NTNU
Norwegian University of Science and Technology

Masters Specialization Project

Sensitivity-Based Economic NMPC with a Path-Following Approach in Python

Brittany Hall

supervised by
Johannes Jäschke and Eka Suwartadi

September 12, 2017

Contents

Chapter 1

Summary

Chapter 2

Introduction

Model predictive control (MPC), along with non-linear model predictive control (NMPC), is an advanced control strategy that involves solving an optimization problem for a set horizon to determine the feedback value of the manipulated variables at each sampling interval. These two control strategies are traditionally used widely in the chemical industry for processes with large time constants (i.e., slow dynamics). Due to modern computation capabilities and algorithm development, this type of control has expanded to more types of systems (even fast dynamics). MPC has a growing interest in both research and industry due to its performance in a variety of processes in addition to its ability to handle constraints, perform optimization and consider economics and nonlinearities of the process [1]. The current areas of interest are: development algorithms for rapid optimization, development better modelling strategies, and new alternatives/variations that lead to improved closed-loop performance or reduce the computation time of the optimization problem.

Most industries care about the profitability of the process which is why economic MPC was developed. This allows for the integration of the economic optimization and the control layer into a single dynamic optimization layer [2]. Economic MPC works by adjusting the inputs such that the economic cost of the operation is directly minimized; thus allowing for the optimization of the cost during operation of the plant. However, nonlinear process models are often used for this style of optimization meaning that a drawback of economic MPC is the requirement of solving large nonlinear optimization problem (NLP) with the NMPC problem at every sample time. This computation can take a significant amount of time and lead to increasingly worse performance and even instability of the process [2].

One idea to reduce the effect of computational delay in NMPC is to use sensitvity-based methods which exploit the fact that the NMPC optimization problems are identical at each sample time with the exception of one changing parameter: the initial state. Therefore, the full nonlinear optimization problem is no longer solved. Instead, the sensitivity of the NLP solution at the previously-computed iteration is used to obtain an approximate solution to the new NMPC problem [2]. One such method is the advanced-step NMPC (asNMPC) which involves solving the full NLP at every sample time but this solution is computed in advance for a predicted initial state. When the new state measurement is available from the process, the NLP solution is corrected using a fast sensitivity update to make the solution match the measured state.

In this project, we focused on applying an improved path-following method for correcting the NLP solution within the advanced-step NMPC framework in Python. We illustrate how asNMPC with the predictor-corrector path-following algorithm performs in the presence of measurement noise and compare it with an ideal NMPC approach, where the NLP is assumed to be solved instantly.

Chapter 3

NMPC

3.1 NMPC Problem Formulations

3.1.1 The NMPC Problem

We consider a nonlinear discrete-time dynamic system expressed as [2]:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{3.1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ denotes the state variable, $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is the control input and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$ is a continuous model function, which calculates the next state $\boldsymbol{x}_{k+1}$ from the previous state $\boldsymbol{x}_k$ and control input $\boldsymbol{u}_k$, where $k \in \mathbb{N}$. This system will be optimized by a nolinear model predictive controller which solves the problem:

$$
\begin{aligned}
(\mathcal{P}_{NMPC}) : \quad & \min_{\boldsymbol{z}_l, \boldsymbol{v}_l} \quad \Psi(\boldsymbol{z}_N + \sum_{l=0}^{N-1} \psi(\boldsymbol{z}_l, \boldsymbol{v}_l) && \text{(3.2a)} \\
& \text{s.t.} \\
& \boldsymbol{z}_{l+1} = f(\boldsymbol{z}_l, \boldsymbol{v}_l), \qquad l = 0, \ldots, N-1, && \text{(3.2b)} \\
& \boldsymbol{z}_0 = \boldsymbol{x}_k, && \text{(3.2c)} \\
& (\boldsymbol{z}_l, \boldsymbol{v}_l) \in \mathcal{Z}, && \text{(3.2d)} \\
& \boldsymbol{z}_N \in \mathcal{X}_f && \text{(3.2e)}
\end{aligned}
$$

at each sample time. Here $\boldsymbol{z}_l \in \mathbb{R}^{n_x}$ is the predicted state variable; $\boldsymbol{v}_l \in \mathbb{R}^{n_u}$ is the predicted control input; and $\boldsymbol{z}_n \in \mathcal{X}_f$ is the final predicted state variable restricted to the terminal region $\mathcal{X}_f \in \mathbb{R}^{n_x}$. The stage cost is denoted by $\psi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}$ and the terminal cost by $\Psi : \mathcal{X}_f \rightarrow \mathbb{R}$. $\mathcal{Z}$ denotes the path constraints where $\mathcal{Z} = \{(\boldsymbol{z}, \boldsymbol{v}) | q(\boldsymbol{z}, \boldsymbol{v}) \leq 0\}$, where $q : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_q}$. The solution to this problem is denoted as $\{\boldsymbol{x}_0^*, \ldots, \boldsymbol{z}_N^*, \boldsymbol{v}_0^*, \ldots, \boldsymbol{v}_{N-1}^*\}$

The idea is that at sample time k, an estimate or measurement of the state $\boldsymbol{x}_k$ is obtained and the problem $\mathcal{P}_{NMPC}$ is solved, The first part of the optimal control sequence is then the plant input such that $\boldsymbol{u}_k = \boldsymbol{v}_0^*$. This part of the solution defines an implicit feedback law $\boldsymbol{u}_k = \kappa(\boldsymbol{x}_k)$, and the system evolves according to Equation 3.1. At the next sample time $k+1$, when the measurement of the new state is obtained, the procedure is repeated. Algorithm 1 summarizes the NMPC algorithm.

Algorithm 3.1: General NMPC algorithm.

```
1 set $k \leftarrow 0$
2 while *MPC is running* do
      1. Measure or estimate $x_k$
      2. Assign the initial state: set $\boldsymbol{z}_0 = x_k$
      3. Solve the optimization problem $\mathcal{P}_{NMPC}$ to find $\boldsymbol{v}_\mathbf{0}^*$.
      4. Assign the plant input $\boldsymbol{u}_k = \boldsymbol{v}_0^*$
      5. Inject $\boldsymbol{u}_k$ to the plant
      6. Set $k \leftarrow k+1$
```

3.1.2 Ideal NMPC and Advanced-Step NMPC Framework

To achieve optimal economic performance and good stability properties, the problem shown in $\mathcal{P}_{NMPC}$ needs to be solved instantaneously, allowing the optimal input to be injected into the process without time delay. This is known as ideal NMPC.

In reality, there will always be some time delay between obtaining the updated values of the states and injecting them into the plant. The main cause of this delay is the time required to solve the optimization problem $\mathcal{P}_{NMPC}$. As the process models grow, so to does the computation time. With sufficiently large systems, this computational delay cannot be neglected. One approach is the advanced-step NMPC (asNMPC) which is based on the following steps:

1. Solve the NMPC problem at time k with a predicted state value of $k+1$
2. When the measurement $\boldsymbol{x}_{k+1}$ becomes available at time $k+1$, compute an approximation of the NLP solution using fast sensitivity methods

3. Update $k \leftarrow k + 1$, and repeat from Step 1

Different fast sensitivity methods can be used and are discussed further in Section **??**.

3.2 Sensitivity-Based Path-Following NMPC

Below we outline sensitivity results and then utilize them in a path-following scheme for obtaining fast approximate solutions to the NLP.

3.2.1 Sensitivity Properties of NLP

The dynamic optimization problem can be written as a generic NLP problem:

$$
\begin{aligned}
(\mathcal{P}_{NLP}) : \quad & \min_{\mathcal{X}} \quad F(\mathcal{X}, \boldsymbol{p}) && \text{(3.3a)}\\
& \text{s.t.} \\
& c(\mathcal{X}, \boldsymbol{p}) = 0, && \text{(3.3b)}\\
& g(\mathcal{X}, \boldsymbol{p}) \leq 0 && \text{(3.3c)}
\end{aligned}
$$

where $\mathcal{X} \in \mathbb{R}^{n_{\mathcal{X}}}$ are the decision variables (typically the state variables and the control input) and $\boldsymbol{p} \in \mathbb{R}^{n_p}$ is the parameter (typically the initial state variable). $F : \mathbb{R}^{n_{\mathcal{X}}} \times \mathbb{R}^{n_p} \to \mathbb{R}$ is the scalar objective function, $c : \mathbb{R}^{n_{\mathcal{X}}} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_c}$ denotes the equality constraints, and $g : \mathbb{R}^{n_{\mathcal{X}}} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_g}$ denotes the inequality constraints. Each instance of the general parameteric NLP shown in Equation 3.3 that are solved for each sample time differ only in the parameter $\boldsymbol{p}$.

Chapter 4

Numerical Case Study

Chapter 5

Results

Chapter 6

Discussion

Chapter 7

Conclusion

Bibliography

[1] Federico Lozano Santamaria and Jorge Gomez. “Framework in PYOMO for the assessment and implementation of (as)NMPC controllers”. In: 92 (May 2016).

[2] Eka Suwartadi, Vyacheslav Kungurtsev, and Johannes Jäscke. “Sensitivity-Based Economic NMPC with a Path-Following Approach”. In: *Processes* 5 (2017), p. 8.