DO-MPC

A toolbox for the easy, efficient and modular implementation of NMPC

User's Guide

Authors:
Sergio Lucia
Alexandru Tatulea-Codrean
Sebastian Engell

DO-MPC

Last updated: May 16, 2015

Contents

Chapter 1

Introduction

1.1 General Description

1.2 Capabilities and Limitations

1.3 Citing DO-MPC

If you use DO-MPC for published work please cite it as:

S. Lucia, A. Tatulea-Codrean, C. Schoppmeyer, and S. Engell. Fast, Efficient and Sustainable Development of Robust NMPC Solutions. Submitted to *IEEE Transactions on Control Systems Technology*, 2015.

1.4 License

Chapter 2

Installation Instructions

DO-MPC is a set of Python modules and scripts that is based on third party software. DO-MPC is strongly based on the use of CasADi (`http://www.casadi.org`). The easiest way to install all the third party software that is necessary for DO-MPC to work is to install one of the precompiled versions of CasADi, which are available for Windows and for Linux, from (`https://github.com/casadi/casadi/wiki/InstallationInstructions`). Depending on your operating system you need some additional software (mainly a Python distribution). For detailed description follow carefully the installation instructions for CasADi.

Once CasADi with IPOPT is running, the examples for DO-MPC should also run. It is highly recommended that you use a better linear solver than the one that comes with the precompiled version of IPOPT (mumps). You can do this by downloading the precompiled libraries from `http://www.hsl.rl.ac.uk/ipopt/`, which are available for Windows and for Linux. Once downloaded, add the path of the directory where the library has been store to your environment variable `PATH` on Windows or to `LD_LIBRARY_PATH` on Linux. After restarting your Python session the linear solver MA27 should be available.

Chapter 3

Your Efficient NMPC in 60 Minutes

The spirit of DO-MPC is that any user with just a moderate knowledge of model predictive control and of a scripting language like Python (or Matlab), should be able to get results of an efficient implementation of robust (or standard) nonlinear model predictive control in less than 60 minutes, just by looking at the examples and modifying the code accordingly in a very intuitive way. To achieve this, we provide templates for the problem specific information (model equations, constraints, cost function, definition of the simulator, observer) so that the user does not have to get into the tedious task of coding the discretization and formulation of the NLP, getting derivatives (thanks to CasaADi) or even plotting the results.

Although everything is programmed using Python, the code runs in C/C++ and therefore an excellent computational performance is achieved, but with a fast and easy development phase. To get your efficient implementation of a robust NMPC in 60 minutes follow the steps enumerated below:

1. Write the model equations in the file `template_model.py`
2. Describe the Optimal Control Problem (OCP) in the file `template_model.py`. The OCP description contains:
   (a) Initial Condition
   (b) Cost function
   (c) Constraints
3. Describe the controller parameters in the file `template_optimizer.py`
4. Describe the simulator parameters in the file `template_simulator.py`
5. Describe the observer in the file `template_observer.py`
6. Choose the plotting of the results by modifying `template_model.py`
7. Run DO-MPC using `python do-mpc.py`

Chapter 4

Structure of DO-MPC

Chapter 5

A Simple Example