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

October 16, 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. 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 ([6]). 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. When an optimization-based controller such as MPC is used, the economic criterion can be included directly in the cost function of the controller ([3]). 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 ([6]).

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 ([6]). 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 [6]:

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

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 $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ 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:

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

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 $\psi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and the terminal cost by $\Psi : \mathcal{X}_f \to \mathbb{R}$. $\mathcal{Z}$ denotes the path constraints where $\mathcal{Z} = \{(\boldsymbol{z}, \boldsymbol{v}) | q(\boldsymbol{z}, \boldsymbol{v}) \leq 0\}$, where $q : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_q}$. The solution to this problem is denoted as $\{\boldsymbol{x}_0^*, \dots, \boldsymbol{z}_N^*, \boldsymbol{v}_0^*, \dots, \boldsymbol{v}_{N-1}^*\}$

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 3.1 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:

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

where $\boldsymbol{\chi} \in \mathbb{R}^{n_{\boldsymbol{\chi}}}$ 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). $F : \mathbb{R}^{n_{\boldsymbol{\chi}}} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}$ is the scalar objective function, $c : \mathbb{R}^{n_{\boldsymbol{\chi}}} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_c}$ denotes the equality constraints, and $g : \mathbb{R}^{n_{\boldsymbol{\chi}}} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_g}$ 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 $\boldsymbol{p}$. (See Suwardti et. al. for the Lagrangian and the Karush-Kuhn-Tucker (KKT) conditions [6].)

It has been shown that the perturbed NLP can be solved by solving a QP problem of the form [2]:

$$
\begin{aligned}
\min_{\Delta\boldsymbol{\chi}} \quad & \frac{1}{2}\boldsymbol{\chi}^T \nabla^2_{\boldsymbol{\chi}\boldsymbol{\chi}} \mathcal{L}(\boldsymbol{\chi}^*, \boldsymbol{p}_0, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\Delta\boldsymbol{\chi} + \Delta\boldsymbol{\chi}^T \nabla^2_{\boldsymbol{p}\boldsymbol{\chi}} \mathcal{L}(\boldsymbol{\chi}^*, \boldsymbol{p}_0, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\Delta\boldsymbol{p} && \text{(3.4a)} \\
\text{s.t.} \quad & \nabla_{\boldsymbol{\chi}} c_i(\boldsymbol{\chi}^*, \boldsymbol{p}_0)^T \Delta\boldsymbol{\chi} + \nabla_{\boldsymbol{p}} c_i(\boldsymbol{\chi}^*, \boldsymbol{p}_0)^T \Delta\boldsymbol{p} = 0 \qquad i = 1, \ldots, n_c, && \text{(3.4b)} \\
& \nabla_{\boldsymbol{\chi}} g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0)^T \Delta\boldsymbol{\chi} + \nabla_{\boldsymbol{p}} g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0)^T \Delta\boldsymbol{p} = 0 \qquad j \in K_+, && \text{(3.4c)} \\
& \nabla_{\boldsymbol{\chi}} c_i(\boldsymbol{\chi}^*, \boldsymbol{p}_0)^T \Delta\boldsymbol{\chi} + \nabla_{\boldsymbol{p}} c_i(\boldsymbol{\chi}^*, \boldsymbol{p}_0)^T \Delta\boldsymbol{p} \leq 0 \qquad j \in K_0 && \text{(3.4d)}
\end{aligned}
$$

where $K_+ = \{j \in \mathbb{Z} : \mu_j > 0\}$ is the strongly-active set and $K_0 = \{j \in \mathbb{Z} : \mu_j = 0 \text{ and } g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0) = 0\}$ denotes the weakly active set. Note that the solution to this QP is the directional derivative of the primal-dual solution of the NLP, it is a predictor step; thus we refer to (3.4a) as a pure-predictor.

It is important to recognize that the QP is only able to produce the optimal solution accurately for small perturbations and cannot be guaranteed to work for larger perturbations. One way of handling cases like this to divide the perturbation into several smaller intervals and to iteratively use the sensitivity to track the path of optimal solutions. This is known as a path-following method.

3.2.2 Path-Following Based on Sensitivity Properties

The basic idea of a path-following method is to reach the solution of the problem at a final parameter value $\boldsymbol{p}_f$ by tracing a sequence of solutions for a series of parameter values given by $\boldsymbol{p}(t_k) = (1 - t_k)\boldsymbol{p}_0 + t_k\boldsymbol{p}_f$ where $0 = t_0 < t_1 < \ldots < t_k < \ldots < t_N = 1$. The new direction is found by evaluating the sensitivity at the current point. It is common to include a corrector element into the QP that helps improve the ability of this method to find the correct solution.

We approximate 3.3 by a QP, linearized with respect to both $\boldsymbol{\chi}$ and $\boldsymbol{p}$, and enforce the equality of the strongly-active constraints.

$$\begin{aligned}
\min_{\Delta\boldsymbol{\chi}\Delta\boldsymbol{p}} \quad & \frac{1}{2}\Delta\boldsymbol{\chi}^T\nabla^2_{\boldsymbol{\chi}\boldsymbol{\chi}}\mathcal{L}(\boldsymbol{\chi}^*, \boldsymbol{p}_0, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)^T\Delta\boldsymbol{\chi} + \Delta\boldsymbol{\chi}^T\nabla^2_{\boldsymbol{p}\boldsymbol{\chi}}\mathcal{L}(\boldsymbol{\chi}^*, \boldsymbol{p}_0, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\Delta\boldsymbol{p} + \nabla_{\boldsymbol{p}}F^T\Delta\boldsymbol{\chi} + \nabla_{\boldsymbol{p}}F\Delta \\
\text{s.t.} \quad & c_i(\boldsymbol{\chi}^*, \boldsymbol{p}_0 - \Delta\boldsymbol{p}) + \nabla_{\boldsymbol{\chi}}c_i(\boldsymbol{\chi}^*, \boldsymbol{p}_0 + \Delta\boldsymbol{p})^T\Delta\boldsymbol{\chi} = 0 \qquad i = 1, ...n_c, \\
& g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0 - \Delta\boldsymbol{p}) + \nabla_{\boldsymbol{\chi}}g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0 + \Delta\boldsymbol{p})^T\Delta\boldsymbol{\chi} = 0 \qquad j \in K_+, \\
& g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0 - \Delta\boldsymbol{p}) + \nabla_{\boldsymbol{\chi}}g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0 + \Delta\boldsymbol{p})^T\Delta\boldsymbol{\chi} \leq 0 \qquad j \in \{1, \ldots, n_g/K_+\}
\end{aligned} \tag{3.5}$$

For the NMPC problem $\mathcal{P}_{NMPC}$, the parameter $\boldsymbol{p}$ corresponds to the current "initial" state ($\boldsymbol{x}_k$). The cost function is independent of $\boldsymbol{p}$ which means that $\nabla_{\boldsymbol{p}}F = 0$. In addition, the parameter is linear in the constraints so $\nabla_{\boldsymbol{p}}c$ and $\nabla_{\boldsymbol{p}}g$ are constants. Applying these simplifications we can write the above QP as:

$$\begin{aligned}
\min_{\Delta\boldsymbol{\chi}} \quad & \frac{1}{2}\Delta\boldsymbol{\chi}^T\nabla_{\boldsymbol{\chi}\boldsymbol{\chi}}\mathcal{L}(\boldsymbol{\chi}^*, \boldsymbol{p}_0 + \Delta\boldsymbol{p}, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\Delta\boldsymbol{\chi} + \nabla_{\boldsymbol{\chi}}F^T\Delta\boldsymbol{\chi} \\
\text{s.t.} \quad & c_i(\boldsymbol{\chi}^*, \boldsymbol{p}_0 - \Delta\boldsymbol{p}) + \nabla_{\boldsymbol{\chi}}c_i(\boldsymbol{\chi}^*, \boldsymbol{p}_0) = 0 \qquad i = 0, \ldots, n_c, \\
& g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0 - \Delta\boldsymbol{p}) + \nabla_{\boldsymbol{\chi}}g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0) = 0 \qquad j \in K_+, \\
& g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0 - \Delta\boldsymbol{p}) + \nabla_{\boldsymbol{\chi}}g_j(\boldsymbol{\chi}^*, \boldsymbol{p}_0) \leq 0 \qquad j \in \{1, \ldots, n_g\}/K_+
\end{aligned} \tag{3.6}$$

The QP formulation shown above is known as the predictor-corrector form. This QP tries to estimate how the NLP solution changes as the parameter does in the predictor component and refines the estimate, as the corrector, so that the KKT conditions are more closely satisfied at the new parameter.

The predictor-corrector QP is well suited for use in a path-following algorithm. Recall that we use the parameter equation: $\boldsymbol{p}(t_k) = (1-t_k)\boldsymbol{p}_0 + t_k\boldsymbol{p}_f$. At each point $\boldsymbol{p}(t_k)$, the QP is solved and the primal-dual solutions are updated as:

$$\boldsymbol{\chi}(t_{k+1}) = \boldsymbol{\chi} + \Delta\boldsymbol{\chi} \tag{3.7}$$

$$\boldsymbol{\lambda}(t_{k+1}) = \Delta\boldsymbol{\lambda} \tag{3.8}$$

$$\boldsymbol{\mu}(t_{k+1}) = \Delta\boldsymbol{\mu} \tag{3.9}$$

where $\boldsymbol{\chi}$ is obtained from the primal solution of the QP (3.6) and where $\Delta\boldsymbol{\lambda}$ and $\Delta\boldsymbol{\mu}$ correspond to the Lagrange multipliers of the QP.

The QP can detect changes in the active set along the path. If a constraint becomes inactive, the corresponding mulitiplier $\boldsymbol{\mu}_j$ will first become weakly active meaning that it is added to the set K_0. If a new constraint becomes active, the corresponding linearized inequality constraint in the QP will be active and tracked at the next iteration.

The path-following algorithm is summarized with its main steps in Algorithm 3.2. This algorithm is used to find a fast approximation of the optimal NLP solution corresponding to the new available state measurement; this is done by following the optimal solution path from the predicted state to the measured state.

Algorithm 3.2: Path-following algorithm

```
   Input: initial variables from NLP $\boldsymbol{\chi}^*(\boldsymbol{p}_0), \boldsymbol{\lambda}^*(\boldsymbol{p}_0), \boldsymbol{\mu}^*(\boldsymbol{p}_0)$
 1 fix stepsize $\Delta t$, and set $N = \frac{1}{\Delta t}$;
 2 set initial parameter value $\boldsymbol{p}_0$,;
 3 set initial parameter value $\boldsymbol{p}_f$,;
 4 set $t = 0$;
 5 for $k \leftarrow 1$ to $N$ do
 6     Compute step $\Delta\boldsymbol{p} = \boldsymbol{p}_k - \boldsymbol{p}_{k-1}$;
 7     Solve QP problem;
 8     if QP is feasible then
 9         $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi} + \Delta\boldsymbol{\chi}$;
10         Update dual variables appropriately using either the
             pure-predictor method or the predictor-corrector method;
11         $t \leftarrow t + \Delta t$;
12         $k \leftarrow k + 1$;
13     else
14         $\Delta t \leftarrow \alpha_1 \Delta t$;
15         $t \leftarrow t - \alpha_1 \Delta t$;
```

3.2.3 Path-Following asNMPC

As previously mentioned, for asNMPC, at every time step the full NLP is solved for a predicted state and when a new measurement is available, the precomputed NLP solution is updated by tracking the optimal solution curve from the predicted initial state to the new measured state. We highlight the fact that the solution of the last QP along the path corresponds to the updated NLP solution and only the inputs from the last QP become inputs to the plant. One unique quality of this method is that strong and weakly active inequality constraints are differentiated between. Strongly-active inequalities are linearized and included as equality constraints in the QP, but weakly active constraints are linearized and included as inequality constraints in the QP. This helps to ensure that the true solution path is tracked more accurately, particularly in the case that the full Hessian of the optimization problem is non-convex [6].

Chapter 4

Simple Example

A simple two-dimensional nonlinear problem was used to prove the validity of the path-following algorithm.

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & p_1 x_1^3 + x_2^2 \\ \text{s.t.} \quad & x_2 - e^{-x_1} \geq 0, \\ & x_1 \geq p_2 \end{aligned} \tag{4.1}$$

We start at the approximate solution to Equation 4.2$(x_0, y_0) = ((0.5, 0.6), 1.2)$ with $p = (1, -4)$ and trace a path to generate an approximate solution for $p = (8, 1)$. Note we will refer to $p = (1, -4)$ as p_0 or p_{init} and $p = (8, 1)$ as p_f or p_{final}.

Figure 4.1 shows the contour plots and constraints for the approximate solution at $p = (1, -4)$ and at $p = (8, 1)$ respectively.

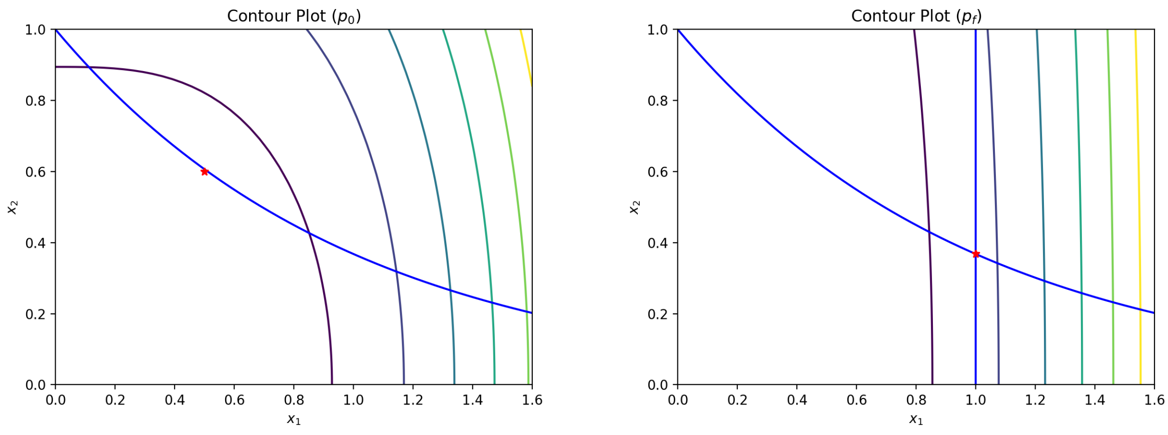

Figure 4.1: Plot of the problem at $t = 0$ and $t = 1$

The first thing to do before being able to apply any algorithm is to rewrite the problem in the more standard NLP form:

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & p_1 x_1^3 + x_2^2 \\ \text{s.t.} \quad & e^{-x_1} - x_2 \leq 0, \\ & p_2 - x_1 \leq 0 \end{aligned} \tag{4.2}$$

We see that this problem has two inequality constraints ($n_{iq} = 2$) and zero equality constraints ($n_{eq} = 0$). Algorithm 3.2 is then applied to this problem which requires that the NLP to be solved to find the initial variables: $\boldsymbol{\chi}(\boldsymbol{p}_0), \boldsymbol{\lambda}^*(\boldsymbol{p}_0), \boldsymbol{\mu}^*(\boldsymbol{p}_0)$. The NLP solution is then fed to a QP solver where the linearized NLP is solved as a QP. If the QP is feasible then the primal variables $\boldsymbol{\chi}$ are updated and the dual variables are updated either using the predictor-corrector method. The step size is then updated as well. If the QP is infeasible, then we will reduce the step size and try to solve the QP again.

Applying Algorithm 3.2 to this simple problem results in k iterations. Figure 4.2 illustrates x_1 with respect to t. The figure shows how x_1 changes steeply as the bound constraints become active.

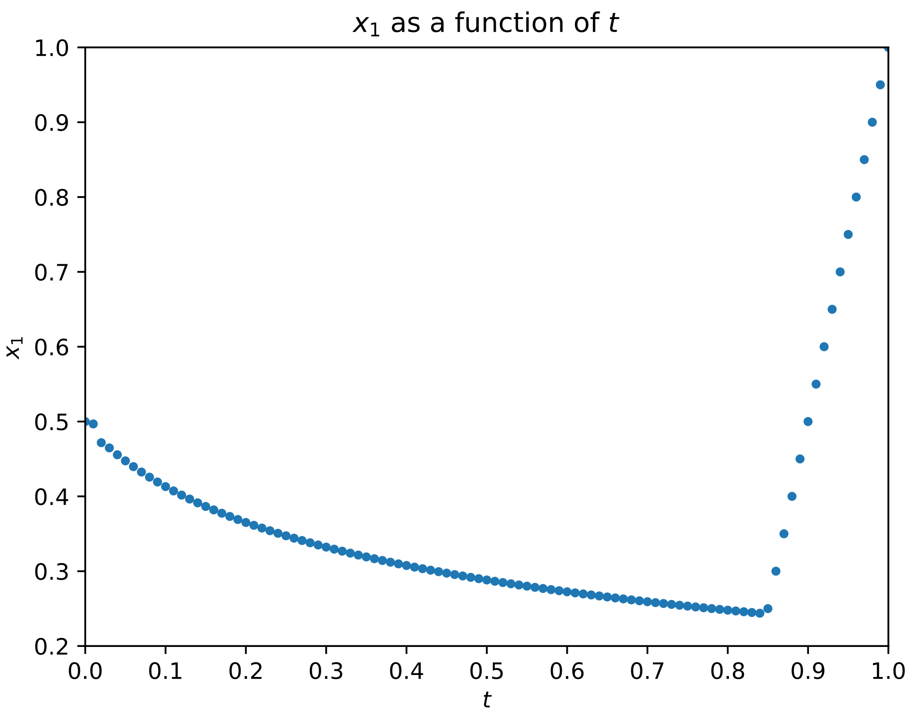

Figure 4.2: Plot of x_1 as a function of t

While this is a simple problem, it is a good test for Algorithm 3.2 since the problem changes substantially both in the nature of the active constraints and the slope of the objective function from p_0 to p_f[4].

Chapter 5

Numerical Case Study

5.1 Process Description

We now apply the path-following NMPC on a more relevant example, an isothermal reactor and separator process shown in Figure 5.1.

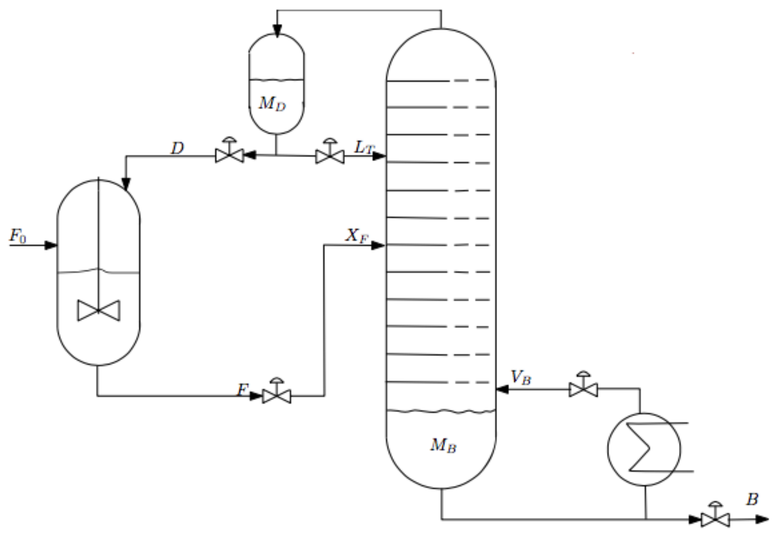

Figure 5.1: Diagram of a continuously-stirred tank reactor (CSTR) and distillation column.

The continuously-stirred tank reactor (CSTR) recieves a stream of pure component $\mathcal{A}$ and a recycle stream $\mathcal{R}$ from the distillation column. A first-order reaction ($\mathcal{A} \longrightarrow \mathcal{B}$) takes place in the CSTR and $\mathcal{B}$ is the desired product. The product is then feed with a flow rate F to the distillation column where the unreacted raw material is then separated from the product and recycled to the reactor. The desired product $\mathcal{B}$ is the bottom product and must have a certain purity. Table 5.1 summarizes the reaction kinetic parameters for the reactor.

Table 5.1: Reaction kinetic parameters

Reaction	**Reaction Rate Constant** (min^{-1})	**Activation Energy** ($\mathrm{J\,mol}^{-1}$)
$\mathcal{A} \longrightarrow \mathcal{B}$	1×10^8	6×10^4

The distillation column model is taken from [5]. The parameters used for the distillation column are summarized in Table 5.2.

Table 5.2: Distillation column parameters

Parameter	**Value**
α_{AB}	1.5
number of stages	41
feed stage location	21

The distillation column is comprised of 40 theoretical stages (39 trays and a reboiler) plus a total condenser. The feed is an equimolar liquid mixture of components $\mathcal{A}$ and $\mathcal{B}$ with a relative volatility of 1.5. The pressure P is assumed constant due to perfect control of P using V_T as an input. The reflux and boilup rates are such that we nominally have 99% purity for reach product (y_D and x_B). The nominal holdup is $M_i^*/F = 0.5$ min for all stages, including the reboiler and condenser. A simple linear relationship $L_i(t) = L_i^* + (M_i(t) - M_i^*)/\tau_L$, where $\tau_L = 0.063$ min is used to model the liquid flow dynamics on all trays. The model uses the following assumptions: binary separation, constant relative volatility, no vapor holdup, one feed and two products, constant molar flows, and a total condenser. Actuator and measurement dynamics are not included.

The whole model (CSTR and distillation column) has a total of 84 state variables: 82 from the distillation column (mole fractions and liquid holdups from each stage) and two from the CSTR (concentration and holdup).

5.1.1 Model Equations

The basic equations used to model the CSTR and distillation column are outlined below. The notation is outlined in Table ??.

i) Total balance on stage i:

$$\frac{dM_i}{dt} = L_{i+1} - L_i + V_{i+1} - V_i \tag{5.1}$$

ii) Material balance for light component on each stage i:

$$\frac{d(M_i x_i)}{dt} = L_{i+1}x_{i+1} + V_{i-1}y_{i-1} - L_i x_i - V_i y_i \tag{5.2}$$

which also gives the following expression for the derivative of the liquid mole fraction:

$$\frac{dx_i}{dt} = \frac{\frac{d(M_i x_i)}{dt} - x_i \frac{dM_i}{dt}}{M_i} \tag{5.3}$$

iii) Algebraic equations(apply to all stages except condenser, feed and reboiler):

- Vapor-liquid equilibrium

$$y_i = \frac{\alpha x_i}{1 + (\alpha - 1)x_i} \tag{5.4}$$

- From assumption of constant molar flows and no vapor dynamics (except if feed is partially vaporized):

$$V_i = V_{i-1} \tag{5.5}$$

- Linearized liquid flow:

$$L_i = L0_i + \frac{(M_i - M0)}{\tau_l} + (V - V_0)_{i-1} \tag{5.6}$$

where $L0_i$ kmol min^{-1} and $M0_i$ kmol are the nominal values for the liquid flow and holdup on stage i.

iv) Feed stage ($i = NF$):

$$\frac{dM_i}{dt} = L_{i+1} - L_i + V_{i-1} - V_i + F \tag{5.7}$$

$$\frac{d(M_i x_i)}{dt} = L_{i+1}x_{i+1} + V_{i-1}y_{i-1} - L_i x_i - V - iy_i + Fz_F \tag{5.8}$$

v) Total condenser ($i = NT$):

$$\frac{dM_i}{dt} = V_{i-1} - L_i - D \tag{5.9}$$

$$\frac{d(M_i x_i)}{dt} = V_{i-1} y_{i-1} - L_i x_i - D x_i \tag{5.10}$$

vi) Reboiler ($i = 1$):

$$M_i = M_B \tag{5.11}$$

$$V_i = V_B = V \tag{5.12}$$

$$\frac{dM_i}{dt} = L_{i+1} - V_i - B \tag{5.13}$$

$$\frac{d(M_i x_i)}{dt} = L_{i+1} x_{i+1} - V_i y_i - B x_i \tag{5.14}$$

5.1.2 Column data

As mentioned above, the column has 41 stages including the reboiler and total condenser with the feed stage located at stage 21. The nominal steady state conditions for this column are summarized in Table **??**.

Table 5.3: Column data

Parameter	**Value**
Feed rate F	1 [kmol min^{-1}]
Feed composition z_F	0.5 [mole fraction unit]
Feed liquid fraction q_F	1 [saturated liquid]
Reflux flow L_T	2.706 [kmol min^{-1}]
Boilup V	3.206 [kmol min^{-1}]
Liquid holdup $M0$	0.5 [kmol]
Time constant for liquid dynamics τ_l	0.063 [min]
λ	0
Distillate D	0.5 [kmol min^{-1}]
Distillate composition $y_D = x_{NT}$	0.99 [mole fraction units]
Bottoms B	0.5 [kmol min^{-1}]
Bottoms composition $x_B = x_1$	0.01 [mole fraction units]

This steady state data can easily be recalculated to simulate different columns (number of stages, feed composition, flows, relative volatility, holdups) by using `col_model.py` and `col_LV.py`.

5.1.3 Detailed LV-model

This model is obtained from model reduction of the detailed 82 state model and only has 5 states. It includes liquid flow dynamics, composition dynamics, and disturbances. In this simplified case the inputs are the reflux (L) and boilup (V) and the controlled outputs are the top and bottom product compositions (y_D and x_B).

This simplified model is used to find initial steady state conditions for the distillation column to initialize the remainder of the simulation.

5.2 Objective Function and Constraints

The economic objective function to be optimized under operation is given by:

$$J = p_F F_0 + p_V V_B - p_B B \tag{5.15}$$

where p_F is the feed cost, p_V is the steam cost and p_B is the product price. The following prices are used in this case study: $p_F = 1\,\$/\text{kmol}, p_V = 0.02\,\$/\text{kmol}$, and $p_B = 2\,\$/\text{kmol}$. The constraints are the concentration of the bottom product ($x_B \leq 0.1$), the liquid holdup at the bottom and the top of the distillation column and in the CSTR ($0.3 \leq M_{(B,D,CSTR)} \leq 0.7\,\text{kmol}$). The control inputs are the reflux flow (L_T), boil-up flow (V_B), feed rate to the distillation column (F), distillate flow rate (D) and bottom product flow rate (B). These control inputs have the following bounds:

$$\begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix} \leq \begin{bmatrix} L_T \\ V_B \\ F \\ D \\ B \end{bmatrix} \leq \begin{bmatrix} 10 \\ 4.008 \\ 10 \\ 1.0 \\ 1.0 \end{bmatrix} \tag{5.16}$$

To solve this problem we must first run a steady-state optimization; we select a feed rate $F_0 = 0.3\,\text{kmol}\,\text{min}^{-1}$. This gives optimal values for control inputs and state variables at this feed rate. The optimal steady state input values are found to be $\boldsymbol{u}_{ss} = \begin{bmatrix} 1.18 & 1.92 & 1.03 & 0.74 & 0.29 \end{bmatrix}^T$.

The optimal state and control inputs are then used to construct a regularization term that is added to the objective function. Regularization terms are often used in optimization problems helps to introduce more information to the function to help solve an ill-posed problem or to prevent overfitting. The regularization term also helps to regulate the different goals of the objective function. The new objective function for the regularized stage is written as:

$$J_m = p_F F_0 + p_V V_B - p_B B - p_D D + (\boldsymbol{z} - \boldsymbol{x}_s)^T \boldsymbol{Q}_1 (\boldsymbol{z} - \boldsymbol{x}_s) + (\boldsymbol{v} - \boldsymbol{u}_s)^T \boldsymbol{Q}_2 (\boldsymbol{v} - \boldsymbol{u}_s) \tag{5.17}$$

The weights ($\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$) are selected to make the rotated stage cost of the steady state problem strongly convex. To find a valid diagonal regularization matrix Q we utilize the Gershgorin property for a matrix which states for a matrix $\boldsymbol{A} = (a_{ij})$:

$$a_{ii} - \sum_{i \neq j} |a_{ij}| \leq \mu_i \leq a_{ii} + \sum_{i \neq j} |a_{ij}| \tag{5.18}$$

where μ_i are the eigenvalues of $\boldsymbol{A}$[3]. This property can be utilized to systematically find the regularization terms such that the rotated stage cost will be strongly convex and thus a stable economic NMPC controller can be obtained using this method. For further details on this method see [3].

The distillation column is initialized using the steady states values for a feed rate of $F_0 = 0.29$ kmol min^{-1} meaning that the controller is essentially controlling for a throughput change from $F_0 = 0.29$ kmol min^{-1} to $F_0 = 0.30$ kmol min^{-1}. The simulation is run for 150 MPC iterations with a sample time of 1 minute. The prediction horizon of the NMPC controller is set to 30 minute. This results in an NLP with 10,314 optimization variables ([6]). To solve the NLP, CasADi with IPOPT is used; for the QPs, we use CasADi with qpoases ([1]).

Chapter 6

Results

6.1 Open-Loop Optimization

We now compare the solutions obtained from the path-following algorithm with the “true” solution of the optimization problem $\mathcal{P}_{NMPC}$ obtained by solving the full NLP.

Chapter 7

Discussion

Chapter 8

Conclusion

Bibliography

[1] Joel Andersson. “A General-Purpose Software Framework for Dynamic Optimization”. PhD thesis. Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10 3001-Heverlee Belgium: Arenberg Doctoral School- KU Leuven, 2013.

[2] J. Frèdèric Bonnans and Alexander Shapiro. “Optimization Problems with Perturbations: A Guided Tour”. In: *SIAM Rev* 40 (1998), pp. 228–264.

[3] Johannes Jäschke, Xue Yang, and Lorenz T. Biegler. “Fast Economic Model Predictive Control Based on NLP-Sensitivities”. In: *Journal of Process Control* (2014).

[4] Vyacheslav Kungurtsev and Moritz Diehl. “Sequential quadratic programming methods for parametric nonlinear optimization”. In: *Computational Optimization and Applications* 59 (2014), pp. 475–509.

[5] Sigurd Skogestad and Ian Postlethwaite. “Multivariable Feedback Control: Analysis and Design”. In: (2005).

[6] Eka Suwartadi, Vyacheslav Kungurtsev, and Johannes Jäscke. “Sensitivity-Based Economic NMPC with a Path-Following Approach”. In: *Processes* 5 (2017), pp. 8–35.