

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

Tudor Oancea

June 2022

Abstract

In this project, we investigate the use of relaxed logarithmic barrier functions in the context of nonlinear model predictive control. We base our work on the one of C. Feller and C. Ebenbauer in [FE17] and use one of their globally stabilizing schemes with terminal costs and without terminal sets. We partially extend the results on the nominal asymptotic stability of the corresponding closed-loop system and the constraint satisfaction guarantees to the case of nonlinear dynamics and show that they are satisfied only in a neighborhood of the target state. The theoretical results are complemented by numerical illustrations based on Real Time Iteration.

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 a very important one 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* (OC), which aims at finding a *control law*, i.e. a way to control a system in order to attain a certain goal, by formulating and solving an optimization problem. For example, if one wants to stabilize the state around a certain target, or 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 OC is applicable, such as path tracking or optimal trajectory generation.

One of the most important algorithm of OC 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 a certain 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. More details on the upsides of RRLB functions can be found in the Introduction of [FE17].

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

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 $\mu : \mathcal{X} \rightarrow \mathcal{U}$ 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 μ , we will solve the following optimization problem :

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

A bit of vocabulary : N is called the *horizon length*, the function l is the *stage cost*, the function F is the *terminal cost* (its role will become clearer in next section) and the function $J_N(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{N-1} l(x_k, u_k) + F(x_N)$ 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

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

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 :

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

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

This general form of the MPC is the one we will use, but other variants 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 ($l(x, u) = x^T Q x + u^T R u$ and $F(x) = x^T P x$ for some positive definite matrices Q, R and P) and the (inequality) constraints will always be polytopic ($\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid C_x x \leq d_x\}$, $\mathcal{U} = \{u \in \mathbb{R}^{n_u} \mid C_u u \leq d_u\}$ for matrices C_x, C_u and vectors d_x, d_u of appropriate dimension). These are very common in real-world applications.

Other variants of MPC are:

- the infinite horizon MPC:

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

In real life applications, we consider that the duration of the experiment is virtually infinite, so this variant may seem more natural. However this optimization problem becomes infinite-dimensional so not easily solvable in practice. 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 they are usually constructed this way.

- the (*infinite-horizon*) *Linear Quadratic Regulator*, or *LQR*:

it is basically an MPC with linear dynamics ($x^+ = Ax + Bu$) and without state and control constraints. These problems are the most basic one can find and are deeply linked to (*Discrete*) *Algebraic Riccati Equations*, or *DARE*:

$$P = Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A \quad (2)$$

The unique positive definite solution of this equation actually verifies $V_\infty(x) = x^T P x$ so it can be used to define the terminal costs of the finite horizon LQR problem. This equation also provides the stabilizing state-feedback control law of the infinite horizon LQR : $\mu_{LQR}(x) = Kx$ where $K = -(R + B^T P B)^{-1} B^T P A$. It is interesting to notice that in this case, the solution P to the DARE also verifies the following equation called *Lyapunov equation* :

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

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

More details on the properties of the DARE can be found in the appendix 6.

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 $\|x(k) - x^*\| \xrightarrow[k \rightarrow \infty]{} 0$ for all $x(0)$ in a certain neighborhood of x^* or in other words if when the system starts close to x^* , it will converge to it.
- *globally attractive* if $\|x(k) - x^*\| \xrightarrow[k \rightarrow \infty]{} 0$ 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 $V : \mathcal{X} \rightarrow \mathbb{R}$ 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.

In our case, the candidate function we are going to be interested in is the objective value function V_N defined in 1.

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 $u^* = 0 \in \mathcal{U}^\circ$.

First let's 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 :

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

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] :

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

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 and that this weight-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 *RRLB* function as follows :

$$B_x(x) = \sum_{i=1}^{q_x} (1 + w_{x,i}) B_{x,i}(x) \quad (3)$$

$$\text{with } B_{x,i}(x) = \begin{cases} \log(d_{x,i}) - \log(d_{x,i} - \text{row}_i(C_x)x) & \text{if } d_{x,i} - \text{row}_i(C_x)x > \delta \\ \beta(d_{x,i} - \text{row}_i(C_x)x; \delta) & \text{otherwise} \end{cases}$$

where β 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 $0 < \delta \leq \min \{d_{x,1}, \dots, d_{x,q_x}, d_{u,1}, \dots, d_{u,q_u}\}$, 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* :

$$\begin{aligned} \tilde{V}_N(x) &= \min_{\mathbf{x}, \mathbf{u}} \tilde{J}_N(\mathbf{x}, \mathbf{u}) \\ \text{s.t. } & x_0 = x \text{ and } x_{k+1} = f(x_k, u_k), \quad k = 0, \dots, N-1 \end{aligned} \quad (4)$$

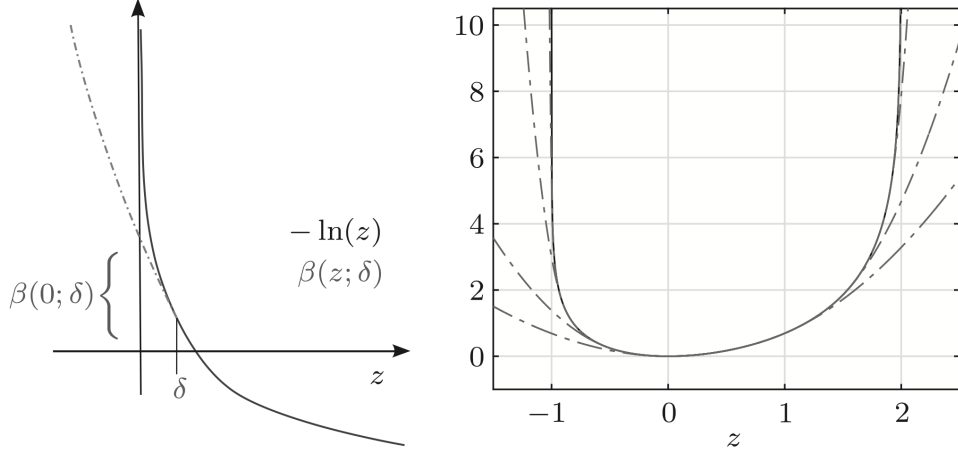


Figure 1: (Left) : Principle of relaxed log barrier function based on quadratic relaxation. (Right) : Regular weight recentered log barrier function (solid line) and RRLB function for $\delta \in \{0.01, 0.1, 0.5, 1\}$ for the constraint $z \in \mathbb{R}$, $-1 \leq z \leq 2$

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 ϵ has in theory the following interpretation : when it goes to zero, the solution of problem 4 converges to the one of 1. However, we will here consider it as fixed for simplicity.

When we consider the RRLB MPC 4 in the case of linear dynamics $x^+ = Ax + Bu$, we know the two following theorems.

Theorem 3.1. Suppose that the pair (A, B) is stabilizable¹ and that the matrix P is chosen as the unique positive definite solution to the following modified DARE :

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

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. 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 :

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

where P_{LQR} is the solution to the classical DARE 2 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 δ 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).

¹We did not discuss the notion of controllability and stabilizability in 2.1 because they do not play a direct role in our proofs. They are however very important in standard control theory and they are needed in the proofs of existence of DAREs. More on them in the appendix 6

4 Theoretical properties of RRLB Nonlinear MPC

In this section, we consider the RRLB MPC 4 with nonlinear dynamics and show that the results of the previous section hold locally, in a certain neighborhood of the origin.

From this point on, for an initial state $x \in \mathbb{R}^{n_x}$ we will denote by $\tilde{\mathbf{x}}(x) = \{\tilde{x}_0(x) = x, \tilde{x}_1(x), \dots, \tilde{x}_N(x)\}$ and $\tilde{\mathbf{u}}(x) = \{\tilde{u}_0(x), \tilde{u}_1(x), \dots, \tilde{u}_{N-1}(x)\}$ 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 4 and re-write it in a simpler way as*

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

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

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

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 Lipchitz. \square

Now our main result :

Theorem 4.2. *Let's consider the problem 4 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 $\nabla_{\mathbf{uu}}^2 J(0, \mathbf{u}(0)) = \nabla_{\mathbf{uu}}^2 J(0, 0) \succ 0$*
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. 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 certain 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 :

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{u}} \quad & \sum_{k=0}^{\infty} x_k^T (Q + \epsilon M_x) x_k + u_k^T (R + \epsilon M_u) u_k \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k, \quad k = 0, \dots, N-1 \end{aligned} \tag{5}$$

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 μ 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 certain neighborhood of the origin. First, by writing the Taylor expansion of the stage costs \tilde{l} we can see that

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

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, MPC Theory, Computation and Design] 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 :

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

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 :

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

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. \square

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 4 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 $\forall k \geq 0$:

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

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

where P_{LQR} is the solution to the DARE associated to 5 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 $\forall k \geq 0$:

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

so by summing everything we get a telescopic sum that we can compute using the fact that the system is asymptotically stable so $\lim_{k \rightarrow \infty} \tilde{V}_N(x(k)) = 0$, we get :

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

Note that 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. \square

We define for ease of notation the following quantities

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

Note that the bounds β_x, β_u depend on the relaxation parameter δ .

Lemma 4.4. For all relaxation parameter δ :

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

Proof. See Lemma 1 in [FE17]. \square

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. \square

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 its 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 describe it in the context of MPC. We don't explicitly deal with the assumptions necessary for SQP to converge.

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 :

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

we can define the following QP subproblem :

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

with the sensitivities :

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

and $H_{i,k} \approx \nabla_{(x_k, u_k), (x_k, u_k)}^2 \mathcal{L}(\mathbf{x}_i, \mathbf{u}_i)$, $H_{i,N} \approx \nabla_{x_N, x_N, u_N}^2 \mathcal{L}(\mathbf{x}_i, \mathbf{u}_i)$, 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 :

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

What's more, when the constraints are polytopic, some sensitivities become explicit :

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

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

```

1: procedure SQP( $\mathbf{x}, \mathbf{x}^{init}, \mathbf{u}^{init}$ )
2:    $\mathbf{x}_0 \leftarrow \mathbf{x}^{init}$  ▷ initial guess for the states specified by the user
3:    $\mathbf{u}_0 \leftarrow \mathbf{u}^{init}$  ▷ initial guess for the controls specified by the user
4:    $k \leftarrow 0$ 
5:   while a certain stopping criterion is not met do
6:     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
7:      $(\Delta \mathbf{x}_i, \Delta \mathbf{u}_i) \leftarrow \text{QP}(x, \mathbf{x}_i, \mathbf{u}_i)$  ▷ Find descent direction as above
8:      $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \Delta \mathbf{x}_i$ 
9:      $k \leftarrow k + 1$ 
10:  end while
11: end procedure

```

5.1.2 SQP for RRLB MPC

Using the same notations as above, the only things that change are how the sensitivities are computed :

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

5.1.3 Remarks on SQP

We can note that 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 :

- using a *locally stabilizing control law* $\kappa : \mathcal{X} \rightarrow \mathcal{U}$ (whenever one is available) and setting $u_{new} = \kappa(x_N)$. 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 estimation 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 :

²This could roughly be defined as a law that leaves the state inside the feasible set, and that does not move the state further from the reference state.

- 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 estimation 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 estimation 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 :

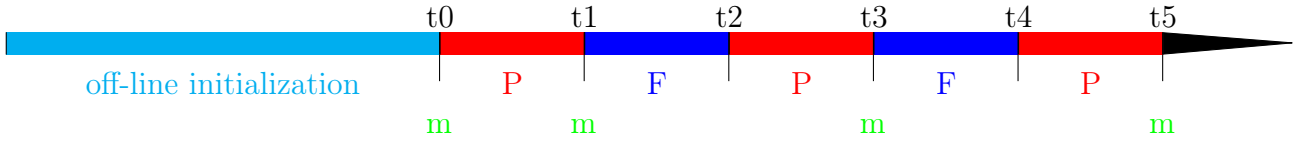


Figure 2: RTI timeline

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 solve 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 $t_1 - t_0 = dt$ and $t(2k + 3) - t(2k + 1) = dt, \forall k \geq 0$.
- 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 $[t(2k), t(2k + 2))$ 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 4 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. They 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 experimentally 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) system.

5.3.1 Experimental setup

The CSTR system describes an exothermic chemical reaction has 4 states (c_A the concentration of cyclopentadiene (reactant), c_B the concentration of cyclopentenol (product), θ the temperature in the reactor and θ_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). We don’t mention here the ODE describing the model, the constraints as well as the MPC costs for brevity purposes, but they can be found in detail in [Die01] and [HFD11]. By a simple steady state analysis procedure³, 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

Ref point	c_A^*	c_B^*	θ^*	θ_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

Some general remarks on the implementation :

- The discrete-time dynamics have been obtained from the model’s ODE by using a 6-step RK4 scheme.
- As suggested in [Die01], we chose a horizon size of 100 and a sampling time of 200.0s, which corresponds to a prediction of 2000.0s in the future. In all our experiments we set the maximum runtime to 5000.0s (simulated seconds, since the sampling time is very long, we skipped the parts where nothing happened) 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} tolerance in norm.
- 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.

³This procedure was applied to the discrete-time dynamics obtained via discretization with RK4.

- 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 problem of the general form

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

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 6.

- 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 designed the following experiments to evaluate the behavior of a given MPC scheme :

Exp1 Verify the theoretical results :

We see if the system is indeed LAS when controlled by RRLB MPC and if the constraints are violated.

Exp2 Number of iterations to convergence :

We see how quickly a scheme makes the state converge to the reference (within a tolerance of 10^{-3}).

Exp3 Compute closed-loop performance measure and compare with infinite horizon MPC :

Given by the sum of all the stage costs evaluated at every time step of the simulation. For the RTI scheme, this corresponds to the time steps when a measurement is available (see figure 2), and for the infinite horizon MPC (approximated by a 5 times bigger horizon).

Exp4 Compute the runtime of each step of RTI : i.e. sensitivities computation, condensation

These experiments were performed with both regular MPC and RRLB MPC to see the advantages of each formulation.

5.3.3 Results and discussion

Exp1 Verify the theoretical results

We can see in the figures 4, 6 and 8 in the annex that the system is indeed LAS for every reference point we have chosen. The initial point was the same in the 3 experiments

Exp2 Number of iterations to convergence

The exact closed-loop trajectories are qualitatively similar between regular MPC and RRLB MPC and can be found in the annex section 6.

Ref	Scheme	Iterations to convergence
1	rrlb	75
1	reg	152
2	rrlb	60
2	reg	251
3	rrlb	73
3	reg	100

Table 2: Number of iterations needed for convergence (within 10^{-3} tolerance) of RRLB MPC compared to regular MPC and (approximate) infinite horizon MPC

Exp3 Compute closed-loop performance measure and compare with infinite horizon MPC

Ref	Scheme	Perf
1	rrlb	1596.685
1	reg	1985.267
1	infhorz	1412.649
2	rrlb	177.769
2	reg	369.008
2	infhorz	144.5
3	rrlb	41.193
3	reg	86.323
3	infhorz	34.085

Table 3: Performance Measure of RRLB MPC compared to regular MPC and (approximate) infinite horizon MPC

Exp4 Compute the runtime of each step of RTI

Ref	Scheme	ASCT	SESCT	ACT	SECT	AST	SEST
1	rrlb	86.62	15.57	74.78	11.89	0.86	0.14
1	reg	46.19	15.66	64.37	21.42	0.74	0.83
2	rrlb	70.18	3.42	61.09	2.53	0.76	0.08
2	reg	41.74	3.38	58.31	3.98	1.02	0.12
3	rrlb	74.26	7.74	64.48	5.29	0.77	0.09
3	reg	43.4	4.27	59.91	4.04	1.06	0.19

Table 4: Runtimes of different phases of RTI for RRLB MPC compared to regular MPC and (approximate) infinite horizon MPC

where ASCT : Average Sensitivity Computation Time, SESCT : Standard Error of Sensitivity Computation Time, ACT : Average Condensation Time, SECT : Standard Error of Condensation Time, AST : Average Solving Time, SEST : Standard Error of Solving Time

5.3.4 Discussion

From the previous tables, we can notice several things :

- First of all, the system stabilizes around the reference state, as hoped after the proof of theorem 4.2. We did not try to estimate the neighborhood of the reference state starting from which this should be true because it is theoretically nearly impossible, and computationally a real burden since the only reliable method would be a naive grid search.
- We see there is indeed a violation of constraints
- Contrary to our initial hypothesis, the solve times are not shorter for RRLB MPC, which would indicate that the optimization problem is not necessarily easier to solve than for Regular MPC.
- The convergence to the objective is always taking fewer iterations with RRLB than with Regular MPC. In particular, for reference points n°2 and 3, there is no convergence for Regular MPC at the end of the 5000s of simulation.

6 Appendix

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.
- [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>.
- [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.

Controllable vs. Stabilizable

More on the DARE

Closed-loop trajectories

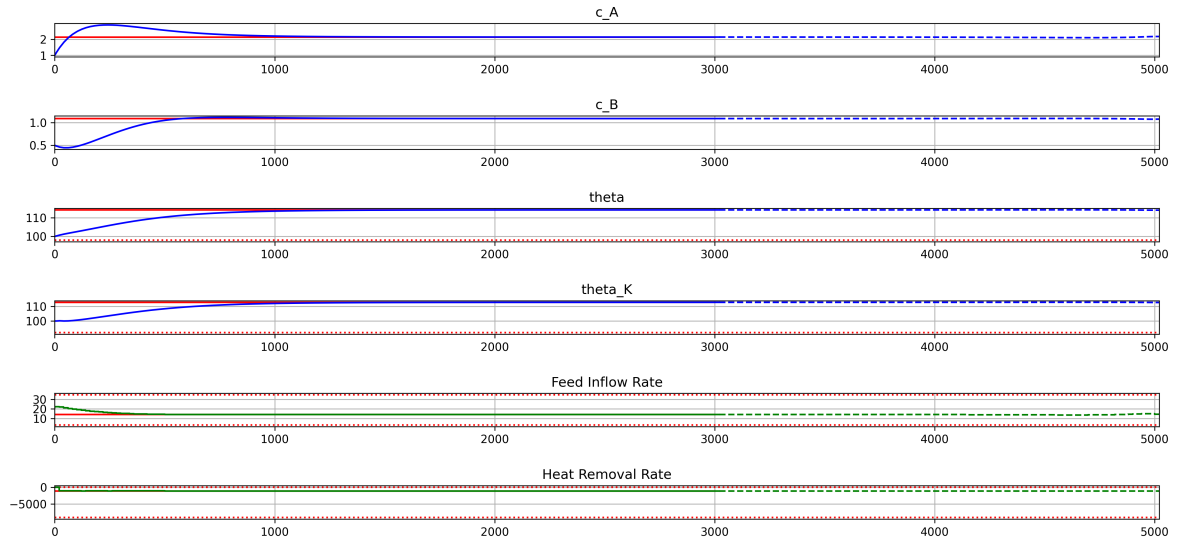


Figure 3: Closed-loop trajectories for the regular MPC with reference point n°1

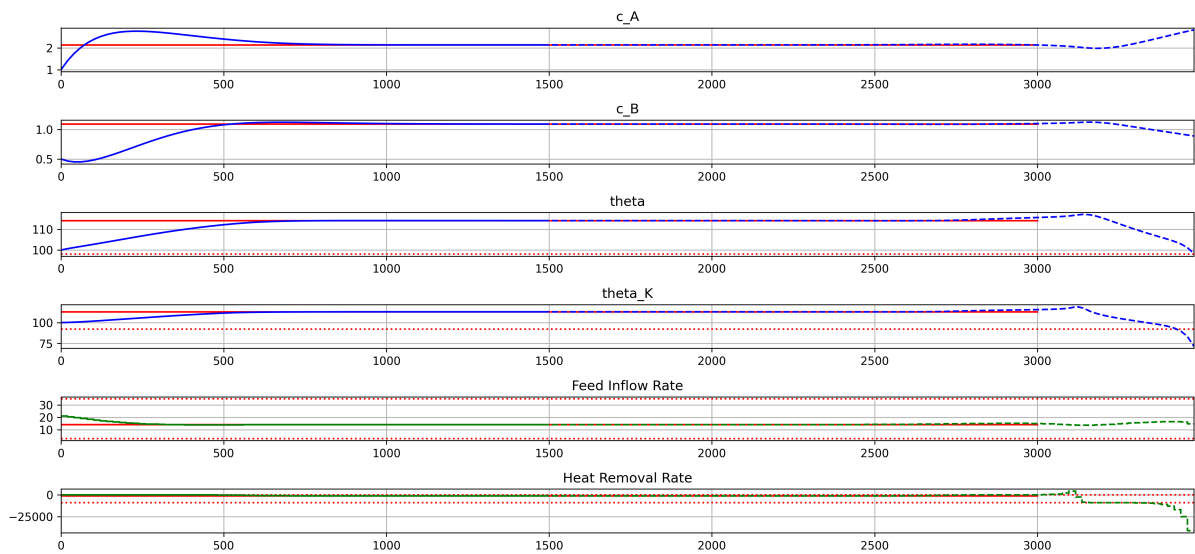


Figure 4: Closed-loop trajectories for the RRLB MPC with reference point n°1

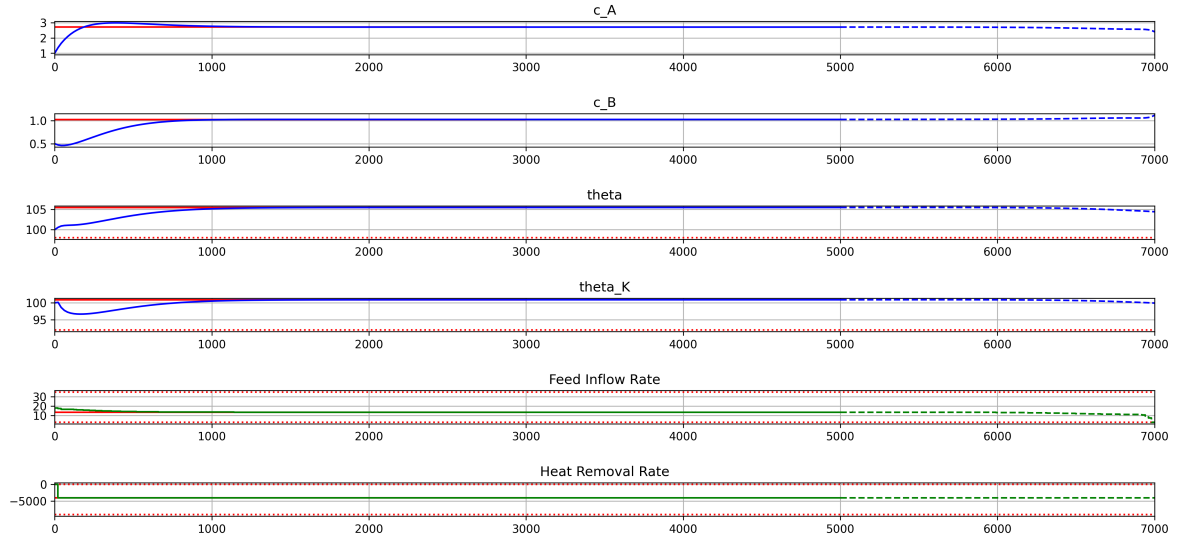


Figure 5: Closed-loop trajectories for the regular MPC with reference point n°2

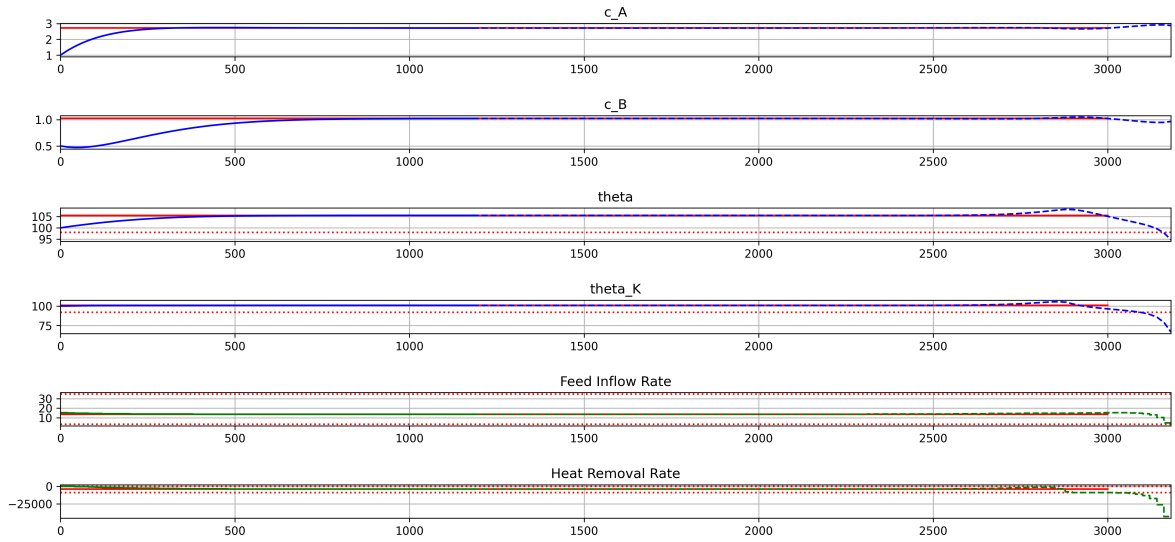


Figure 6: Closed-loop trajectories for the RRLB MPC with reference point n°2

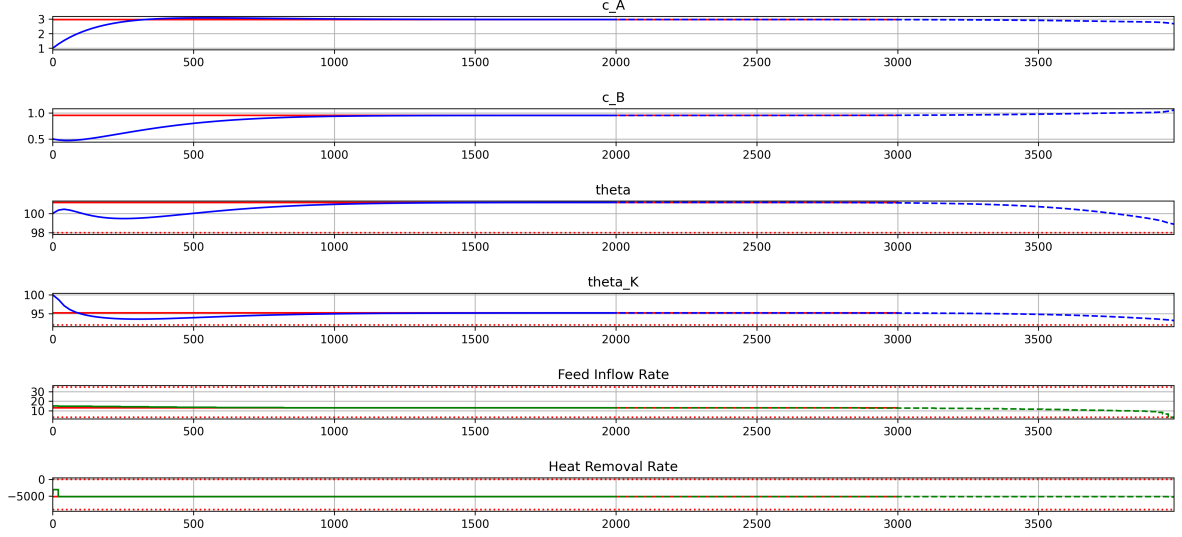


Figure 7: Closed-loop trajectories for the regular MPC with reference point n°3

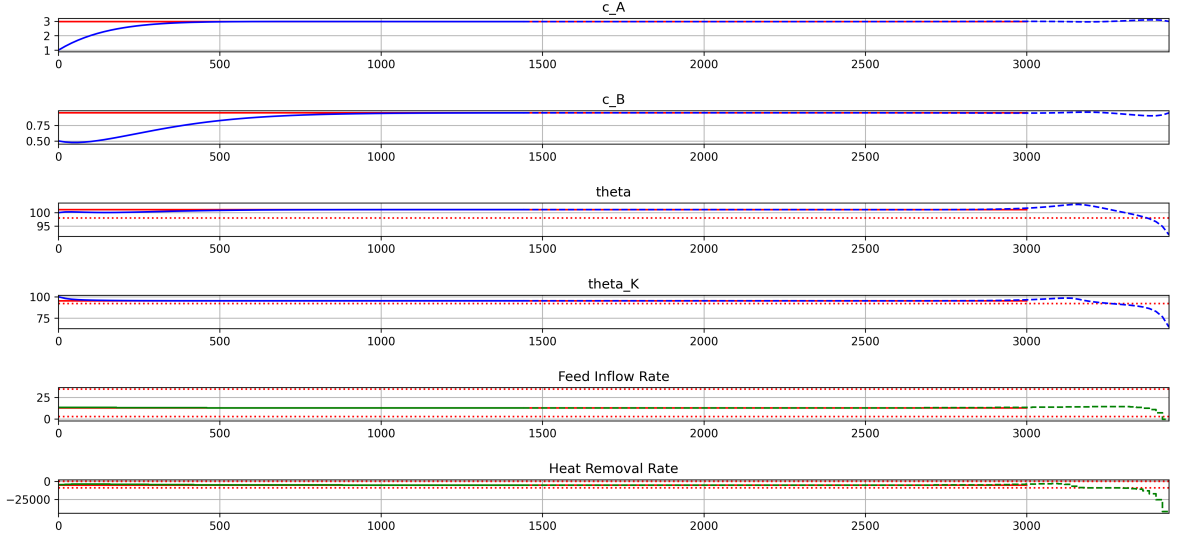


Figure 8: Closed-loop trajectories for the RRLB MPC with reference point n°3

OSQP condensation

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

vectors in the OSQP formulation will be:

$$\begin{aligned}
q_{OSQP} &= 2 \begin{pmatrix} Q & & & & \\ & \ddots & & & \\ & & Q & & \\ & & & P & \\ & & & & R \\ & & & & & \ddots \\ & & & & & & R \end{pmatrix} \begin{pmatrix} x_0^{init} \\ \vdots \\ x_N^{init} \\ u_0^{init} \\ \vdots \\ u_{N-1}^{init} \end{pmatrix} + \epsilon \begin{pmatrix} \nabla B_x(x_0^{init}) \\ \vdots \\ \nabla 0 \\ \nabla B_u(u_0^{init}) \\ \vdots \\ \nabla B_u(u_{N-1}^{init}) \end{pmatrix} \\
P_{OSQP} &= 4 \begin{pmatrix} Q & & & & \\ & \ddots & & & \\ & & Q & & \\ & & & P & \\ & & & & R \\ & & & & & \ddots \\ & & & & & & R \end{pmatrix} + 2\epsilon \begin{pmatrix} \nabla^2 B_x(x_0^{init}) & & & & \\ & \ddots & & & \\ & & 0 & & \\ & & & \ddots & \\ & & & & \nabla^2 B_u(u_0^{init}) \\ & & & & & \ddots \\ & & & & & & \nabla^2 B_u(u_{N-1}^{init}) \end{pmatrix} \\
A_{OSQP} &= \begin{pmatrix} I_{n_x} & & & & 0 \\ A_0^{init} & -I_{n_x} & & & B_0^{init} & \ddots \\ & A_1^{init} & -I_{n_x} & & & \ddots \\ & & \ddots & & & \\ & & & A_{N-1}^{init} & -I_{n_x} & B_{N-1}^{init} & 0 \\ \textcolor{red}{C}_x & & & & & & \\ & \textcolor{red}{C}_x & & & & & \\ & & \ddots & & & & \\ & & & \textcolor{red}{C}_x & & & \\ & & & & \textcolor{red}{C}_x & & \\ & & & & & \textcolor{red}{C}_u & \\ & & & & & \ddots & \\ & & & & & & \ddots \\ & & & & & & & \textcolor{red}{C}_u \end{pmatrix} \\
l_{OSQP} &= \begin{pmatrix} x - x_0^{init} \\ x_1^{init} - f(x_0^{init}, u_0^{init}) \\ \vdots \\ x_N^{init} - f(x_{N-1}^{init}, u_{N-1}^{init}) \\ -\infty \\ \vdots \\ -\infty \end{pmatrix} \quad u_{OSQP} = \begin{pmatrix} x - x_0^{init} \\ x_1^{init} - f(x_0^{init}, u_0^{init}) \\ \vdots \\ x_N^{init} - f(x_{N-1}^{init}, u_{N-1}^{init}) \\ \textcolor{red}{d}_x - \textcolor{red}{C}_x x_0^{init} \\ \vdots \\ \textcolor{red}{d}_x - \textcolor{red}{C}_x x_N^{init} \\ \vdots \\ \textcolor{red}{d}_u - \textcolor{red}{C}_u u_0^{init} \\ \vdots \\ \textcolor{red}{d}_u - \textcolor{red}{C}_u u_{N-1}^{init} \end{pmatrix}
\end{aligned}$$

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

Misc