sNMPC

Citation for published version (APA):
Eyüboglu, M., & Lazar, M. (2022). sNMPC: A Matlab Toolbox for Computing Stabilizing Terminal Costs and Sets. *IFAC-PapersOnLine*, *55*(30), 19-24. https://doi.org/10.1016/j.ifacol.2022.11.022

Document license:
CC BY-NC-ND

DOI:
10.1016/j.ifacol.2022.11.022

Document status and date:
Published: 01/01/2022

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 04. Oct. 2023

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 55-30 (2022) 19–24

sNMPC: A Matlab Toolbox for Computing Stabilizing Terminal Costs and Sets

Mert Eyüboğlu * **Mircea Lazar** *

* *Eindhoven University of Technology, Eindhoven, The Netherlands (e-mails: m.eyuboglu@student.tue.nl, m.lazar@tue.nl).*

Abstract: This paper presents a Matlab toolbox that implements methods for computing stabilizing terminal costs and sets for nonlinear model predictive control (NMPC). Given a discrete–time nonlinear model provided by the user, the toolbox computes quadratic/ellipsoidal terminal costs/sets and local control laws for the following options: (i) cyclically time–varying or standard terminal ingredients; (ii) first or quasi–second order Taylor approximation of the dynamics; (iii) linear or nonlinear local control laws. The YALMIP toolbox and the MOSEK solver are used for solving linear matrix inequalities and the IPOPT solver (with global search) is used for nonlinear programming. Simulation of the resulting stabilizing NMPC algorithms is provided using the CasADi toolbox.

Copyright © 2022 The Authors. This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

Keywords: Stability of nonlinear systems, nonlinear predictive control, terminal costs and sets.

1. INTRODUCTION

The most common constructive approach for guaranteeing stability within the nonlinear model predictive control (NMPC) framework is to introduce stabilizing terminal ingredients (i.e., terminal costs and sets). Typically, the terminal set and the terminal cost are chosen as an invariant set and a control Lyapunov function, respectively. It is already possible to compute these terminal ingredients via the MPT3 toolbox (Herceg et al., 2013) for linear time–invariant, piecewise affine and mixed logical dynamical systems. However, computation of terminal ingredients for nonlinear systems remains as an open problem of interest.

The most common approach is to first compute terminal ingredients for linearized dynamics and then compensate for the approximation error. Such methods for continuous–time systems can be found in (Michalska and Mayne, 1993), (Chen and Allgöwer, 1998) and (Chen et al., 2003). A discrete–time equivalent of the same correction method was worked out in (Kwon and Han, 2005) and (Yu et al., 2015). An improvement of this method for continuous–time and discrete–time systems was presented in (Rajhans et al., 2016) and (Rajhans et al., 2019), respectively, by introducing additional degrees of freedom for shaping the terminal sets. Other approaches, (Hu and Chen, 2007) and (Darup and Cannon, 2015), use linear difference inclusions (LDIs) to approximate the nonlinear dynamics. Another method for continuous–time NMPC (Lucia et al., 2015) uses higher order Taylor series expansions and sum–of–squares techniques to further alleviate conservatism. In (Lazar and Tetteroo, 2018) a framework for computing cyclically time–varying terminal ingredients for discrete–time NMPC was developed based on finite–step control Lyapunov functions (Lazar and Spinu, 2015). (Fitri and Kim, 2020) presented a method for increasing the domain of attraction of NMPC by combining multiple ellipsoidal terminal sets.

Following the advances in nonlinear programming (NLP), toolboxes such as MATMPC (Chen et al., 2019) or the Matlab MPC Toolbox now have modules for efficient implementation of NMPC. In addition to these, it is possible to define and solve NMPC problems with optimization toolboxes such as YALMIP (Löfberg, 2004), CasADi (Andersson et al., 2018), ACADO (Houska et al., 2011), etc. However, these toolboxes do not feature methods for computing stabilizing terminal ingredients for NMPC.

This paper presents a Matlab toolbox for computing stabilizing terminal ingredients for discrete–time NMPC, which builds on the framework of (Lazar and Tetteroo, 2018). The stabilizing NMPC (sNMPC) toolbox computes ellipsoidal terminal sets and quadratic terminal cost functions for NMPC with the following design choices: (i) cyclically time–varying or standard terminal ingredients; (ii) first or quasi–second order Taylor approximation of the nonlinear dynamics; (iii) linear or nonlinear local control laws. The sNMPC toolbox utilizes the following toolboxes and solvers: YALMIP (Löfberg, 2004) for defining optimization problems; MOSEK (ApS, 2019) for solving LMIs; (Wächter and Biegler, 2006) with a multistart method for solving nonlinear optimization problems. OPTI Toolbox (Currie and Wilson, 2012) is used for enabling the interface between YALMIP and IPOPT. The Ellipsoidal Toolbox (Kurzhanskiy and Varaiya, 2006) is used for plotting the ellipsoidal terminal sets as well as for computing their volumes. Furthermore, the polytopic library of MPT3 (Herceg et al., 2013) is used for plotting polytopic admissible sets. The CasADi nonlinear optimization toolbox (Andersson et al., 2018) is utilized for simulating the stabilizing NMPC algorithms.

2. THEORETICAL FOUNDATION

Consider a discrete–time nonlinear system

$$x(k+1) = \phi(x(k), u(k)), \quad k \in \mathbb{N}, \tag{1}$$

2405-8963 Copyright © 2022 The Authors. This is an open access article under the CC BY-NC-ND license.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2022.11.022

where $\phi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the system dynamics, $x(k)$ is the state and $u(k)$ is the input at time instant k. It is assumed that $\phi(0,0) = 0$. The NMPC optimization problem with a prediction horizon N is:

$$\min_{U_k} J(x_{0|k}, U_k) = \min_{U_k} \{F(x_{N|k}, k) + \sum_{i=0}^{N-1} l(x_{i|k}, u_{i|k})\}, \tag{2a}$$

subject to:

$$x_{i+1|k} = \phi(x_{i|k}, u_{i|k}), i \in \{0, \ldots, N-1\}, \tag{2b}$$
$$x_{i|k} \in \mathbb{X} \subseteq \mathbb{R}^n, i \in \{0, \ldots, N-1\}, \tag{2c}$$
$$u_{i|k} \in \mathbb{U} \subseteq \mathbb{R}^m, i \in \{0, \ldots, N-1\}, \tag{2d}$$
$$x_{N|k} \in \mathbb{X}_T(k) \subseteq \mathbb{X}, \tag{2e}$$

where $x_{0|k} = x(k)$, $u_{0|k} = u(k)$, $F : \mathbb{R}^n \times \mathbb{N} \to \mathbb{R}_{\geq 0}$ is the terminal cost, $\mathbb{X}_T$ is the terminal set and $l : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ is the stage cost. Note that the terminal cost and the terminal set are allowed to be time–varying. It is assumed that $l(0,0) = 0$ and $l(x, \cdot) \geq \alpha_3(||x||)$, for all $x \in \mathbb{X}$ where α_3 is a $\mathcal{K}_\infty$ class function.

For any finite $M \in \mathbb{N}_{\geq 1}$, let $\{(F_j(\cdot), \mathcal{S}_j, h_j(\cdot))\}_{j=0,\ldots,M-1}$ denote a set of terminal ingredients, where $F_j : \mathbb{R}^n \to \mathbb{R}_+$ with $F_j(0) = 0$ are terminal costs, $\mathcal{S}_j$ are compact terminal sets that contain the origin in their interior and $h_j : \mathbb{R}^n \to \mathbb{R}^m$ with $h_j(0) = 0$ are terminal control laws for all $j = 0, \ldots, M-1$. Define $\Phi_j(x) := \phi(x, h_j(x))$ and for a subset $\mathcal{S}$ of $\mathbb{R}^n$ define $\Phi_j(\mathcal{S}) := \{\Phi_j(x) \ : \ x \in \mathcal{S}\}$. Let the terminal ingredients satisfy the conditions given in Assumption 1 of (Lazar and Tetteroo, 2018). Theorem 3 of (Lazar and Tetteroo, 2018) establishes that the MPC problem (2) is recursively feasible and the corresponding closed–loop system is asymptotically stable for all $x(0) \in \mathbb{X}_f(N) \subseteq \mathbb{X}$ for such a design of terminal ingredients.

Then, following Assumption 1 of (Lazar and Tetteroo, 2018), the time–varying terminal costs/sets are defined as:

$$\text{If } k = 0 \ \& \ \mathbb{X}_T(0) = \mathcal{S}_j, \quad \forall j = 0, \ldots, M-1$$
$$\Rightarrow \mathbb{X}_T(k) := \mathcal{S}_{(j+k)\text{mod}M}, \tag{3a}$$
$$\Rightarrow F(x, 0) = F_j(x) \ \& \ F(x, k) = F_{(j+k)\text{mod}M}(x). \tag{3b}$$

The initial terminal set and cost pair to be used at time $k = 0$ must be determined by solving an optimization problem. Note that for $M = 1$, the original design with a single invariant terminal set, terminal cost and local control law is recovered. Using multiple control laws and terminal cost functions introduces a degree of freedom for satisfying the required assumptions, which allows for a larger domain of attraction when compared to being restricted to a single control law and terminal cost.

Computation of Terminal Ingredients
Let $l(x,u) = x^T Q x + u^T R u$ with $Q \succ 0, R \succ 0$. Set $F_j(x) := x^T P_j x$ with $P_j \succ 0$ and $\mathcal{S}_j := \{x \ : \ F_j(x) \leq \alpha\}$ with $\alpha > 0$ and set $h_j(x) := K_j x, K_j \in \mathbb{R}^{n \times m}$ for all $j = 0, \ldots, M-1$. Next, define the approximation error

$$r_j(x) := \phi(x, K_j x) - (A + BK_j)x = \Phi_j(x) - (A + BK_j)x, \tag{4}$$

for all $j = 0, \ldots, M-1$. By defining $A_{K_j} := A + BK_j$, and $\Phi_j(x) := A_{K_j}x + r_j(x)$, the inequalities in Assumption 1-(ii) of (Lazar and Tetteroo, 2018) can be written as

$$(A_{K_j}x + r_j(x))^T P_{j+1}(A_{K_j}x + r_j(x)) - x^T P_j x + x^T(Q + K_j^T R K_j)x \leq 0. \tag{5}$$

For $\mathcal{S}_j := \{x \ : \ x^T P_j x \leq 1\}$ and $F_j(x) = \frac{1}{\alpha}x^T P_j x$, the inequality (5) can be split into the LMI

$$\begin{bmatrix} (1-\kappa_j)O_j & (AO_j + BY_j)^T & O_j & Y_j^T \\ AO_j + BY_j & O_{j+1} & 0 & 0 \\ O_j & 0 & \bar{\alpha}Q^{-1} & 0 \\ Y_j & 0 & 0 & \bar{\alpha}R^{-1} \end{bmatrix} \succeq 0, \tag{6}$$

and the nonlinear inequality

$$\mathcal{R}_j(x) := r_j(x)^T P_{j+1} r_j(x) + 2x^T A_{K_j}^T P_{j+1} r_j(x) - \kappa_j x^T P_j x \leq 0, \tag{7}$$

where $0 < \kappa_j < 1$, $\bar{\alpha} = \frac{1}{\alpha}$, $O_j = P_j^{-1}$, $Y_j = K_j P_j^{-1}$ with $P_j \succ 0$ for all $j = 0, \ldots, M-1$.

Remark 1. In (Lazar and Tetteroo, 2018), after introducing α into the LMIs the terminal set is defined incorrectly as $\mathcal{S}_j := \{x \ : \ F_j(x) \leq 1\}$ throughout the paper with $F(x) = \frac{1}{\alpha}x^T P x$. Hereby, the terminal set definition is corrected as $\mathcal{S}_j := \{x \ : \ x^T P_j x = \alpha F_j(x) \leq 1\}$.

Note that (6) and (7) only have to hold for all $x \in \mathcal{S}_j$. Thus, to find a solution to the original terminal cost inequalities in Assumption 1 of (Lazar and Tetteroo, 2018), it is sufficient to find a solution to (6) and to check if $\max_{x \in \mathcal{S}_j} \mathcal{R}_j(x) \leq 0$. The volume of the resulting terminal sets can be maximized by maximizing over $\log\det(O_j)$ while solving the LMIs (6). The resulting feedback gains $\{K_j\}_{\{j=0,\ldots,M-1\}}$ will yield a stable (monodromy) matrix $A_{K_{M-1}} A_{K_{M-2}} \ldots A_{K_0}$. After solving (6), $\max_{x \in \mathcal{S}_j} \mathcal{R}_j(x) \leq 0$ can be checked by using a NLP solver. If the check fails, $\mathcal{S}_j$ must be appropriately scaled down.

A quasi–second order Taylor approximation of $\phi(\cdot,\cdot)$ can also be used, which requires modification of the LMIs in (6), as defined in equation (9) of (Lazar and Tetteroo, 2018). For input affine nonlinear systems (i.e., $\phi(x,u) = f(x) + g(x)u$) we can further reduce conservatism by employing auxiliary nonlinear control laws as defined in Proposition 1 of (Raimondo et al., 2009), i.e.,

$$h_j(x) := -(g^T(x)P_j g(x) + \alpha R)^{-1} g^T(x) P_j f(x). \tag{8}$$

This requires the following update:

$$r_j(x) := \phi_j(x, h_j(x)) - (A + BK_j)x, \tag{9}$$

where $r_j(x) = r_{1,j}(x) + g(x) r_{2,j}(x)$ with

$$r_{1,j}(x) := f(x) + g(x)K_j x - (A + BK_j)x,$$

and $r_{2,j}(x) := h_j(x) - K_j x$. In this case (5) must be replaced by

$$(A_{K_j}x + r_j(x))^T P_{j+1}(A_{K_j}x + r_j(x)) - x^T P_j x + x^T Q x + x^T(K_j + r_{1,j}(x))^T R(K_j + r_{2,j}(x)) \leq 0. \tag{10}$$

This yields no changes in the LMIs while the corresponding nonlinear inequality becomes

$$\mathcal{R}_j(x) := r_j(x)^T P_{j+1} r_j(x) + 2x^T A_{k_j}^T P_{j+1} r_j(x) + r_{2,j}(x)^T \alpha R r_{2,j}(x) + 2x^T K_j^T \alpha R r_{2,j}(x) - \kappa_j x^T P_j x \leq 0. \tag{11}$$

In summary, the key tuning knobs are: (i) the number M of terminal ingredients, (ii) the order of the approximation of the nonlinear dynamics (first or quasi–second order), and (iii) linear or nonlinear control laws. Additionally, the parameters κ_j can be optimized.

3. TOOLBOX IMPLEMENTATION IN MATLAB

The sNMPC toolbox consists of several Matlab functions which are classified under three modules enabling different functionalities as depicted in Fig. 1.

Computation of terminal ingredients Module
This module yields periodically time–varying terminal ingredients consisting of terminal sets S_j and their corresponding terminal costs $F_j(x)$ for all $j = 0, \dots, M-1$. To use this module, one must first define the design choices such as M, the type of approximation and control law, κ_j, ρ, η and the number of gridpoints as well as the MPC cost parameters Q and R and the prediction horizon N. Afterwards, the following functions can be employed:

- `get_ABHessian()`: Computes the Jacobian and Hessian matrices used for the first or quasi–second order approximation of the system dynamics.
- `solve_LMIs()`: Solves the set of LMIs in (6), (13), (14) and (15). The LMIs are defined using YALMIP (Löfberg, 2004) and solved with MOSEK (ApS, 2019) while maximizing over $\log(\det(O_j))$, where $j = \max(1, \lfloor \frac{M}{2} \rfloor)$ is used as in (Lazar and Tetteroo, 2018).
- `solve_NLP_bisection()`: Checks if $\max_{x \in \mathcal{S}_j} \mathcal{R}_j(x) \leq 0$ holds for the candidate terminal sets and scales down the sets by bisection until the check is satisfied. The NLP is again defined using YALMIP (Löfberg, 2004) and solved with IPOPT (Wächter and Biegler, 2006) using a multistart approach.

After calling the above 3 functions, the terminal ingredients corresponding to the selected design choices are obtained.

A set of state and input polytopic constraints can be provided by the user, i.e., $\mathbb{X} = \{x : H_x x \leq 1\}$ and $\mathbb{U} = \{u : H_u \leq 1\}$. Then letting $[H_x]_{p:}$ denote the p-th line of H_x and $[H_u]_{p:}$ denote the p-th line of H_u, the inequalities

$$[H_x]_{p:} O_j [H_x]_{p:}^T \leq 1, \quad \forall p : 1, \dots, n_x, \quad \forall j \in \{0, \dots, M-1\}, \tag{12}$$

ensure that $\mathcal{S}_j \subseteq \mathbb{X}$ holds for $\mathcal{S}_j = \{x : x^T P_j x \leq 1\}$, while the inequalities

$$\begin{bmatrix} 1 & [H_u]_{p:} Y_j \\ Y_j^T [H_u]_{p:}^T & O_j \end{bmatrix} \geq 0, \quad \forall p : 1, \dots, n_u, \quad \forall j \in \{0, \dots, M-1\}, \tag{13}$$

ensure that $h_j(\mathcal{S}_j) \subseteq \mathbb{U}$ holds for $\mathcal{S}_j = \{x : x^T P_j x \leq 1\}$. The above LMIs are included in the function `solve_LMIs()`.

If a quasi–second order Taylor approximation is chosen, the sets $\mathcal{S}_j$ must lie within a bounding polytope $\mathcal{P} \subseteq \mathbb{X}$ with set of vertices $\mathcal{P}_v$. To ensure this, define $\mathcal{P}_v := \eta\mathbb{X} = \{x : H_{\mathcal{P}} x \leq 1\}$ for $0 < \eta \leq 1$ and let $[H_{\mathcal{P}}]_{p:}$ denote the p-th line of the matrix $H_{\mathcal{P}}$ with $n_{\mathcal{P}}$ rows. Then the LMIs

$$[H_{\mathcal{P}}]_{p:} O_j [H_{\mathcal{P}}]_{p:}^T \leq 1, \ \forall p : 1, \dots, n_{\mathcal{P}}, \ \forall j \in \{0, \dots, M-1\}, \tag{14}$$

included in the function `solve_LMIs()` ensure $\mathcal{S}_j \subset \mathcal{P}$.

Since (14) is equivalent to (12) for $\eta = 1$, only (14) is used in all cases where $\eta = 1$ if a first order Taylor approximation is chosen and the user determined η value is used otherwise. If the LMIs end up being infeasible when a quasi–second order approximation is used, the bounding polytope $\mathcal{P}_v := \eta\mathbb{X}$ is further scaled down by decreasing η, in steps of size $\eta/30$, until the LMIs become feasible.

Table 1. Toolbox functions and their corresponding inputs and outputs.

Function	Inputs	Outputs
`get_ABHessian()`	`sys`	`sys`
`solve_LMIs()`	`sys, p, Mode, opt_L`	`P, K,` α`, E1, VOL1, XUset, Xset_scaled`
`solve_NLP_bisection()`	`sys, p, P, K, alpha, Mode, opt_NL`	`alpha, E2, VOL2`

Notice that while computing cyclically time–varying terminal sets, variance within the terminal sets orientation is enforced by adding the constraint

$$Y_{j+1} \geq \rho Y_j, \quad \forall j \in \{0, \dots, M-2\}, \tag{15}$$

where $\rho > 1$ is a scaling factor.

Remark 2. For the sake of generalization, the approximation error is defined as $r_j(x) := r_{1,j}(x) + g(x) r_{2,j}(x)$ with

$$r_{1,j}(x) := f(x) + g(x)K_j x - (\bar{A}(x) + \bar{B}(x)K_j)x,$$

and $r_{2,j}(x) = h_j(x) - K_j x$ in all cases. Thus, the nonlinear inequality (11) is considered always while it is equivalent to (7) if $r_{2,j}(x) = 0$.

To solve the NLP porblem (11) with a multistart method, for all $j = 0, \dots, M-1$ a certain number of gridpoints, defined by the user, inside the terminal set $\mathcal{S}_j = \{x : x^T P_j x \leq 1\}$ are selected randomly with a uniform distribution employing the algorithm presented in (Dezert and Musso, 2001). These gridpoints are used as initial guesses for the multistart optimization. If $\max_{x \in \mathcal{S}_j} \mathcal{R}_j(x) \leq 0$ is violated for any of the gridpoints for some j, the terminal sets are scaled down by bisection with a binary search algorithm until the check is satisfied for all initial points for all $j = 0, \dots, M-1$ and the scaling factor $\gamma \leq 1$ is returned. Using the obtained scaling factor γ the terminal sets are updated as $\mathcal{S}_j = \{x : x^T P_j x \leq \gamma\}$. The bisection tolerance can be determined by the user.

The inputs and outputs of the functions in this module are presented in Table 1. There, `sys` contains the system dynamics and constraints, `p` contains the user defined parameters which are $\kappa_j, M, N, QR, \rho, \eta$, number of gridpoints used in nonlinear optimization, bisection tolerance and maximum scaling during bisection, `opt_L` and `opt_NL` are the linear and nonlinear solvers respectively, `Mode` is the type of approximation and control law chosen by the user, `XUSet` and `Xset_scaled` denote polytopic boundaries used for plotting while `P,K,E1,VOL1,E2` and `VOL2` denote P_j, K_j and ellipsoidal sets and their volumes for all $j = 0, \dots, M-1$.

Simulation Module
This module delivers the resulting closed–loop state trajectories for the sNMPC algorithms with terminal ingredients. This is enabled by the following functions which use the CasADi toolbox (Andersson et al., 2018) for solving the NMPC optimization problems:

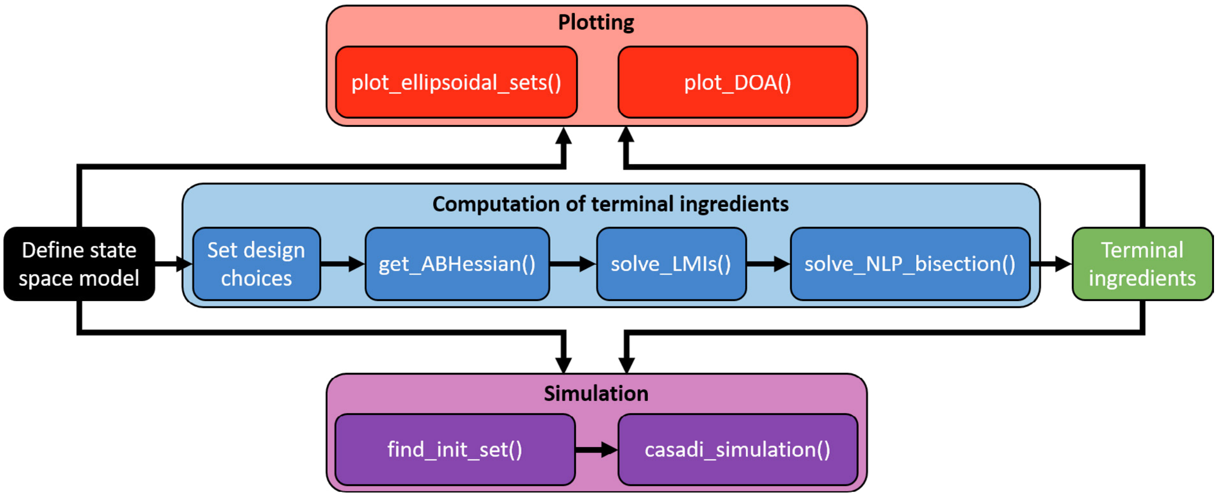

Fig. 1. Flowchart of the sNMPC toolbox functions.

- `find_init_set()`: Checks the feasibility of an initial condition $x(0)$ and determines the optimal terminal cost and set to be used at $k = 0$ (if $M > 1$ is chosen). This is done by solving (19) and determining if the problem is feasible and in which terminal set the terminal state $x_{N|0}$ lies in. If $x_{N|0}$ lies inside more than one terminal set, for the sake of optimality, the terminal set resulting in the smallest cost is chosen.
- `casadi_simulation()`: Simulates the closed–loop system by solving the optimization problem (20) and applying the first control input, $\forall k \in \mathbb{N}$.

For M ellipsoidal terminal sets defined by $\mathcal{S}_j = \{x : x^T P_j x \leq 1\}$, $P_j \succ 0$, $j = 0, \dots, M-1$, define

$$\mathcal{S}_\lambda := \{x \ : \ x^T P_\lambda x \leq 1\}, \tag{16}$$

where

$$P_\lambda := \sum_{j=0}^{M-1} \lambda_j P_j, \text{ and } \sum_{j=0}^{M-1} \lambda_j = 1, \tag{17}$$

and $\lambda_j \geq 0$ for all $j = 0, \dots, M-1$. In (Fitri and Kim, 2020), it is shown that

$$\mathcal{S}_\lambda \subset \cup_{j=0}^{M-1} \mathcal{S}_j, \tag{18}$$

and for a proper choice of λ_j, $\mathcal{S}_\lambda$ can cover any point that is inside the union $\cup_{j=0}^{M-1} \mathcal{S}_j$. (18) implies that any point inside the union $\cup_{j=0}^{M-1} \mathcal{S}_j$ also belongs to at least one of the terminal sets $\mathcal{S}_j$, $j = 0, \dots, M-1$. Hence, by solving a single optimization problem with $\mathcal{S}_\lambda$ as the terminal set,

$$\min_{U_0} J(x_{0|0}, U_0) = \min_{U_0} \Big\{ \frac{1}{\alpha} x_{N|0}^T P_\lambda x_{N|0} + \sum_{i=0}^{N-1} l(x_{i|0}, u_{i|0}) \Big\}, \tag{19a}$$

subject to:

$$\begin{aligned} x_{i+1|0} &= \phi(x_{i|0}, u_{i|0}), i \in \{0, \dots, N-1\}, && (19b)\\ x_{i|0} &\in \mathbb{X} \subseteq \mathbb{R}^n, i \in \{0, \dots, N-1\}, && (19c)\\ u_{i|0} &\in \mathbb{U} \subseteq \mathbb{R}^m, i \in \{0, \dots, N-1\}, && (19d)\\ x_{N|0} &\in \mathcal{S}_\lambda, && (19e)\\ \sum_{j=0}^{M-1} \lambda_j &= 1, \ \lambda_j \geq 0, j \in \{0, \dots, N-1\}, && (19f) \end{aligned}$$

we can determine the initial terminal set. For a feasible initial condition $x_{0|0} = x(0)$, let $\mathcal{S}_{j_0^*}$ denote the terminal set with the lowest corresponding terminal cost $\frac{1}{\alpha} x_{N|k}^T P_{j_0^*} x_{N|k}$ such that $x_{N|0} \in \mathcal{S}_{j_0^*}$. Consequently, for any $k \in \mathbb{N}$, there exists a $j \in \{0, \dots, M-1\}$ such that $j = (j_0^* + k) \mathrm{mod} M$. Thus, for all $k \in \mathbb{N}$, the NMPC optimization problem is defined as

$$\min_{U_k} J(x_{0|k}, U_k) = \min_{U_k} \Big\{ \frac{1}{\alpha} x_{N|k}^T P_j x_{N|k} + \sum_{i=0}^{N-1} l(x_{i|k}, u_{i|k}) \Big\}, \tag{20a}$$

subject to:

$$\begin{aligned} x_{i+1|k} &= \phi(x_{i|k}, u_{i|k}), i \in \{0, \dots, N-1\}, && (20b)\\ x_{i|k} &\in \mathbb{X} \subseteq \mathbb{R}^n, i \in \{0, \dots, N-1\}, && (20c)\\ u_{i|k} &\in \mathbb{U} \subseteq \mathbb{R}^m, i \in \{0, \dots, N-1\}, && (20d)\\ x_{N|k} &\in \mathcal{S}_j, \ j = (j_0^* + k) \mathrm{mod} M. && (20e) \end{aligned}$$

CasADi Implementation

The optimal control problems in `find_init_set()` and `casadi_simulation()` are implemented and solved via CasADi (Andersson et al., 2018). Thus, to use these functions one must redefine the state–space model using the `SX` data type symbolic variables, instead of the Matlab `sym` data type. Then, the components of the optimization problem are defined (by the toolbox) as follows:

$$\text{Decision variables} = \begin{cases} \begin{bmatrix} u_{0|k} & \dots & u_{N-1|k} & \lambda_0 & \dots & \lambda_{M-1} \end{bmatrix}, & \text{for (19)},\\ \begin{bmatrix} u_{0|k} & \dots & u_{N-1|k} \end{bmatrix}, & \text{for (20)}, \end{cases}$$

$$\text{Constraints vector} := \begin{cases} \begin{bmatrix} x_{1|k} & \dots & x_{N|k} \\ & x_{N|k} P_\lambda x_{N|k} & \sum_{j=0}^{M-1} \lambda_j \end{bmatrix}, & \text{for (19)},\\ \begin{bmatrix} x_{1|k} & \dots & x_{N|k} & x_{N|k} P_j x_{N|k} \end{bmatrix}, & \text{for (20)}, \end{cases}$$

$$\text{Cost function} := \begin{cases} \frac{1}{\alpha} x_{N|k}^T P_\lambda x_{N|k} + \sum_{i=0}^{N-1} l(x_{i|k}, u_{i|k}), & \text{for (19)},\\ \frac{1}{\alpha} x_{N|k} P_j x_{N|k} + \sum_{i=0}^{N-1} l(x_{i|k}, u_{i|k}), & \text{for (20)}, \end{cases}$$

for all $j = 0, \dots, M-1$. Then, M different NMPC problems are defined using the above components which are solved in a periodic manner as previously explained. The upper and lower limit vectors for the constraints vector and the decision variables are constructed according to the defined state and input constraints as well as $\sum_{j=0}^{M-1} \lambda_j = 1$ and $0 \leq \lambda_j \leq 1$ if (19) is of concern. Then the NMPC problem is solved using IPOPT (Wächter and Biegler, 2006), either once for $k = 0$ (19) or within a for loop for $k \in \mathbb{N}$ (20). A shifted version of the optimal solution obtained at $k-1$, i.e., $\{u_{1|k-1}, \dots, u_{N-1|k-1}, h_j(x_{N|k-1})\}^T$ is provided to the solver as an initial guess/warm start.

Plotting Module This module provides functions for plotting closed–loop trajectories and estimating the domain of attraction of the resulting NMPC controllers, as follows:

- `plot_ellipsoidal_sets()`: Plots the obtained terminal sets by making use of the Ellipsoidal Toolbox (Kurzhanskiy and Varaiya, 2006). The polytopic state boundaries implied by the state constraints are plotted as well by using MPT3 (Herceg et al., 2013). For high dimensional systems, projections of the terminal sets on 2D planes are plotted instead.
- `plot_DOA()`: Obtains a grid of feasible and infeasible initial conditions which provides an approximation of the domain of attraction for the NMPC controller. The feasibility of any initial condition $x(0)$ is determined by calling `find_init_set()` for each grid point to check if (19) is feasible. By calling `plot_ellipsoidal_sets()`, it plots a domain of attraction representation of the closed–loop system on top of the obtained terminal sets. This plot is only available for 1D/2D systems, but the number of feasible and infeasible gridpoints are also returned for numerical comparison when higher dimensional systems are of concern.

4. ILLUSTRATIVE EXAMPLES

Consider an inverted pendulum example (Iles et al., 2015):

$$\phi_1(x,u) = x_1 + T_s mgL\sin(x_2) + C_m T_s u,$$
$$\phi_2(x,u) = x_2 + T_s x_1,$$

with $T_s = 0.2$s, $C_m = 14$, $mgL = 2$. The state and input constraints are $\|x\|_\infty \leq 10$ and $|u| \leq 2$. Let $Q = (0.05)I_2$, $R = 0.1$ and in (6) set $\bar{\alpha} \leq 10^6$. With the design parameters set as $\eta = 1$, $\rho = 2$, $\kappa_j = \kappa = 0.05$ and $M = 10$, a quasi–second order approximation and a nonlinear local control law are chosen. `solve_LMIs()` yields the candidate terminal ingredients where the P and K matrices corresponding to the largest terminal set are

$$P_5 = \begin{bmatrix} 0.010557876915815 & 0.002426931164891 \\ 0.002426931164891 & 0.010557877055499 \end{bmatrix},$$

$$K_5 = [-0.075334772611226 \;\; -0.203393867313070],$$

with $\alpha = 4.742387356691633 \times 10^{-6}$. `solveNLPbisection()` was then called for these candidate terminal ingredients which certified that (11) was already satisfied and $\gamma = 1$ was returned. The volume of the largest terminal set is obtained as $3.057465536528358 \times 10^2$. The resulting $M = 10$ terminal sets are plotted in Fig. 2 in blue.

In Fig. 2 it can be seen that for $M = 10$, different terminal sets are covering different areas of the state–space which actually corresponds to further improvements in the domain of attraction. Using a single terminal set and linear local control law, which corresponds to the standard method as implemented in (Kwon and Han, 2005), yields a much smaller terminal set, as it can be seen in Fig. 2.

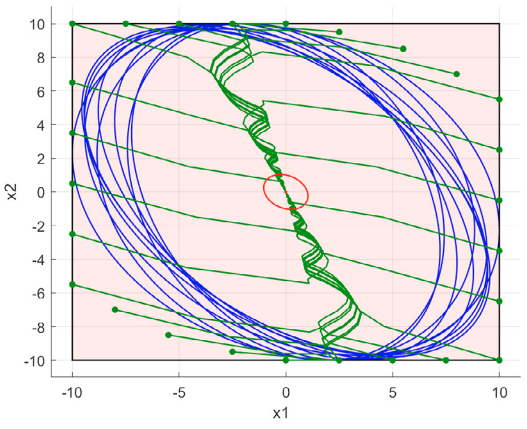

Fig. 2. Terminal set for $\kappa_j = 0.05$ (Kwon and Han, 2005) in red and sNMPC terminal sets (quasi–second order approximation, nonlinear control law, $M = 10$) in blue. State trajectories for some initial conditions on the boundary of the domain of attraction for $N = 1$ employing the sNMPC terminal costs and sets.

Table 2. Volume of the largest set computed using (Kwon and Han, 2005) versus sNMPC.

System	(Kwon and Han, 2005)	sNMPC
Inv. pend.	3.279275893066	305.7465536528
Oscillator	16.31201535119	20.78049118918
Cart	4.88197679500	15.74898441913
Flex. joint	0.04039957065	0.08499855865
2DOF arm	537.7987359438189	733.1966909073088

However, it should be noted that increasing M does not always result in larger sets. Similarly, employing a quasi–second order approximation is not always superior to a first order approximation since it introduces conservatism in a different way. Using a nonlinear control law results in terminal sets that are always large than or equal to the ones obtained using a linear control law. A linear control law is still provided as an option so that the application of the toolbox is not limited to input affine systems.

The sNMPC toolbox was also tested on a 2D oscillator (Darup and Cannon, 2015), a 2D cart spring–damper (Raimondo et al., 2009), a 4D flexible joint (Ghahramani and Towhidkhah, 2009) and a 4D 2DOF robotic arm (Cisneros et al., 2018). The volumes of the largest terminal sets calculated using the sNMPC toolbox for these examples are presented in Table 2 along with the volumes for terminal sets corresponding to the method in (Kwon and Han, 2005). The results show that by making use of the several available degrees of freedom, the sNMPC toolbox consistently results in enlarged terminal sets for various nonlinear dynamics.

Remark 3. The volumes presented in Table 1 of (Lazar and Tetteroo, 2018) were computed by $\sqrt{\det(P^{-1})}$ which are proportional but not equal to the actual volume of the ellipsoidal sets. In the sNMPC toolbox, the exact volumes are calculated using the Ellipsoidal Toolbox (Kurzhanskiy and Varaiya, 2006).

5. CONCLUSIONS

This paper presented the sNMPC Matlab toolbox for the computation of stabilizing terminal ingredients, which mainly builds on the framework presented in (Lazar and Tetteroo, 2018) and includes other existing methods as specific cases. The toolbox provides several degrees of freedom in the design of terminal ingredients such that significant improvements in the domain of attraction can be obtained for the resulting NMPC algorithms. The toolbox also includes functions for simulating the closed–loop NMPC systems and visualizing the terminal sets with aid of other existing Matlab toolboxes.

The toolbox can be freely downloaded at the link: `https://github.com/mlazar04/sNMPC`.

REFERENCES

Andersson, J., Gillis, J., Horn, G., Rawlings, J., and Diehl, M. (2018). Casadi — a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.

ApS, M. (2019). *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*

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

Chen, H., O'Reilly, J., and Ballance, D.J. (2003). On the terminal region of model predictive control for nonlinear systems with input/state constraints. *International Journal of Adaptive Control and Signal Processing*, 17(3), 195–207.

Chen, Y., Bruschetta, M., Picotti, E., and Beghi, A. (2019). Matmpc - a matlab based toolbox for real-time nonlinear model predictive control. In *2019 18th European Control Conference (ECC)*, 3365–3370.

Cisneros, P.S.G., Sridharan, A., and Werner, H. (2018). Constrained predictive control of a robotic manipulator using quasi-lpv representations. *IFAC-PapersOnLine*, 51, 118–123.

Currie, J. and Wilson, D. (2012). Opti: Lowering the barrier between open source optimizers and the industrial matlab user.

Darup, M.S. and Cannon, M. (2015). A missing link between nonlinear MPC schemes with guaranteed stability. In *2015 54th IEEE Conference on Decision and Control (CDC)*.

Dezert, J. and Musso, C. (2001). An efficient method for generating points uniformly distributed in hyperellipsoids.

Fitri, I.R. and Kim, J.S. (2020). A nonlinear model predictive control with enlarged region of attraction via the union of invariant sets. *Mathematics*, 8(11).

Ghahramani, N.O. and Towhidkhah, F. (2009). Constrained incremental predictive controller design for a flexible joint robot. *ISA Transactions*, 48(3), 321–326.

Herceg, M., Kvasnica, M., Jones, C., and Morari, M. (2013). Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*. Zürich, Switzerland.

Houska, B., Ferreau, H., and Diehl, M. (2011). ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3), 298–312.

Hu, X.B. and Chen, W.H. (2007). Model predictive control for non-linear missiles. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 221(8), 1077–1089.

Iles, S., Lazar, M., and Matusko, J. (2015). Stabilizing model predictive control based on flexible set-membership constraints. In *2015 European Control Conference (ECC)*.

Kurzhanskiy, A.A. and Varaiya, P. (2006). Ellipsoidal toolbox (et). In *Proceedings of the 45th IEEE Conference on Decision and Control*, 1498–1503.

Kwon, W.H. and Han, S.H. (2005). *Receding Horizon Control: Model Predictive Control for State Models.* Springer Verlag.

Lazar, M. and Spinu, V. (2015). Finite-step terminal ingredients for stabilizing model predictive control. *IFAC-PapersOnLine*, 48(23), 9–15.

Lazar, M. and Tetteroo, M. (2018). Computation of terminal costs and sets for discrete–time nonlinear mpc. *IFAC-PapersOnLine*, 51(20), 141–146.

Löfberg, J. (2004). Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*. Taipei, Taiwan.

Lucia, S., Rumschinski, P., Krener, A.J., and Findeisen, R. (2015). Improved design of nonlinear model predictive controllers. *IFAC-PapersOnLine*, 48(23), 254–259.

Michalska, H. and Mayne, D. (1993). Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 38(11), 1623–1633.

Raimondo, D.M., Limon, D., Lazar, M., Magni, L., and Camacho, E.F. (2009). Min-max model predictive control of nonlinear systems: A unifying overview on stability. *European Journal of Control*, 15(1), 5–21.

Rajhans, C., Griffith, D.W., Patwardhan, S.C., Biegler, L.T., and Pillai, H.K. (2019). Terminal region characterization and stability analysis of discrete time quasi-infinite horizon nonlinear model predictive control. *Journal of Process Control*, 83, 30–52.

Rajhans, C., Patwardhan, S.C., and Pillai, H. (2016). Two alternate approaches for characterization of the terminal region for continuous time quasi-infinite horizon nmpc. In *2016 12th IEEE International Conference on Control and Automation (ICCA)*, 98–103.

Wächter, A. and Biegler, L. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106, 25–57.

Yu, S., Hou, C., Qu, T., and Chen, H. (2015). A revisit to MPC of discrete-time nonlinear systems. In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*.