sNMPC Toolbox User Manual

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

The main goal of the sNMPC toolbox is to enable computation of stabilizing terminal ingredients for NMPC. Given a system model and set of constraints the main functionality of this tool is to deliver a set of terminal sets and terminal cost functions which guarantee the stability and recursive feasibility of the NMPC algorithm while providing several design choices. To complement the main functionality, the tool also contains additional functions for plotting the obtained terminal sets and simulating the controlled system employing the obtained terminal ingredients.

2 Installation

- Download and unzip sNMPC files. Add the corresponding directory to your Matlab path.
- Install YALMIP via https://yalmip.github.io/tutorial/installation/ (used within the `solve_LMIs()` and `solve_NLP_bisection()` to define the optimization problems.)
- Install MPT3 via https://www.mpt3.org/Main/Installation (used within the `solve_LMIs()` function to define the polytopic admissible sets.)
- Install MOSEK via https://www.mosek.com/downloads/ (used as a solver in the `solve_LMIs()` function to solve LMIs.)
- Install OPTI Toolbox via https://github.com/jonathancurrie/OPTI (used within the `solve_NLP_bisection()`, enables YALMIP to use IPOPT as a solver to solve the nonlinear optimization problems.)
- Install Ellipsoidal Toolbox via https://github.com/SystemAnalysisDpt-CMC-MSU/ellipsoids/wiki/Toolbox-2.1-installation-for-end-users (used within the `solve_LMIs()` and `solve_NLP_bisection()` to define and calculate volume of ellipsoidal sets. Also used in the "Plotting" module to plot ellipsoidal sets.)
- Install Casadi via https://web.casadi.org/get/ (used within the "simulation" module to define and solve nonlinear optimization problems within the MPC framework.)

3 Quick start - Typical flow

The best way to familiarize with the toolbox is to start from one of the several examples provided. One should first define their own system of interest following the structure in the examples. Then, again by following the structure in the examples, design choices should be set and toolbox functions should be employed in their typical order.

- When the system of interest is an input affine system it is recommended to always employ a nonlinear control law since it always outperforms or at worst recovers a linear one. For general nonlinear systems one has to employ a linear control law.
- The effects of other tuning parameters are observed to be highly dependent on the particular system dynamics thus, it is best to optimize those by trial and error.

3.1 Computation and plotting of terminal ingredients

- Define `sys` as a structure array containing system dynamics and constraints.
- Define `p` as a structure array containing user defined parameters such as cost matrices, number of terminal ingredients, κ_j, ρ etc.
- Define your choice of LMI solver (MOSEK), nonlinear optimization problem solver (IPOPT), and MPT3 YSet solver (MOSEK).

- Enter the choice of your "mode" as an integer from 1 to 4 where:
  - 1: first–order approximation linear control law,
  - 2: quasi second–order approximation linear control law,
  - 3: first–order approximation nonlinear control law,
  - 4: quasi second–order approximation nonlinear control law.
- Call `get_ABHessian()` to obtain the Jacobian and Hessian matrices.
- Call `solve_LMIs()` to solve the LMIs and obtain resulting terminal ingredients.
  - Returns `P, K, alpha, E1, VOL1, XUset, Xset_scaled`.
  - `K`: Set of terminal control laws.
  - `E1`: Set of ellipsoidal terminal sets.
  - `P`: Set of P_i matrices for $i = 0, 1, \ldots, M$.
  - Terminal sets are defined as `E1{1,i}=ellipsoid(inv((P{1,i})))`.
  - Terminal cost functions can be defined as `F1{1,i}=(1/alpha)*x'*P{1,i}*x`.
- Call `solve_NLP_bisection)` to solve nonlinear optimization problem on the approximation error and scale the terminal ingredients via bisection.
  - Returns `alphascale, E2, VOL2`.
  - `K`: Set of terminal control laws.
  - `E2`: Set of ellipsoidal terminal sets.
  - `P`: Set of P_i matrices remain unchanged.
  - Terminal sets are defined as `E2{1,i}=ellipsoid(inv((P{1,i}/alphascale)))`.
  - Terminal cost functions remain unchanged as `F2{1,i}=(1/alpha)*x'*P{1,i}*x`.
- Call `plot_ellipsoidal_sets()` to plot the terminal sets obtained after calling `solve_LMIs()` or `solve_NLP_bisection()`.

After obtaining the terminal ingredients, one may simulate the system using those as explained in the next subsection.

3.2 Simulating the systems employing the obtained terminal ingredients

- Redefine the system dynamics and constraints using '`SX`' data type of Casadi.
- First define a grided space, then call `plot_DOA()` to obtain a representation of the domain of attraction.
- For an initial condition, call `find_init_set()` to determine a feasible set to be used as an initial terminal set at $k = 0$ when time-varying sets are employed.
- Call `casadi_simulation(()` for an initial condition and simulation time of choice to simulate the system employing the terminal ingredients obtained by the toolbox.