and/or constraint satisfaction?
- If we move the initial state away from the reference point, will the regular MPC scheme yield divergent trajectories before RRLB MPC (for suitable horizon lengths)?
- Could we improve our numerical results by using 2 or more iterations of SQP in our RTI scheme?

Our numerical illustrations showed a case where the system succeeded in stabilizing, but it might be due to the simplicity of the dynamics of CSTR. We could think of more complex systems governed by stiff ODEs, where the linearization would find a reduced accuracy. One such example is the nonlinear hanging chain problem introduced in [Kou+18].

# 7 Appendix

## 7.1 Controllable vs. Stabilizable

The notions of controllability and stabilizability are central in control theory and are the first hypothesis one should plan to verify on a concrete system. They are intrinsic properties of the equation describing the evolution of the system (either an ODE for a continuous-time system or a map for a discrete-time system) and could roughly be defined as follows:

- A system is *controllable* if it can be steered from any feasible state to any other reference state in a finite time.
- A system is *stabilizable* if it can be steered from any feasible state to any other feasible state in an infinite time, or equivalently that it asymptotically approaches this reference state.

These definitions can be made formal in the case of linear time invariant (or LTI) dynamics of the form $\dot{x} = Ax + Bu$ in the continuous-time case, and $x^+ = Ax + Bu$ in the discrete-time case:

**Definition 7.1** (Controllability)**.**

- A continuous LTI dynamical system $\dot{x} = Ax+Bu$ is controllable if $\forall \lambda \in \text{eig}(A),\ \text{rank}(A - \lambda I_{n_x}, B) = n_x$ or equivalently if

$$\text{rank} \underbrace{\begin{pmatrix} B & | & AB & | & \dots & | & A^{n_x-1}B \end{pmatrix}}_{:=C} = n_x$$

  where $C$ is called the *controllability matrix*.

- A discrete LTI system $x^+ = Ax + Bu$ is controllable if any of the conditions above holds for $(A, B)$.

**Definition 7.2** (Stabilizability)**.**

- A continuous LTI dynamical system $\dot{x} = Ax + Bu$ is stabilizable if $\forall \lambda \in \mathbb{R},\ \text{rank}(A - \lambda I_{n_x}, B) \leq n_x \implies \Re(\lambda) < 0$ or equivalently if $\exists K \in \mathbb{R}^{n_x \times n_u} \ : \ A + BK$ is Hurwitz, i.e. only has eigenvalues with negative real part.
- A discrete LTI system $x^+ = Ax + Bu$ is stabilizable if any of the conditions above holds for $(A, B)$.

*Remark.* The stabilizable matrix $K$ can be explicitly constructed in the case of LQR as mentioned in section 2.1.

## 7.2 Derivation of the classical DARE

If we consider the LQR problem presented in 2.1, then Bellman's principle of optimality reads:

$$V_\infty(x) = \min_{u \in \mathbb{R}^{n_u}} \underbrace{x^T Q x + u^T R u + V_\infty(Ax + Bu)}_{=:\phi(x,u)}$$

If we suppose that $V_\infty$ is itself a quadratic function, i.e. $V_\infty(x) = x^T P x$, then the first order optimality conditions at a minimizer $u^*$ are:

$$\nabla_u \phi(x, u^*) = 2Ru^* + 2B^T P(Ax + Bu^*) = 0 \implies u^* = \underbrace{-R + B^T P B^{-1} B^T P A}_{=:K} x$$

and by plugging this value back into the minimization problem and developing the form of $K$, we can obtain the DARE:

$$\begin{aligned} V_\infty(x) = x^T P x &= x^T Q x + u^{*\,T} R u^* + V_\infty(Ax + Bu^*) \\ &= x^T P x = x^T Q x + x^T K^T R K x + V_\infty((A + BK)x) \\ &= \cdots = x^T (Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A) x \\ \implies & \boxed{P = Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A} \end{aligned}$$

## 7.3 OSQP condensation

If we consider the optimization variable of the QP subproblem (as described in section 5.1) to be $z = (\Delta x_0, \ldots, \Delta x_N, \Delta u_0, \ldots, \Delta u_{N-1})$, and we write $x$ for the system's current state and $(x_0^{init}, \ldots, x_N^{init}, u_0^{init}, \ldots, u_{N-1}^{init})$ for the initial guess (obtained by shift), then the matrices and vectors in the OSQP formulation will be:

$$q_{OSQP} = 2 \begin{pmatrix} Q & & & & & & \\ & \ddots & & & & & \\ & & Q & & & & \\ & & & P & & & \\ & & & & R & & \\ & & & & & \ddots & \\ & & & & & & R \end{pmatrix} \begin{pmatrix} x_0^{init} \\ \vdots \\ x_N^{init} \\ u_0^{init} \\ \vdots \\ u_{N-1}^{init} \end{pmatrix} + \epsilon \begin{pmatrix} \nabla B_x(x_0^{init}) \\ \vdots \\ \nabla 0 \\ \nabla B_u(u_0^{init}) \\ \vdots \\ \nabla B_u(u_{N-1}^{init}) \end{pmatrix}$$

$$P_{OSQP} = 4 \begin{pmatrix} Q & & & & & & \\ & \ddots & & & & & \\ & & Q & & & & \\ & & & P & & & \\ & & & & R & & \\ & & & & & \ddots & \\ & & & & & & R \end{pmatrix} + 2\epsilon \begin{pmatrix} \nabla^2 B_x(x_0^{init}) & & & & & & \\ & \ddots & & & & & \\ & & 0 & & & & \\ & & & \nabla^2 B_u(u_0^{init}) & & & \\ & & & & \ddots & & \\ & & & & & \nabla^2 B_u(u_{N-1}^{init}) \end{pmatrix}$$

$$A_{OSQP} = \begin{pmatrix} I_{n_x} & & & & & & & & 0 & & \\ A_0^{init} & -I_{n_x} & & & & & & & B_0^{init} & \ddots & \\ & A_1^{init} & -I_{n_x} & & & & & & & \ddots & \\ & & \ddots & \ddots & & & & & & & \\ & & & A_{N-1}^{init} & -I_{n_x} & & & & B_{N-1}^{init} & 0 & \\ C_x & & & & & & & & & & \\ & C_x & & & & & & & & & \\ & & \ddots & & & & & & & & \\ & & & C_x & & & & & & & \\ & & & & C_x & & & & & & \\ & & & & & C_u & & & & & \\ & & & & & & \ddots & & & & \\ & & & & & & & \ddots & & & \\ & & & & & & & & C_u & & \end{pmatrix}$$

$$l_{OSQP} = \begin{pmatrix} x - x_0^{init} \\ x_1^{init} - f(x_0^{init}, u_0^{init}) \\ \vdots \\ x_N^{init} - f(x_{N-1}^{init}, u_{N-1}^{init}) \\ -\infty \\ \vdots \\ -\infty \end{pmatrix} \qquad u_{OSQP} = \begin{pmatrix} x - x_0^{init} \\ x_1^{init} - f(x_0^{init}, u_0^{init}) \\ \vdots \\ x_N^{init} - f(x_{N-1}^{init}, u_{N-1}^{init}) \\ d_x - C_x x_0^{init} \\ \vdots \\ d_x - C_x x_N^{init} \\ \vdots \\ d_u - C_u u_0^{init} \\ \vdots \\ d_u - C_u u_{N-1}^{init} \end{pmatrix}$$

In the above, all the red terms only appear for regular MPC, and the blue terms only for RRLB MPC.

## 7.4 CSTR model, constraints and MPC costs

The evolution of the system is described by the following system of ODEs:

$$\begin{aligned}
\dot{c}_A(t) &= u_1(t)(c_{A0} - c_A(t)) - k_1(\vartheta(t))c_A(t) - k_3(\vartheta(t))c_A(t)^2 \\
\dot{c}_B(t) &= -u_1(t)c_B(t) + k_1(\vartheta(t))c_A(t) - k_2(\vartheta(t))c_B(t) \\
\dot{\vartheta}(t) &= u_1(t)(\vartheta_0 - \vartheta(t)) + \frac{k_w A_R}{\rho C_p V_R}(\vartheta_K(t) - \vartheta(t)) \\
&\quad - \frac{1}{\rho C_p}\left[k_1(\vartheta(t))c_A(t)H_1 + k_2(\vartheta(t))c_B(t)H_2 + k_3(\vartheta(t))c_A(t)^2 H_3\right] \\
\dot{\vartheta}_K(t) &= \frac{1}{m_K C_{PK}}\left[u_2(t) + k_w A_R(\vartheta(t) - \vartheta_K(t))\right]
\end{aligned}$$

where $k_i(\vartheta) = k_{i,0}\exp\left(\frac{E_i}{\vartheta + 273.15}\right)$ for $i = 1, 2, 3$. All the coefficients appearing in these equations are given in table 2:

| Symbol | Value | Symbol | Value |
|---|---|---|---|
| $k_{10}$ | $1.287e12\,h^{-1}$ | $\rho$ | $0.9342\,\frac{kg}{L}$ |
| $k_{20}$ | $1.287e12\,h^{-1}$ | $C_p$ | $3.01\,\frac{kJ}{kg \cdot K}$ |
| $k_{30}$ | $9.043e9\,h^{-1}$ | $k_w$ | $4032\,\frac{kJ}{h \cdot m^2 \cdot K}$ |
| $E_1$ | $-9758.3$ | $A_R$ | $0.215\,m^2$ |
| $E_2$ | $-9758.3$ | $V_R$ | $10\,L$ |
| $E_3$ | $-8560$ | $m_K$ | $5\,kg$ |
| $H_1$ | $4.2\,\frac{kJ}{mol}$ | $C_{PK}$ | $2.0\,\frac{kJ}{kg \cdot mol}$ |
| $H_2$ | $-11.0\,\frac{kJ}{mol}$ | $c_{A0}$ | $5.1\,\frac{mol}{L}$ |
| $H_3$ | $-41.85\,\frac{kJ}{mol}$ | $\vartheta_0$ | $104.9\,°C$ |

Table 2: CSTR model coefficients

The constraints on the controls are given by the following equations:

$$\begin{aligned}
3.0\,h^{-1} \leq u_1 &\leq 25.0\,h^{-1} \\
-9000.0\,\frac{kJ}{h} \leq u_2 &\leq 0.0\,\frac{kJ}{h}
\end{aligned}$$

There were no constraints originally in [Die01], but some were introduced in [HFD11] on $\vartheta, \vartheta_K$. We added some more to make sure that the feasible set of the states was polytopic and not just polyhedral (i.e. it is bounded), which ensures that the RRLB function exists.

$$\begin{aligned} 0.0 \leq c_A &\leq 10.0 \\ 0.0 \leq c_B &\leq 10.0 \\ 98.0 \leq \vartheta &\leq 150.0 \\ 92.0 \leq \vartheta_K &\leq 150.0 \end{aligned}$$

The costs are finally given by

$$Q = \begin{pmatrix} 0.2\, mol^{-2}L^2 & 0 & 0 & 0 \\ 0 & 1.0\, mol^{-2}L^2 & 0 & 0 \\ 0 & 0 & 0.5°C^{-2} & 0 \\ 0 & 0 & 0 & 0.2°C^{-2} \end{pmatrix}, \qquad R = \begin{pmatrix} 0.5\, h^2 & 0 \\ 0 & 5.0e-7\, kJ^{-2}h^2 \end{pmatrix}$$

# 8 Bibliography

# References


[And+19] Joel A E Andersson et al. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[Bel57] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957. ISBN: 9780486428093.

[DBS05] Moritz Diehl, Hans Georg Bock, and Johannes P. Schlöder. "A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control". In: *SIAM Journal on Control and Optimization* 43.5 (2005), pp. 1714–1736. DOI: 10.1137/S0363012902400713. eprint: https://doi.org/10.1137/S0363012902400713. URL: https://doi.org/10.1137/S0363012902400713.

[Die01] Moritz Diehl. "Real-Time Optimization for Large Scale Nonlinear Processes". PhD thesis. Universität Heidelberg, 2001.

[FE15] Christian Feller and Christian Ebenbauer. "Weight recentered barrier functions and smooth polytopic terminal set formulations for linear model predictive control". In: *2015 American Control Conference (ACC)*. 2015, pp. 1647–1652. DOI: 10.1109/ACC.2015.7170969.

[FE17] Christian Feller and Christian Ebenbauer. "Relaxed Logarithmic Barrier Function Based Model Predictive Control of Linear Systems". In: *IEEE Transactions on Automatic Control* 62.3 (2017), pp. 1223–1238. DOI: 10.1109/TAC.2016.2582040.

[Gro+16] Sebastien Gros et al. "From linear to nonlinear MPC: bridging the gap via the real-time iteration". In: *International Journal of Control* 93 (Sept. 2016), pp. 1–19. DOI: 10.1080/00207179.2016.1222553.

[HFD11] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range". In: *Automatica* 47.10 (2011), pp. 2279–2285. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2011.08.020. URL: https://www.sciencedirect.com/science/article/pii/S0005109811003918.

[KM00] Eric Kerrigan and Jan Maciejowski. "Soft Constraints And Exact Penalty Functions In Model Predictive Control". In: (Sept. 2000).

[Kou+18] Dimitris Kouzoupis et al. "Recent advances in quadratic programming algorithms for nonlinear model predictive control". In: *Vietnam Journal of Mathematics* 46.4 (Sept. 29, 2018), pp. 863–882. DOI: 10.1007/s10013-018-0311-1. URL: https://cdn.syscop.de/publications/Kouzoupis2018.pdf.

[LG17] Jürgen Pannek Lars Grüne. *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer, 2017. ISBN: 9783319460246. URL: https://link.springer.com/book/10.1007/978-3-319-46024-6.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 2e. New York, NY, USA: Springer, 2006.

[RMD17] J.B. Rawlings, D.Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017. ISBN: 9780975937730. URL: https://books.google.ch/books?id=MrJctAEACAAJ.

[Ste+20] B. Stellato et al. "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672. DOI: 10.1007/s12532-020-00179-2. URL: https://doi.org/10.1007/s12532-020-00179-2.

[Sti18] Georg Still. *Lectures on parametric optimization : an introduction*. 2018.

[WB06] Andreas Wächter and Lorenz Biegler. "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming". In: *Mathematical programming* 106 (Mar. 2006), pp. 25–57. DOI: 10.1007/s10107-004-0559-y.