MODEL PREDICTIVE STATIC PROGRAMMING CONTROL APPLIED TO MINERAL PROCESSING PLANTS

by

Zander Meindert Noome

Submitted in partial fulfillment of the requirements for the degree
Master of Engineering (Electrical Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

May 2023

SUMMARY

MODEL PREDICTIVE STATIC PROGRAMMING CONTROL APPLIED TO MINERAL PROCESSING PLANTS

by

Zander Meindert Noome

Supervisor:	Prof. J.D. le Roux
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Master of Engineering (Electrical Engineering)
Keywords:	Flotation, grinding mills, mineral processing, model predictive control, model predictive static programming, optimal control, process control

In a mineral processing plant, the separation of valuable material from ore has multiple stages. Usually, the ore is crushed or ground into smaller parts through multiple crushers or grinding mills. This is called the communition process. This process is typically the first stage for extracting valuable material and is important for further down-stream processes. The output of the communintion stage is usually regulated to achieve a stable throughput and a specific ore particle size. After the ore is crushed and ground to a specified size, the valuable material in the ore needs to be separated from the undesired materials.

The properties of the desired material influence the method used for separation. These methods include froth flotation, gravitational separation, magnetic separation and electrostatic separation. The separation process can include multiple process streams to get a high grade of the desired minerals out of the ore. In froth flotation, the main objective is to extract the desired material from the ore to obtain a large mineral recovery. Because the flotation process relies on the flotation of particles, particle size is extremely important.

The use of control systems in mineral processing plants has been adopted to improve throughput, optimize power usage, ensure safe process operation and to running at a stable operating condition. The control of these plants makes use of different advanced process control strategies which include but are not limited to cascaded control, where multiple layers of control systems are applied, and model predictive control. These different control strategies can range from regulatory control to supervisory control. Because of the large number of inputs to these plants, efficient controllers are necessary to obtain desired results.

The use of Nonlinear Model Predictive Control (NMPC) is an attractive option for most mineral processing plants because of the constraint management capabilities of the controller. Unfortunately, the NMPC method has a large computational load which requires sufficient resources to make it a viable option. Another model predictive control method known as Model Predictive Static Programming (MPSP) has shown promise to improve the computational time of a standard NMPC controller. The MPSP control philosophy generates a static optimization problem which is less computationally difficult to solve compared to the dynamic optimization problem that is generated through NMPC.

In this dissertation, the control of a single-stage grinding mill circuit and a four-cell flotation circuit with an MPSP controller to reduce the computational load is proposed. The computational efficiency and the output performance of MPSP controllers are compared to NMPC controllers as a motivation for the use thereof. The comparison is done by simulating two mineral processing stages, namely the communition phase and the separation phase. The simulations considered different configurations for both the MPSP and NMPC controllers.

The comparison of the controllers in the simulations shows that the MPSP controller obtained similar or improved plant results while also having a reduced computational time compared to the NMPC controller. The MPSP controller also displays scalability improvements compared to the NMPC controllers which can be beneficial for supervisory control of large-scale processing plants.

ACKNOWLEDGMENT

I am very thankful for my supervisor Prof. Derik le Roux and all his support, guidance and willingness to share his expertise. I have learned a lot from him in my postgraduate studies and I am privileged to have been given the opportunity to study under his supervision.

I would also like to thank Prof. Radikant Padhi for all the insight that he has provided on the applied model predictive controller in this study.

I am thankful for SACAC and the IFAC Foundation for their financial support, sponsorships and making it possible for me to attend the MMM2022 conference.

I am truly grateful to the people that made it possible for me do my postgraduate studies, namely Denzel Jansen Van Vuuren and his wife Christelle.

Thank you to my supportive and loving family, friends and specifically my fiancé for all the words of encouragement. I am grateful for their continual investment in my life, and I can rightly say that I am the man I am today because of them.

Lastly, I would like to thank God for giving me the strength and perseverance to complete this milestone.

LIST OF ABBREVIATIONS

ADP	Approximate Dynamic Programming
LTV	Linear Time-Varying
MPC	Model Predictive Control
MPSP	Model Predictive Static Programming
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
QP	Quadratic Programming
RTI	Real-Time Iteration
SQP	Sequential Quadratic Programming

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Context of the problem

The use of process control in industrial plants is an efficient way to maintain consistent product quality, improve throughput, optimise power usage, and ensure safe process operation. Since most industrial processes are multi-variable and nonlinear, they can be difficult to control. Nonlinear Model Predictive Control (NMPC) is an attractive solution to control large systems with highly interactive and nonlinear input-output responses. The process industry uses MPC regularly because of its constraint management capabilities and control simplicity (Mayne, 2014; Cisneros et al., 2016). NMPC is ideal for processes with relatively slow dynamics since it can be computationally intensive to apply online (Bemporad et al., 2002; Schwenzer et al., 2021).

By way of example for the mineral processing industry, a robust NMPC was implemented in simulation on a grinding mill circuit by Coetzee et al. (2010), but the computational time was too long for practical implementation. To produce a practically viable controller, it is necessary to reduce the computational time of the MPC without compromising its performance.

NMPC formulations which can include multiple constraints and operating conditions can potentially increase the computational burden of the algorithm. The reason for the computational burden of these algorithms is the nonlinear nature of the models being used. The nonlinearities of a system add complexity in such a way that tailored algorithms for the NMPC implementation might be necessary to achieve optimal results within a given sampling time (Kouzoupis et al., 2015). Different fast NMPC schemes are available to improve the computational time of NMPC algorithms. Some of these fast NMPC schemes use suboptimal update methods or sensitivity-based update methods to approximate the optimal solution to the controller objective functions (Wolf and Marquardt, 2016). Other fast

NMPC schemes include a real-time iterations (RTI) algorithm to reduce the computational time of the NMPC algorithm (Diehl et al., 2005; Wolf and Marquardt, 2016; Gros et al., 2020). The RTI algorithm is implemented in such a way that all the preliminary calculations that can be done without the initial state estimate of a system are derived from the NMPC objective function. The preliminary calculations are then used together with the state estimates when they are obtained, either through sampling or through a state estimation algorithm, to get an optimal solution to the objective function (Gros et al., 2020).

Another fast NMPC scheme known as advanced-step NMPC uses the previously calculated control input of the plant to predict the future plant states and solves the respective objective function of the NMPC controller in advance (Zavala and Biegler, 2009; Wolf and Marquardt, 2016).

Explicit NMPC is another technique that is used to decrease the computational time of NMPC controllers (Pistikopoulos, 2009). The challenge with Explicit NMPC is that the offline calculations become very difficult to solve for high-dimensional complex processes (more than five state dimensions) (Wang and Boyd, 2010). Chen et al. (2018) and Zhang et al. (2021) used reinforcement learning to approximate the polyhedral regions of explicit MPC into a function for the explicit control law. These significantly improve the computational time of standard explicit MPC with fewer required computer resources.

Most of the online computational fast MPC methods work on the principle of either decreasing the number of decision variables of the optimization problem or altering the optimization routine to optimize the computational time of the specific plant model. Some of the ways to decrease the decision variables for the optimization problem is by including move-blocking, where the number of control moves to be calculated is reduced by keeping a set of control moves constant between time iterations (Cagienard et al., 2007). Faroni et al. (2017) show that the decision variable can be decreased by choosing the prediction time steps independently from the sampling time and the control horizon. Another method to increase optimization efficiency is warm-starting, where the calculated control input of the previous iteration is used as the initial control solution for the current iteration (Wright, 1996; Wang and Boyd, 2010).

The computational time of the MPC controllers can also be decreased by defining the termination constraints of the optimization algorithm (Wang and Boyd, 2010). This decreases the number of

function evaluations and can lead to suboptimal solutions. This fast MPC control technique was implemented on a gas turbine system by Hou et al. (2020), where the optimization routine complexity was reduced by using the original obstacle point method. A barrier parameter for the inequality constraints simplified the optimization problem and the warm-starting method was also implemented to further reduce computational time. The above-mentioned methods of decreasing the number of decision variables of the optimization routine and defining termination conditions can be used for general NMPC controllers as well.

Kunz et al. (2013) approximated a fast dynamics nonlinear plant model as a Linear Time-Varying (LTV) model for the MPC problem formulation with a flatness-based trajectory. The method obtains a discrete-time LTV model for the defined prediction period. The method was able to generate a control solution for a micro-coaxial helicopter at approximately 30 % of its sampling time.

Padhi and Kothari (2009) developed a different approach to NMPC known as Model Predictive Static Programming (MPSP). MPSP combines two different philosophies: NMPC and Approximate Dynamic Programming (ADP) (Kumar and Padhi, 2014). This method was implemented in simulation as a boost phase guidance scheme and showed a close correlation to the optimal control solution.

The MPSP algorithm was also implemented in simulation to an air-to-air missile guidance scheme (Bhitre and Padhi, 2014). The simulations handled inequality state constraints by using a slack variable approach which transforms the inequality constraints into an unconstrained problem.

Li et al. (2019) manipulated MPSP to include input constraints for a guidance law for air-to-ground missile cooperation attacks. The input constraints were implemented by inserting the constraints as a penalty function to the objective function. Kumar et al. (2019) further adapted the MPSP method to include state and input constraints. The constrained MPSP method was implemented in simulation and in real-time for the energy management of a Parallel Hybrid Electric Vehicle by (Biswas et al., 2022). They found that the MPSP algorithm has a faster computational time compared to the traditional MPC algorithms. Furthermore, there are various examples of MPSP being applied in the aerospace industry (Tripathi and Padhi, 2016; Zhang et al., 2016; Bin et al., 2018; Hong et al., 2019; Wang et al., 2019).

In terms of mineral processing, an unconstrained MPSP controller was applied in simulation to a

grinding mill circuit. Results showed that the unconstrained MPSP controller had a similar overall performance as an unconstrained NMPC controller when there were disturbances and measurement noise added to the plant. The MPSP method had a significantly shorter computational time than the NMPC method (Le Roux et al., 2014).

1.1.2 Research gap

There is still a big drive to discover or develop new control techniques that take less computational power. Although more powerful computer technology can handle larger control problems, there is still a need to decrease the computational time for the control algorithm itself.

Faster computational times mean that more complex and larger process models can be used in model-based controllers. A mineral processing plant contains large process units, long transport time delays, nonlinear process responses, and strict constraints. The models can become large, especially if the process is modelled from a plant-wide perspective for economic optimization purposes. The aim is to provide a solution so that these large models can be used as part of a model-based control strategy.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

1.2.1 Research objective

The proposed research is focused on comparing constrained MPSP to constrained NMPC algorithms in the context of mineral processing, specifically communition and flotation. The aim is to determine if a constrained MPSP controller takes less computational time compared to a constrained NMPC controller without compromising plant performance. Multiple simulations will need to be executed to obtain the necessary information.

1.2.2 Research questions

- *Computation time*: Is the computational cost of constrained MPSP better than that of constrained NMPC for different mineral processing plants?
- *Performance*: Can MPSP perform as well as NMPC in terms of controlling a mineral processing plant with specific objectives and constraints?
- *Trade-off*: What is the trade-off between the NMPC and the MPSP method in the application of a grinding mill circuit and a flotation circuit?

1.3 RESEARCH GOALS

- Simulate the constrained MPSP and constrained NMPC controllers on a grinding mill circuit.
- Simulate the constrained MPSP and constrained NMPC controllers on a flotation circuit.
- Obtain computational time information of the different controllers on the different mineral processing plant models.
- Conclude the trade-off between constrained MPSP and constrained NMPC for the simulated mineral processing circuits.

1.4 RESEARCH PROCEDURE

Specific information needs to be obtained to compare the constrained MPSP technique to the NMPC technique. Computational time is the time it takes for a computer to complete a set of computations. The computational time will be measured by the time it takes to calculate the following iterative control inputs of the plant. The difference between the techniques will then be quantifiable and consistent.

1.4.1 Obtain the plant model

A working model will need to be obtained to implement the two control techniques. The control techniques will be implemented on a grinding mill circuit model and a flotation circuit model in simulation (Le Roux et al., 2013; Oosthuizen et al., 2021). The model will have to include multiple state and control constraints to be able to use the two control methods.

1.4.2 Solve the objective functions

The constrained MPSP and NMPC techniques calculate the minimum of a chosen objective function with constraints. The minimisation problem of the constrained NMPC is known as a Nonlinear programming (NLP) problem whereas the constrained MPSP minimisation problem is a standard quadratic programming (QP) problem. The MPSP and the NMPC controller should use the same optimization method because it will make the comparison between the two control methods unbiased. The constrained MPSP technique usually has more iterative steps but a simpler cost function, whereas the constrained NMPC usually has a more complex cost function with fewer iteration steps.

1.4.3 Obtain performance data

Disturbance rejection and set-point tracking is a measure of the performance of a control technique. All these parameters can be compared between the NMPC method and the MPSP method, which will indicate if the MPSP method is as efficient as the non-linear implementation of the industry standard

MPC technique (Schwenzer et al., 2021). The constrained MPSP method has not been implemented in an industrial mineral process thus far, and the results obtained from the performance data can potentially motivate the use thereof.

After all the procedures are followed, whether the constrained MPSP method can reduce the computational time of a mineral processing plant, specifically a grinding mill plant and a flotation circuit, with similar performance to a constrained NMPC controller, can be answered.

1.5 RESEARCH CONTRIBUTION

The contribution of the research is extending the use of an unconstrained MPSP algorithm on a grinding mill circuit to a constrained MPSP algorithm. The work successfully simulates a constrained MPSP controller on a grinding mill circuit with activated constraints. The work is extended further by implementing a constrained MPSP controller simulation on a four-cell flotation circuit.

1.6 RESEARCH OUTPUTS

A conference publication resulting from this study is as follows:

- Noome, Z.M., Le Roux, J.D., 2022. Controlling a grinding mill circuit using constrained model predictive static programming. IFAC-PapersOnLine 55, pp. 49-54.

A journal article based on the study was submitted as follows:

- Noome, Z.M., Le Roux, J.D., Padhi, R., 2023, "Optimal Control of Mineral Processing Plants using Model Predictive Static Programming", submitted to J. Process Control.

1.7 OVERVIEW OF THE STUDY

In Chapter 2 a literature review is given on the control methods that were simulated, namely the NMPC and the MPSP method. In Chapter 3 the research methodology is discussed which shows the process followed for both the simulation of the grinding mill circuit and the flotation circuit. Chapter 4 is the description of the grinding mill circuit. The chapter also includes the simulation setup and results. In Chapter 5 the flotation circuit description, plant model, simulation setup and simulation results are given. In Chapter 6 concluding remarks are given between the results of the MPSP and the NMPC controllers from the obtained simulation results.

CHAPTER 2 LITERATURE STUDY

Section 2.1 gives an overview, the standard formulation and different approaches to reduce the computational time of the NMPC algorithm. Section 2.2 gives an overview of the MPSP method and how it has developed, then it discusses the derivation of the unconstrained and constrained MPSP algorithm as well as gives the summarized sequential steps for the algorithm.

2.1 NONLINEAR MODEL PREDICTIVE CONTROL

2.1.1 Overview of NMPC

The invention of linear MPC in the 1970s has been used in multiple applications from the aerospace industry to the chemical process industry (Allgower et al., 2004). Most processes or systems are nonlinear, but the use of linear MPC requires a linear plant model. In the industry, plant models are usually derived from first principles or derived empirically through experimental data. These plant models are also nonlinear and can be linearized for simplicity and computational efficiency in applications of MPC (Nise, 2020).

In mineral processing, there is a large incentive to increase the operating performance of a plant. These performance requirements can be improved by using the nonlinear models of the systems (Allgower et al., 2004). Thus, the use of nonlinear MPC has become a desirable control technique.

The use of NMPC compared to MPC comes at the cost of larger computational times. With the continual increase in computational power as well as the development of faster and more efficient algorithms like fast NMPC schemes, NMPC has become an attractive option for plant operation (Wolf and Marquardt, 2016). Most of the fast NMPC schemes either decrease the degrees of freedom of the objective function or uses a different/tailored control architecture to improve the computational time.

The formulation of the normal discrete NMPC algorithm is shown below with the inclusion of techniques to reduce the computational time (Allgower et al., 2004).

2.1.2 Mathematical formulation of NMPC

The constrained NMPC optimization problem can be formulated as

$$\min_{U_k, U_{k+1}, \ldots, U_{k+N_c}} J(X_k, U_k),$$

where

$$\begin{aligned} J(X_k, U_k) =& \frac{1}{2} \sum_{k=1}^{N_p} (Y_k - Y_k^*)^T Q_{nmpc} (Y_k - Y_k^*) \\ &+ \frac{1}{2} \sum_{k=1}^{N_c} (U_{k+1} - U_k)^T R_{nmpc} (U_{k+1} - U_k) \end{aligned} \tag{2.1}$$

s.t.

$$\begin{aligned} X_{k+1} &= F_k (X_k, U_k) \\ Y_k &= h_k (X_k, U_k) \\ U^{LB} &\leq U \leq U^{UB} \\ X^{LB} &\leq X \leq X^{UB} \\ Y^{LB} &\leq Y \leq Y^{UB}, \end{aligned} \tag{2.2}$$

where N_p is the prediction horizon, N_c is the control horizon, Y_k^* is the desired set-point, U_k is the inputs, X_k is the states of the plant, U^{LB} and U^{UB} are the lower and upper bounds of the input constraints respectively, X^{LB} and X^{UB} are the lower and upper bounds of the state constraints respectively, Y^{LB} and Y^{UB} are the lower and upper bounds of the output constraints respectively, and R and Q are the input deviation weighting matrices and output error weighting matrices respectively. The minimization problem in (2.1) can be solved using an appropriate numerical optimization routine (Nocedal and Wright, 2006). A graphical representation of the receding horizon of the NMPC algorithm is shown in Fig. 2.1.

2.1.3 Nonlinear Model Predictive Control Summary

1. Obtain an initial control input U_k.
2. Calculate the prediction state and output trajectory X_k and Y_k using the control input U_k.
3. Calculate the value of the objective function (2.1).
4. Change the input U_k and re-evaluate steps 2 and 3.
5. Continuously repeat step 4 to minimize the objective function (2.1) for a specific number of iterations, or until a user-defined tolerance is met.

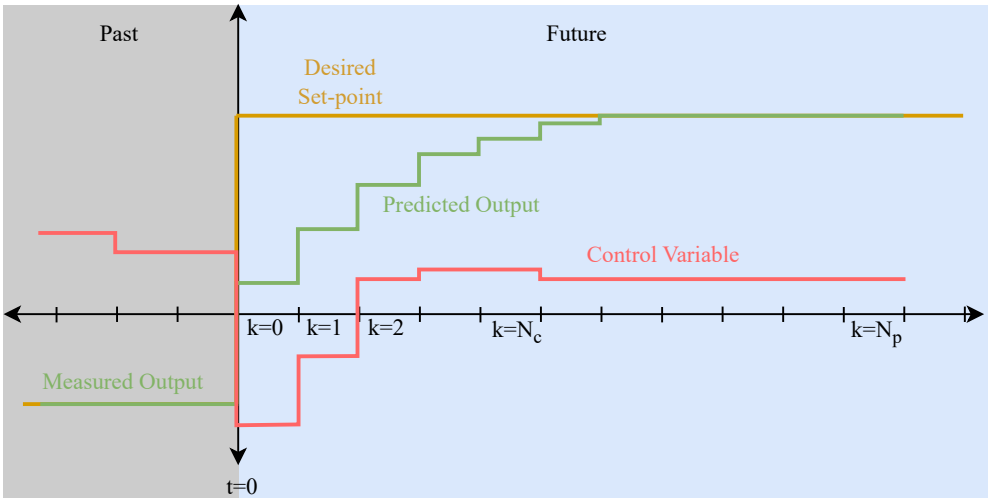

Figure 2.1. Receding horizon of the NMPC method.

2.1.4 Move-blocking

The number of control moves for the NMPC method can be reduced by using move blocking. This decreases the complexity of the objective function by reducing the degrees of freedom of the optimisation problem (Gros et al., 2020). Move-blocking is when the same control input is used for a predefined number of time steps N_b of the control horizon N_c (Cagienard et al., 2007). For example, if $N_b = 4$ and $N_c = 12$, there are three control moves within the control horizon where each control move is held constant for 4 time steps.

2.1.5 Warm-starting

The initial conditions input to the optimisation problem is very important because a nonlinear problem will have multiple local maxima and minima. For the optimisation algorithm to obtain a global minimum, the initial guess should be close to the optimal solution. A good initial guess will also ensure that the convergence of the optimisation routine is fast. Warm-starting is a method used to choose an initial guess input. A typical warm-starting method for the NMPC algorithm is to use the previously calculated control input as the initial control input for the next control time step k (Wang and Boyd, 2010). In other words, the previous control input U_k is shifted by one time step and the last control input U_{N_c} is set to U_{N_c-1} (Gros et al., 2020).

2.1.6 Suboptimal update methods

One of the optimization routines that can be used to solve (2.1) is known as sequential quadratic programming (SQP) which starts with an initial guess of $\mathbf{U} = U_k, U_{k+1}, \ldots, U_{k+N_c}$. These SQP algorithms

are fast update methods that produce an approximation of the optimal solution to (2.1) which leads to faster computational times (Wolf and Marquardt, 2016).

This suboptimal update method gives an update based on the nonlinear optimization problem with the initial parameters X_0. The parameters in the case of the objective function (2.1) would be the initial conditions of the plant at time step $k = 0$. The suboptimal update is calculated by solving an iteration of a QP problem to produce a search direction and thus improve the initial guess input $\mathbf{U}$. Multiple QP problems are then solved to iteratively change $\mathbf{U}$ and minimize (2.1). The reader is referred to Section 3.3.1 for an in-depth description of the SQP method.

2.1.7 Real-time iteration

The real-time iteration (RTI) method was introduced as a solution to the real-time dilemma. The real-time dilemma is described by the delay between obtaining a state estimation for the system and calculating a future input strategy according to the state estimation. The calculations performed to calculate a new control input can be significantly long, that in turn makes the state estimation outdated when the optimization problem is solved (Gros et al., 2020).

The RTI method negates this real-time dilemma by obtaining a state estimate at the latest possible time. The state estimate in conjunction with preliminary calculations is then used to solve the objective function with one full Newton-method step (Schwenzer et al., 2021; Gros et al., 2020; Wolf and Marquardt, 2016). The RTI method has been implemented in multiple applications including distillation columns, gasoline engines, aircrafts, kites and robots (Wolf and Marquardt, 2016).

2.1.8 Reinforcement Learning

Online computation can take a lot of computational power which gives rise to the explicit MPC (EMPC) method. EMPC is a method that does all the controller calculations before the controller is implemented. The EMPC controller decides on a control input determined from a look-up table which was calculated by solving an optimization problem for multiple situations of the plant to be controlled (Wang and Boyd, 2010; Schwenzer et al., 2021).

Recently, reinforcement and deep learning algorithms have been employed to replace the look-up table of the EMPC method with a neural network (Chen et al., 2018; Zhang et al., 2021). However, using reinforcement learning (RL) as a replacement for EMPC method can be challenging because the neural network needs to be trained offline and might require thousands of computations (Nian et al., 2020).

The design of the reward function, along with including the input, output and state constraints of the respective plant to be controlled, can be a challenging process when considering RL.

RL control law is derived from training and does not operate on a plant model. Thus, the control law is not easily explainable. Since safety is a concern in processing plants, RL methods can be a risk because the control law is not deterministic (Nian et al., 2020).

2.1.9 Dynamic programming to static programming

MPSP combines the philosophies of NMPC and ADP and formulates the optimisation problem with the framework of static optimisation. The MPSP technique converts a dynamic programming problem into a static programming problem, which makes it applicable to multiple classes of nonlinear optimal control problems (Padhi, 2008).

Because the MPSP technique is derived from an optimization control perspective, the method is generic and can be used for multiple problems (Padhi, 2008). Multiple case studies have shown that MPSP has faster computational times compared to traditional methods (Le Roux et al., 2014; Tripathi and Padhi, 2016; Zhang et al., 2016; Bin et al., 2018; Hong et al., 2019; Wang et al., 2019; Biswas et al., 2022). There are few studies of this method in the mineral processing field. Thus, MPSP is investigated in this thesis as a possible alternative for NMPC control of mineral processing plants.

2.2 MODEL PREDICTIVE STATIC PROGRAMMING

2.2.1 Overview of MPSP

The MPSP algorithm was invented by Padhi and Kothari (2009) and did not include any input or output constraints. The initial method solved an input deviation objective function and was not initially designed to include set-point tracking. The unconstrained MPSP algorithm was simulated on a boost guidance scheme for ballistic missiles (Padhi and Kothari, 2009).

The MPSP algorithm was then implemented in simulation to an air-to-air missile guidance scheme (Bhitre and Padhi, 2014). The simulations handled inequality state constraints by using a slack variable approach which transforms the inequality constraints into an unconstrained problem.

In Kumar and Padhi (2014) the MPSP algorithm was extended to include set-point reference tracking. This reference tracking MPSP (T-MPSP) included an output error weighting to the objective function

of the original MPSP algorithm. The new adaptation of the MPSP algorithm still did not include any input, state or output constraints.

Another way MPSP was manipulated to include input constraints is done on a guidance law for air-to-ground missile cooperation attacks in Li et al. (2019). The input constraints are implemented by using the Courant penalty function. The size of the input constraint violation is added to the cost function of the original problem and multiplied by a factor of 10^7 (Li et al., 2019).

The T-MPSP method was expanded to include input and state constraints by Kumar et al. (2019). This extension of the T-MPSP algorithm shows the current state of the MPSP method at the time of this dissertation. The T-MPSP technique which includes input and state constraints is shown in the following section.

2.2.2 Mathematical Derivation

A nonlinear system is written in discrete form as

$$\begin{aligned} X_{k+1}^i &= F_k\left(X_k^i, U_k^i\right) \\ Y_k^i &= h_k\left(X_k^i, U_k^i\right), \end{aligned} \tag{2.3}$$

where $X_k \, \varepsilon \, \Re^n$, $U_k \, \varepsilon \, \Re^m$ and $Y_k \, \varepsilon \, \Re^p$ represent the states, inputs and outputs of the system respectively (Kumar and Padhi, 2014). The subscript k represents the time step and the superscript i represents the iteration index. The MPSP method aims to calculate a control history U_k^{i+1}, $k = 1, 2, ..., N$, so that the output Y_k^{i+1} will converge to the desired output Y_k^* for $k = 1, 2, ..., N$. The MPSP technique requires multiple iterations to converge, and the user can define the convergence of the technique according to specific requirements. The MPSP might not converge from the first iteration if the initial guess of the inputs history U_k^0 is poor.

The relationship of the states, inputs and outputs between two different iteration intervals at time step k is defined as

$$\begin{aligned} X_k^{i+1} &\triangleq X_k^i + \Delta X_k^i \\ U_k^{i+1} &\triangleq U_k^i + \Delta U_k^i \\ Y_k^{i+1} &\triangleq Y_k^i + \Delta Y_k^i. \end{aligned} \tag{2.4}$$

The output Y_k^{i+1} can be expanded using small error approximation, where the higher order terms of the Taylor series expansion is neglected.

$$\begin{aligned} Y_k^{i+1} &= h(X_k^{i+1}) = h(X_k^i + \Delta X_k^i, U_k^i + \Delta U_k^i) \\ &\approx Y_k^i + \left[\frac{\partial Y_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} \Delta X_k^i + \left[\frac{\partial Y_k}{\partial U_k}\right]_{(X_k^i, U_k^i)} \Delta U_k^i \end{aligned} \tag{2.5}$$

Substituting Y_k^{i+1} from (2.4) into (2.5) and making ΔY_k^i the subject, the equation

$$\Delta Y_k^i = Y_k^{i+1} - Y_k^i \approx \left[\frac{\partial Y_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} \Delta X_k^i + \left[\frac{\partial Y_k}{\partial U_k}\right]_{(X_k^i, U_k^i)} \Delta U_k^i, \tag{2.6}$$

is obtained. Using (2.4), small error approximation can be applied to X_{k+1}^{i+1} such that

$$\begin{aligned} X_{k+1}^{i+1} &= F_k\left(X_k^{i+1}, U_k^{i+1}\right) \\ &= F_k\left(X_k^i + \Delta X_k^i, U_k^i + \Delta U_k^i\right) \\ &\approx X_{k+1}^i + \left[\frac{\partial F_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} \Delta X_k^i + \left[\frac{\partial F_k}{\partial U_k}\right]_{(X_k^i, U_k^i)} \Delta U_k^i \\ \Delta X_{k+1}^i &\approx \left[\frac{\partial F_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} \Delta X_k^i + \left[\frac{\partial F_k}{\partial U_k}\right]_{(X_k^i, U_k^i)} \Delta U_k^i. \end{aligned} \tag{2.7}$$

Assuming that the states, inputs and outputs all have small deviations, (2.6) and (2.7) can be rewritten as

$$\begin{aligned} dY_k^i &= \left[\frac{\partial Y_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} dX_k^i + \left[\frac{\partial Y_k}{\partial U_k}\right]_{(X_k^i, U_k^i)} dU_k^i \\ dX_{k+1}^i &= \left[\frac{\partial F_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} dX_k^i + \left[\frac{\partial F_k}{\partial U_k}\right]_{(X_k^i, U_k^i)} dU_k^i, \end{aligned} \tag{2.8}$$

where dX_k^i and dU_k^i are the deviations in the state and inputs at the k-th time step and the i-th iteration respectively. Writing the output deviation dY_k^i in terms of the state and input deviations at time steps $(k-1)$, $(k-2)$, ... , until the first time step, gives

$$\begin{aligned} dY_k^i = &\left[\frac{\partial Y_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]_{(X_{k-1}^i, U_{k-1}^i)} \cdots \left[\frac{\partial F_1}{\partial X_1}\right]_{(x_1^i, U_1^i)} dX_1^i \\ &+ \left[B_1^k\right]^i dU_1^i + \cdots + \left[B_{k_1}^k\right]^i dU_{k-1}^i, \end{aligned} \tag{2.9}$$

where $[B_j^k]^i$ is the sensitivity matrix at the i^{th} iteration, for $j = 1, 2, 3, ..., k-1$ defined as

$$\begin{aligned} \left[B_j^k\right]^i = &\left[\frac{\partial Y_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]_{(X_{k-1}^i, U_{k-1}^i)} \\ &\cdots \left[\frac{\partial F_{j+1}}{\partial X_{j+1}}\right]_{(X_{j+1}^i, U_{j+1}^i)} \left[\frac{\partial F_j}{\partial X_j}\right]_{(X_j^i, U_j^i)} \\ \left[B_k^k\right]^i = &\left[\frac{\partial Y_k}{\partial U_k}\right]_{(X_k^i, U_k^i)}. \end{aligned} \tag{2.10}$$

The state error at the first time step dX_1 in (2.9) is zero if it is assumed the states are known at that time. This means that the output deviation can reduce to

$$dY_k^i = \sum_{j=1}^{k-1} \left[B_j^k\right]^i dU_j^i, \tag{2.11}$$

and the state deviation can reduce to

$$dX_k^i = \sum_{j=1}^{k-1} \left[A_j^k\right]^i dU_j^i, \tag{2.12}$$

where

$$\left[A_j^k\right]^i = \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]_{(X_{k-1}^i, U_{k-1}^i)} \cdots \left[\frac{\partial F_{j+1}}{\partial X_{j+1}}\right]_{(X_{j+1}^i, U_{j+1}^i)} \left[\frac{\partial F_j}{\partial U_j}\right]_{(X_j^i, U_j^i)}. \tag{2.13}$$

While deriving (2.11) it is clear that the output deviation is independent of the previous state and input values. The input is a decision variable and can change independently at any point in time. It should be noted that (2.11) represents the sensitivity of the output dY_k^i at the k-th iteration with respect to the input changes dU_j^i at all the previous grid points ($j = 1, 2, ..., k-1$). Calculating $[B_j^k]^i$ and $[A_j^k]^i$ for all $k = 1, 2, 3, \ ..., \ N$ where N represents the control and prediction horizon, can be computationally heavy. The following recursive algorithm reduces the computational cost, where $j, k = 1, \ 2, \ ..., N$.

$$\left.\begin{aligned}
&\left[\phi_k^k\right]^i = I_{n\times n} \\
&\left[\phi_j^k\right]^i = \left[\phi_{j+1}^k\right]^i \left[\frac{\partial F_j}{\partial X_j}\right]_{(X_j^i, U_j^i)} \\
&\left[A_j^k\right]^i = \left[\phi_{j+1}^k\right]^i \left[\frac{\partial F_j}{\partial U_j}\right]_{(X_j^i, U_j^i)} \\
&\left[B_j^k\right]^i = \left[\frac{\partial Y_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} \left[A_j^k\right]^i
\end{aligned}\right\} \quad \forall j < k$$

$$\begin{aligned}
&\left[B_j^k\right]^i = \left[\frac{\partial Y_k}{\partial U_k}\right] (X_k^i, U_k^i) && \forall j = k \\
&\left[A_j^k\right]^i = [0]_{n\times m} && \forall j \geq k \\
&\left[B_j^k\right]^i = [0]_{p\times m} && \forall j > k
\end{aligned} \tag{2.14}$$

2.2.3 Cost function

In (2.11) there are $(N-1)m$ unknowns and p equations. In general $p << (N-1)m$ which indicates an under-constrained system of equations. A cost function can be included for tracking a desired output.

The cost function chosen for each i-th iteration is

$$\begin{aligned} J^i =& \frac{1}{2}\sum_{k=1}^{N}\left(Y_k^{i+1}-Y_k^*\right)^{\mathrm{T}} Q_k \left(Y_k^{i+1}-Y_k^*\right) \\ &+\frac{1}{2}\sum_{k=1}^{N}\left(U_k^{i+1}-U_k^i\right)^{\mathrm{T}} R_k \left(U_k^{i+1}-U_k^i\right), \end{aligned} \tag{2.15}$$

where Y_k^* is the desired output, Q_k is the output weighting matrix and R_k is the input deviation weighting matrix. The cost function can be simplified further by using the definition in (2.4) and assuming small deviation steps to

$$\begin{aligned} J^i =& \frac{1}{2}\sum_{k=2}^{N}\left(\Delta Y_k^i-\Delta Y_k^*\right)^T Q_k \left(\Delta Y_k^i-\Delta Y_k^*\right) \\ &+\frac{1}{2}\sum_{k=1}^{N-1}\left(\Delta U_k^i\right)^T R_k \left(\Delta U_k^i\right), \end{aligned} \tag{2.16}$$

where $\Delta Y_k^* = Y_k^* - Y_k^i$ is the output error with respect to the desired output Y_k^*. Minimizing the cost function in (2.16) will result in the measured output to draw closer to the desired output at each grid point for the next iteration ($Y_k^{i+1} \to Y_k^*,\ \forall\, k = 2,3,...,N$). Assuming that the output error is small and that the input have small deviations, (2.16) can be written as

$$\begin{aligned} J^i =& \frac{1}{2}\sum_{k=2}^{N}\left(dY_k^i-\Delta Y_k^*\right)^T Q_k \left(dY_k^i-\Delta Y_k^*\right) \\ &+\frac{1}{2}\sum_{k=1}^{N-1}\left(dU_k^i\right)^T R_k \left(dU_k^i\right). \end{aligned} \tag{2.17}$$

2.2.4 Unconstrained MPSP

In the unconstrained MPSP case, where there are no limitations to the states and inputs, the global minimum of the objective function (2.17) can be obtained through the optimality conditions of $\partial J^i/\partial(dU_l^i) = 0,\ \forall l = 1,...,N$ (Kumar et al., 2019). The iteration index i is excluded from the following equations for readability. The optimality condition is then

$$\frac{\partial J}{\partial\left(dU_l\right)} = \sum_{k=1}^{N}\left(B_l^{kT} Q_k \sum_{j=1}^{k} B_j^k dU_j\right) - \sum_{k=1}^{N} B_l^{kT} Q_k \Delta Y_k^* + R_l dU_l. \tag{2.18}$$

The first term on the right-hand side of the equation can be simplified further as

$$\begin{aligned} \sum_{k=1}^{N}\left(B_l^{kT} Q_k \sum_{j=1}^{k} B_j^k dU_j\right) =& \sum_{m=1}^{N} B_l^{mT} Q_m B_1^m dU_1 + \cdots + \sum_{m=N}^{N} B_l^{mT} Q_m B_N^m dU_N \\ =& C_{l1} dU_1 + C_{l2} dU_2 + \cdots + C_{lN} dU_N. \end{aligned} \tag{2.19}$$

Equations (2.18) and (2.19) can be combined and simplified with the definition of optimality as $\partial J^i/\partial(dU_l^i) = 0;\ \forall l = 1,...,N$, to obtain

$$\sum_{k=1}^{N} B_l^{kT} Q_k \Delta Y_k^* = C_{l1} dU_1 + \cdots + C_{lN} dU_N + R_l dU_l, \tag{2.20}$$

where the matrix $C_{lj}\ \varepsilon\ \Re^{N\times N}$ is defined as

$$C_{lj} = \sum_{e=j}^{N} (B_l^e)^{\mathrm{T}} Q_e B_j^e, \tag{2.21}$$

for all $l = 1,...,N$ and $j = 1,...,N$. The solution to this system of equations can be written as

$$[dU_l] = \left[C_{lj} + \delta_{lj} R_l\right]^{-1} [b_l], \tag{2.22}$$

where δ_{lj} is the Kronecker-delta function and b_l is

$$b_l = \sum_{k=1}^{N} \left(B_l^k\right)^{\mathrm{T}} Q_k \Delta Y_k^* \tag{2.23}$$

for all $l = 1,...,N$.

2.2.5 Constrained MPSP

Constrained MPSP, where constraints are added to the system, uses a condensed form of the cost function in (2.16) by substituting (2.11) and converting it into vector format to get

$$\begin{aligned} J^i = &\frac{1}{2}(\delta U^i)^T \left(R_k + ([B]^i)^T Q_k [B]^i\right) \delta U^i \\ &- (\delta U^i)^T \left([B]^i\right)^T Q_k (\Delta Y^{*i}) \\ &+ \frac{1}{2}(\Delta Y^{*i})^T Q_k \Delta Y^{*i}, \end{aligned} \tag{2.24}$$

where Q_{mpsp}, R_{mpsp}, ΔY^{*i} and $[B]^i$ are

$$Q_k \triangleq \mathrm{diag}([Q_1],[Q_2],\ ...,\ [Q_N])$$

$$R_k \triangleq \mathrm{diag}([R_1],[R_2],\ ...,\ [R_N])$$

$$\Delta Y^{*i} \triangleq \left[(\Delta Y_1^{*i})^T\ \ (\Delta Y_2^{*i})^T\ \ \cdots (\Delta Y_N^{*i})^T\right]$$

$$[B]^i \triangleq \begin{bmatrix} [B_1^1]^i & [B_2^1]^i & \cdots & [B_N^1]^i \\ [B_1^2]^i & [B_2^2]^i & \cdots & [B_N^2]^i \\ \vdots & \vdots & \ddots & \vdots \\ [B_1^N]^i & [B_2^N]^i & \cdots & [B_N^N]^i \end{bmatrix}.$$

The state and input constraints applied in (2.24) are

$$\begin{bmatrix} [A]^i \\ -[A^i] \\ I \\ -I \end{bmatrix} \delta U^i \leq \begin{bmatrix} X^{UB} - X^i \\ X^i - X^{LB} \\ U^{UB} - U^i \\ U^i - U^{LB} \end{bmatrix}, \tag{2.25}$$

where X^{UB} and X^{LB} are the upper and lower bound constraints of the states respectively and U^{UB} and U^{LB} are the upper and lower bound constraints of all the control inputs respectively. The matrix $[A]^i$ is

the same as represented in (2.13) but in matrix form

$$[A]^i \triangleq \begin{bmatrix} [A_1^1]^i & [A_2^1]^i & \dots & [A_N^1]^i \\ [A_1^2]^i & [A_2^2]^i & \dots & [A_N^2]^i \\ \vdots & \vdots & \ddots & \vdots \\ [A_1^N]^i & [A_2^N]^i & \dots & [A_N^N]^i \end{bmatrix}.$$

Finally, δU^i is the small control errors

$$\delta U^i \triangleq \begin{bmatrix} (dU_1^i)^T & (dU_2^i)^T & \cdots & (dU_N^i)^T \end{bmatrix}.$$

Output constraints can be added by using the definition in (2.11) to obtain

$$\begin{bmatrix} [B]^i \\ -[B^i] \end{bmatrix} \delta U^i \leq \begin{bmatrix} Y^{UB} - Y^i \\ Y^i - Y^{LB} \end{bmatrix}, \tag{2.26}$$

where Y^{UB} and Y^{LB} are the upper and lower output constraints respectively. The cost function in (2.24) and the constraints in (2.25) and (2.26) can be solved with any standard quadratic programming solving method (Kumar et al., 2019).

2.2.6 Model Predictive Static Programming Summary

1. Obtain an initial inputs guess history U_k^0 and start the indexing of the technique at zero ($i = 0$), $\forall k = 1, 2, ...N$.
2. Obtain the state trajectory X_k^i, $\forall k = 1, 2, ..., N$, using the known initial states X_1, initial control inputs U_k^i, and the system dynamics in (2.3).
3. Determine the output trajectory Y_k^i from the state trajectory and calculate the output error ΔY_k^*, which is the error between the desired output and the current output trajectory, $\forall k = 1, 2, ..., N$.
4. Stop the algorithm if the output error is smaller than the user-defined tolerance, i.e. $\sum_{k=1}^{N} |\Delta Y_k^{*i}| \leq \varepsilon_Y$, or if the input control history has converged within a user-define tolerance, i.e. $\sum_{k=1}^{N} |U_k^i - U_k^{i-1}| \leq \varepsilon_U$. This step is valid after the first iteration $i \geq 1$. If the above-mentioned tolerances are not satisfied, continue through steps 5 and 6, then 2 to 4.
5. Increase the iteration index i and calculate the MPSP sensitivity matrices A_j^k and B_j^k by using the recursive method in (2.14).
6. Update the control input for the next iteration $(i+1)$ by minimising the objective function (2.24) for the constrained case, or by solving the explicit solution (2.22) for the unconstrained case. After the control inputs are calculated, return to step 2.

Step 5 is the main feature of the constrained MPSP method because it converts the objective function (2.1) of the NMPC, which is an NLP problem, into a lower-dimensional static problem with a strictly convex QP problem format (Kumar et al., 2019).

2.3 CHAPTER SUMMARY

The MPSP algorithm relates the deviations of the outputs or states to the deviations of a control inputs through the representations of the sensitivity matrices. These sensitivity matrices need to be calculated for every iteration and can be obtained recursively. The sensitivity matrices are used to define the state and output constraints as well as the objective function, which is a strictly convex quadratic problem. To calculate the sensitivity matrices, the partial derivatives of the plant model need to be obtained.

The NMPC algorithm only uses the nonlinear discrete plant model to predict the future outputs of the plant. Thus, the objective function is a nonlinear problem. The implementation of the NMPC controller does not require partial derivative calculations to define the objective function of the algorithm, whereas the MPSP algorithm needs the derivative calculations to set up the objective function.

The main difference between the two algorithms is that the final objective function that needs to be solved for every iteration is linear for the MPSP method and nonlinear for the NMPC algorithm.

CHAPTER 3 METHODOLOGY

3.1 CHAPTER OVERVIEW

The chapter describes the methodology followed to implement the constrained MPSP and the constrained NMPC controllers to be able to compare their computational efficiency. A flow diagram of the method is shown in Fig. 3.1. The software tools used are given in this chapter as well as a short description of the optimization algorithms to motivate the use thereof. This chapter only describes the implementation of the different controllers which can be generalized to any plant model.

3.2 PARTIAL DERIVATIVES OF THE MODEL

The MPSP method uses the partial derivatives of the specific nonlinear plant model in its algorithm. These partial derivatives were derived analytically using a library for symbolic mathematics (Meurer et al., 2017). The required partial derivates are shown in Section 2.2 for the calculation of the sensitivity matrices $[A]$ and $[B]$.

3.3 OPTIMIZATION ROUTINES

Multiple optimization routines were used during the simulation of the respective plant model. During the grinding mill simulation in Section 4.3 both the NLP and QP solvers were used. The motivation for the use of different solvers will be given after the NLP algorithm is explained. Below is a summary of the algorithms used in the MPSP and NMPC simulations.

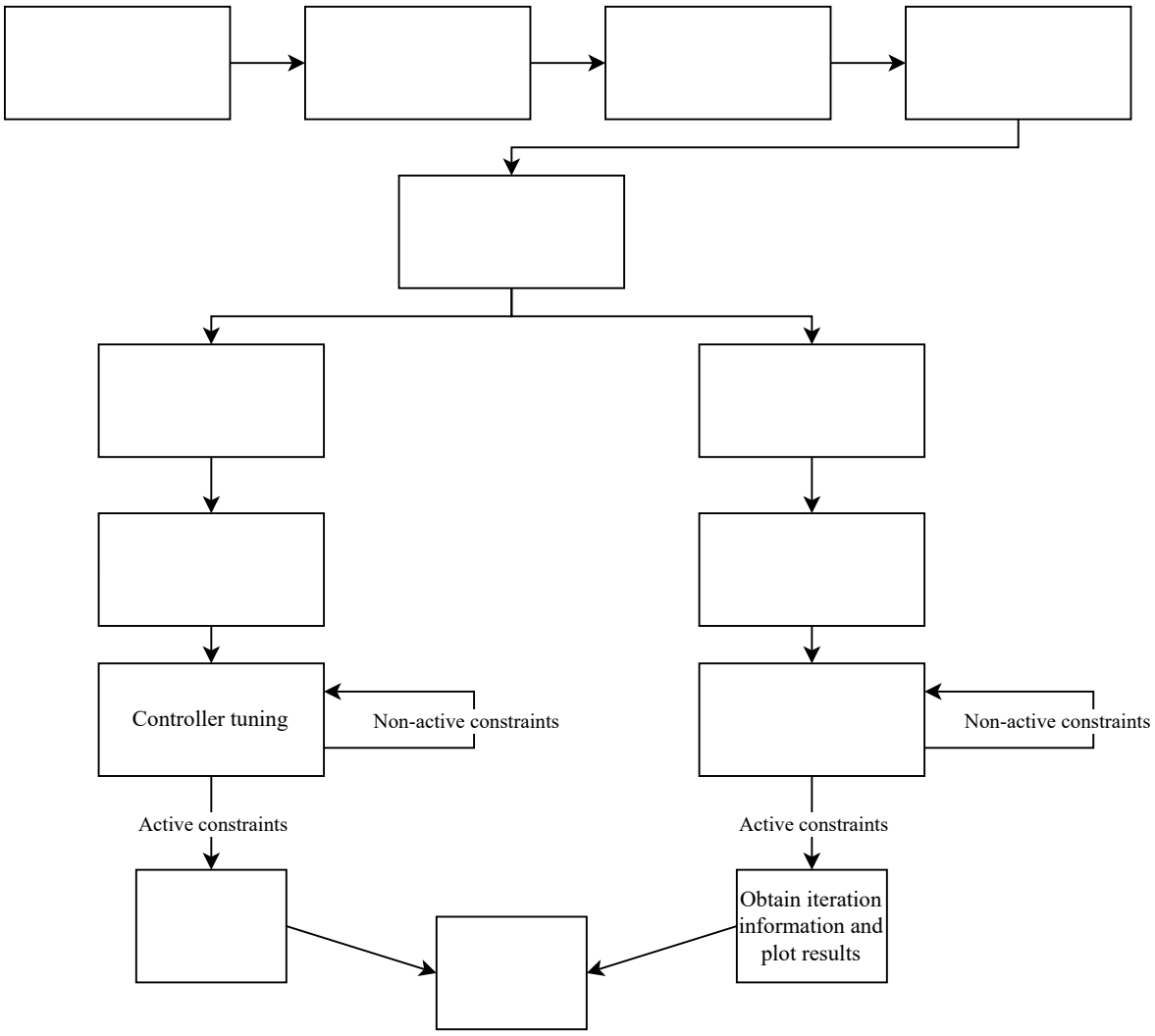

Figure 3.1. Methodology flow diagram for both simulation controllers.

3.3.1 Sequential Quadratic Programming

A computationally efficient method to solve a general NLP problem is known as sequential quadratic programming (Kraft, 1988). The general NLP problem is described as

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & c_j(x) = 0, \qquad j = 1,\ldots,m_e \\ & c_j(x) \geq 0, \qquad j = m_e+1,\ldots,m_i \\ & lb \leq x \leq ub, \end{aligned} \tag{3.1}$$

where c_j represents the equality and inequality constraints, m_e and m_i represents the number of equality and inequality constraints respectively, lb represents the input lower bounds, and ub represents the input

upper bounds. The NLP is solved iteratively by updating the inputs during iterative steps as

$$x^{k+1} = x^k + \alpha^k d^k, \tag{3.2}$$

where x^k is the current input, x^{k+1} is the next iteration input, α^k is the step length and d^k is the search direction within the k^{th} step.

The search direction is determined by solving a quadratic approximation of the Lagrange function of (3.1) as well as a linear approximation of the constraints as

$$\begin{aligned} \min_d \quad & \frac{1}{2} d^T B^k d + \nabla f(x^k) d \\ \text{s.t.} \quad & \nabla c_j(x^k) d + c_j(x^k) = 0, \quad j = 1, \ldots, m_e \\ & \nabla c_j(x^k) d + c_j(x^k) \geq 0, \quad j = m_e + 1, \ldots, m_i, \end{aligned} \tag{3.3}$$

where $\nabla f(x^k)$ and $\nabla c_j(x^k)$ are the gradients of the nonlinear function $f(x^k)$ and the constraints $c_j(x^k)$. The above equation has the structure of a quadratic programming problem. The matrix B^k is calculated as

$$B = \nabla^2_{xx} L(x, \lambda), \tag{3.4}$$

where

$$L(x, \lambda) = f(x) - \sum_{j=1}^{m} \lambda_j c_j(x) \tag{3.5}$$

is known as the Lagrange function. The constraints in (3.3) can become inconsistent during the different iteration steps. The algorithm thus includes an additional variable δ to overcome this problem. Thus, the new QP is defined as

$$\begin{aligned} \min_d \quad & \frac{1}{2} d^T B^k d + \nabla f(x^k) d + \frac{1}{2} \rho^k (\delta^k)^2 \\ \text{s.t.} \quad & \nabla c_j(x^k) d + \delta^k c_j\left(x^k\right) = 0, \quad j = 1, \ldots, m_e \\ & \nabla c_j(x^k) d + \delta^k_j c_j(x^k) \geq 0, \quad j = m_e + 1, \ldots, m_i \\ & 0 \leq \delta^k \leq 1, \end{aligned} \tag{3.6}$$

where δ is defined as

$$\delta^k_j := \begin{cases} 1, & \text{if } c_j\left(x^k\right) > 0 \\ \sigma^k, & \text{otherwise.} \end{cases} \tag{3.7}$$

The reader is referred to Kraft (1988) for further reading on the algorithm. In summary, the NLP problem is solved iteratively by calculating a step length (which is not included in this report) and a search direction by solving a QP problem.

Using the SQP method for NLP problems means that multiple QP problems are solved iteratively as illustrated above. In the case of the MPSP algorithm, the objective function in (2.24) is already in a strictly convex QP problem format. Thus, it is redundant to implement the NLP problem solver on the MPSP algorithm.

3.3.2 MPSP controller implementation

The constrained MPSP method is implemented in simulation, where the objective function for each iteration has a strictly convex QP problem format. The objective function in (2.24) and the constraints in (2.25) can be solved with any standard quadratic programming solving method (Kumar et al., 2019). The solver used for the simulation is *quadprog* (Goldfarb and Idnani, 1983). This algorithm solves convex quadratic programs in the standard form, which is defined as

$$\begin{aligned} \min_{x} \quad & f(x) = \frac{1}{2}x^T Px + q^T x \\ \text{s.t.} \quad & Gx \leq h \\ & Ax = b \\ & lb \leq x \leq ub, \end{aligned} \tag{3.8}$$

where G and h is used to describe the linear inequality constraints, and A and b is used to describe the equality constraints of the QP problem. Comparing (3.8) to (2.24) it is evident that

$$P = R_{mpsp} + ([B]^i)^T Q_{mpsp} [B]^i$$

and

$$q = -\left(([B]^i)^T Q_{mpsp} (\Delta Y^{*i})\right)^T.$$

The inequality constraints are used as defined in (2.25) and (2.26). In the case of the MPSP algorithm, the G matrix is a function of the sensitivity matrices $[A]^i$ and $[B]^i$.

3.3.3 NMPC controller implementation

The minimization problem in (2.1) can be solved using an appropriate numerical optimization routine (Nocedal and Wright, 2006). The solver used during the simulation of the NMPC controller is the *Sequential Least-Squares Quadratic Programming* algorithm (Kraft, 1988).

A sequential method is used for the NMPC method because of the advantage of being able to stop the algorithm at a set iteration count, termination condition or time constraint (Cannon, 2004).

3.4 CONTROLLER TUNING

Both the MPSP controller and the NMPC controller are subject to constraints. These constraints can cause the computational time of each optimization algorithm to increase. Thus, the tuning of each controller is important.

There are multiple methods of tuning model predictive controllers, but the initial tuning parameters to obtain satisfactory results for the NMPC controllers can be difficult. For the initial tuning of the controllers in the case of the grinding mill circuit, the weights were chosen to normalize the inputs and outputs according to a prioritised output set-point (Le Roux et al., 2014). The plant disturbance was then altered until at least one of the constraints of the plant were active. The activation of constraints is important for the study, as the constrained control algorithms are compared to one another and not the unconstrained alternatives. The simulations need to verify that the algorithms do not violate any constraints.

In the case of the flotation circuit, the initial tuning suggestions of Alhajeri and Soroush (2020) were followed for $Q = I$ and $R = \rho I$, where ρ is a scalar value usually smaller than one. The tuning was then altered until at least one of the plant constraints were active.

3.5 MEASURING COMPUTATIONAL TIME

Because of the differences of the proposed controllers, a formal definition of the computational times is required. For this study, the calculation time is defined as the time measured to calculate a new control move after taking a sampling instance for each time step.

3.6 CHAPTER SUMMARY

The objective functions are the main contributors to the computational time of each controller. The NMPC controller requires an SQP solver to optimise the objective function, whereas the MPSP controller only requires a QP solver to optimise the objective function. The computational time is evaluated as the time between sampling instances.

CHAPTER 4 MPSP APPLIED TO A MILLING CIRCUIT

4.1 CHAPTER OVERVIEW

The constrained MPSP controller in Section 2.2 and the constrained NMPC controller in Section 2.1 are applied in simulation to a grinding mill circuit. In this chapter, the process description and the chosen nonlinear plant model of a single-stage grinding mill circuit are given. In Section 4.3 the simulation configuration is discussed, which includes the definition of the input and output constraints of the grinding mill, plant disturbances, and the specific controller configurations.

4.2 GRINDING MILL CIRCUIT PROCESS DESCRIPTION

A single-stage grinding mill circuit is shown in Fig. 4.1 (Le Roux and Steyn, 2022; Le Roux et al., 2013). The variables in Fig. 4.1 are described in Table 4.1. Only a brief overview of the process is given below.

The mill, sump and hydrocyclone are the three main elements of the grinding mill circuit. The mill has four inputs: underflow from the hydrocyclone, mill inlet water (u_{MIW}), mill feed ore (u_{MFO}) and mill feed balls (u_{MFB}). These four inputs mix in the mill to form a slurry. The fraction of the mill filled with charge is given by y_{JT}. The slurry is discharged from the mill into a sump through an end-discharge screen. The slurry inside the sump is diluted with water (u_{SFW}) after which the slurry is pumped into the hydrocyclone. The volume of slurry in the sump and the feed flow-rate of the slurry into the cyclone is represented by y_{SVOL} and u_{CFF} respectively. The hydrocyclone separates small particles from large particles. The hydrocyclone overflow contains the small particles which are sent to a downstream process. The fraction of particles in the overflow smaller than 75 μm is given by y_{PSE}. The hydrocyclone underflow contains large particles which return to the mill for further breakage.

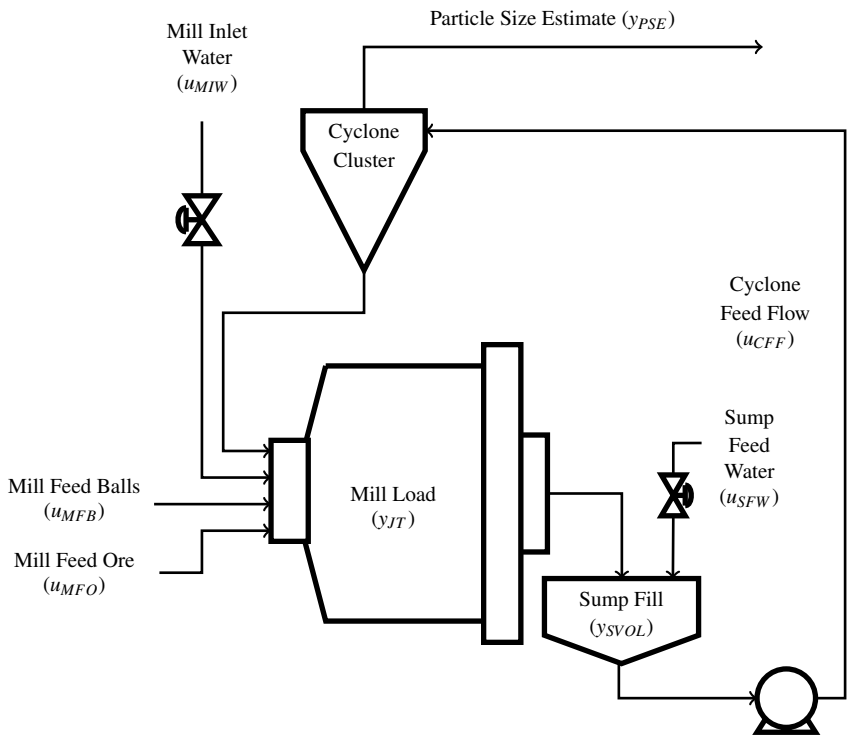

Figure 4.1. A single-stage grinding mill circuit. Adapted from Le Roux and Steyn (2022), with permission.

4.2.1 Grinding mill circuit process model

A brief overview of the model of the grinding mill circuit in Fig. 4.1 is given below. A complete description can be found in Le Roux et al. (2013) and Le Roux and Steyn (2022). The model nomenclature is given in Table 4.2. The variable and parameter values were taken from Le Roux et al. (2014).

The charge inside the mill is divided into five states: rocks, solids, fines, balls and water. The solids are ore that can discharge from the mill, whereas rocks are ore too large to discharge via the end-discharge screen. The solids are the sum of the fine and coarse ore, where fine ore is classified as a product below specification size, and coarse ore is classified as a product above specification size (Le Roux et al., 2013).

The state-space model of the grinding mill circuit is

$$
\begin{aligned}
\dot{x}_{mw} &= u_{MIW} - \frac{d_q \varphi x_{mw} x_{mw}}{x_{ms} + x_{mw}} + Q_{cwu} \\
\dot{x}_{ms} &= \frac{u_{MFO}}{\rho_o}(1 - \alpha_r) - \frac{d_q \varphi x_{mw} x_{ms}}{x_{ms} + x_{mw}} + Q_{csu} + \\
&\quad \frac{\varphi P_{mill}}{\rho_o K_{RC}} \left(\frac{x_{mr}}{x_{mr} + x_{ms}} \right) \\
\dot{x}_{mf} &= \frac{u_{MFO}}{\rho_o} \alpha_f - \frac{d_q \varphi x_{mw} x_{mf}}{x_{ms} + x_{mw}} + Q_{cfu} + \frac{P_{mill}}{\rho_o K_{FP}} \\
\dot{x}_{mr} &= \frac{u_{MFO}}{\rho_o} \alpha_r - \frac{P_{mill} \varphi}{\rho_o K_{RC}} \left(\frac{x_{mr}}{x_{mr} + x_{ms}} \right) \\
\dot{x}_{mb} &= \frac{u_{MFB}}{\rho_b} - \frac{P_{mill} \varphi}{K_{BC}} \left(\frac{x_{mb}}{\rho_o (x_{mr} + x_{ms}) + \rho_b x_{mb}} \right) \\
\dot{x}_{sw} &= \frac{d_q \varphi x_{mw} x_{mw}}{x_{ms} + x_{mw}} - \frac{u_{CFF} x_{sw}}{x_{sw} + x_{ss}} + u_{SFW} \\
\dot{x}_{ss} &= \frac{d_q \varphi x_{mw} x_{ms}}{x_{ms} + x_{mw}} - \frac{u_{CFF} x_{sw}}{x_{sw} + x_{ss}} \\
\dot{x}_{sf} &= \frac{d_q \varphi x_{mw} x_{mf}}{x_{ms} + x_{mw}} - \frac{u_{CFF} x_{sf}}{x_{sw} + x_{ss}},
\end{aligned}
\tag{4.1}
$$

where x_{mw}, x_{ms}, x_{mf}, x_{mr} and x_{mb} [m^3] are the volume of water, solids, fines, rocks and balls inside the mill respectively, x_{sw}, x_{ss} and x_{sf} [m^3] are the water, solids and fines inside the sump respectively, and Q_{cwu}, Q_{csu} and Q_{cfu} [m^3/h] are the cyclone water, solids and fines underflow respectively.

Table 4.1. Circuit variable descriptions.

	Manipulated Variables
u_{MIW}	Flow-rate of water to the mill [m^3/h]
u_{MFO}	Flow-rate of ore to the mill [t/h]
u_{MFB}	Flow-rate of steel balls to the mill [t/h]
u_{SFW}	Flow-rate of water to the sump [m^3/h]
u_{CFF}	Flow-rate of slurry to the hydrocyclone [m^3/h]
	Controlled Variables
y_{JT}	Fraction of the mill filled [-]
y_{SVOL}	Volume of slurry in the sump [m^3]
y_{PSE}	Fraction of particles within specification [-]

Table 4.2. Grinding mill circuit nomenclature.

Parameter	Value	Description
α_f	0.055	Fraction fines in the ore
α_r	0.465	Fraction rock in the ore
ϕ_c	0.72	Fraction of critical mill speed
α_{su}	1.50	Parameter related to fraction solids in underflow
C_1	0.6	Constant
C_2	0.7	Constant
C_3	4.0	Constant
δ_s	2.90	Power-change parameter for fraction solids in the mill
δ_v	2.90	Power-change parameter for the volume of mill filled
ρ_b	7.85	Density of steel balls [t/m^3]
ρ_o	3.2	Density of feed ore [t/m^3]
ε_c	111.85	Maximum fraction solids by volume of slurry at zero slurry flow
ε_{sv}	0.6	Parameter related to coarse split [m^3/h]
K_{BC}	90.0	Ball consumption factor [kWh/t]
K_{FP}	31.31	Fines production factor [kWh/t]
K_{RC}	8.06	Rock consumption factor [kWh/t]
φ_N	0.57	Rheology normalization factor draw
P_{max}	1670	Maximum mill motor power draw [kW]
v_{mill}	59.12	Mill volume [m^3]
v_{Pmax}	0.34	Fraction of mill volume filled for maximum power draw
d_q	84.50	Discharge rate [h^{-1}]

The outputs are

$$\begin{aligned} y_{JT} &= \frac{x_{mw}+x_{ms}+x_{mr}+x_{mb}}{v_{mill}} \\ y_{SVOL} &= x_{ss}+x_{sw} \\ y_{PSE} &= \frac{Q_{cfo}}{Q_{cso}}, \end{aligned} \tag{4.2}$$

where Q_{cfo} and Q_{cso} [m^3/h] are the volumetric flow-rates of the fines and the solids at the overflow of the hydrocyclone respectively.

The intermediate variables required in (4.1) for the mill are

$$\varphi = \begin{cases} \sqrt{1-\left(\varepsilon_c^{-1}-1\right)\frac{x_s}{x_w}}; & \frac{x_s}{x_w} \leq \left(\varepsilon_0^{-1}-1\right)^{-1} \\ 0; & \frac{x_s}{x_w} > \left(\varepsilon_0^{-1}-1\right)^{-1} \end{cases}$$
$$P_{\text{mill}} = P_{\max}\left\{1-\delta_v\left(\frac{x_{mw}+x_{mr}+x_{ms}+x_{mb}}{v_{\text{mill}}\, v_{P_{\max}}}-1\right)^2-\delta_s\left(\frac{\varphi}{\varphi_N}-1\right)^2\right\}\phi_c, \tag{4.3}$$

where φ is an empirically defined rheology factor and P_{mill} [kW] is the power draw of the grinding mill. The intermediate variables required in (4.1) and (4.2) for the hydrocyclone are

$$\begin{aligned} Q_{ccu} =& \frac{u_{CFF}\left(x_{ss}-x_{sf}\right)}{x_{sw}+x_{ss}}\left(1-C_1\exp\left(\frac{-u_{CFF}}{\varepsilon_c}\right)\right)\times \\ & \left(1-\left(\frac{x_{ss}}{C_2\left(x_{sw}+x_{ss}\right)}\right)^{C_3}\right)\left(1-\left(\frac{x_{sf}}{x_{ss}}\right)^{C_3}\right) \\ F_u =& 0.6-\left(0.6-\frac{x_{ss}}{x_{sw}+x_{ss}}\right)\exp\left(\frac{-Q_{ccu}}{\alpha_{su}\varepsilon_c}\right) \\ Q_{cwu} =& \frac{x_{sw}\left(Q_{ccu}-F_uQ_{ccu}\right)}{F_ux_{sw}+F_ux_{sf}-x_{sf}} \\ Q_{cfu} =& \frac{x_{sf}\left(Q_{ccu}-F_uQ_{ccu}\right)}{F_ux_{sw}+F_ux_{sf}-x_{sf}} \\ Q_{csu} =& q_{ccu}+\frac{x_{sf}\left(Q_{ccu}-F_uQ_{ccu}\right)}{F_ux_{sw}+F_ux_{sf}-x_{sf}} \\ Q_{cso} =& \frac{u_{CFF}x_{ss}}{x_{ss}+x_{sw}}-Q_{csu} \\ Q_{cfo} =& \frac{u_{CFF}x_{sf}}{x_{ss}+x_{sw}}-Q_{cfu}, \end{aligned} \tag{4.4}$$

where Q_{ccu} [m^3/h] is the hydrocyclone coarse underflow, Q_{cso} and Q_{cfo} [m^3/h] are the hydrocyclone solid and fines overflow respectively, and F_u is the fraction of solids in the hydrocyclone underflow.

4.3 GRINDING MILL CIRCUIT SIMULATION

4.3.1 Simulation configuration

To compare the performance of the constrained MPSP and NMPC as applied to the grinding mill circuit, the following general configuration is used.

The simulation duration time is 5 h and each controller has a sampling time of $T_s = 10$ s. A longer sampling time may allow the sump to run dry or overflow before corrective action can be taken (Coetzee et al., 2010). The nonlinear state-space description of the circuit in (4.1) is simulated using the Runge-Kutta fourth-order method.

Although it is not trivial to design an observer for the grinding mill circuit, the state of the grinding mill circuit can be estimated given sufficient industrial measurements (Le Roux et al., 2016, 2017; Le Roux and Steyn, 2022). Since the observer design falls outside the scope of this study, full-state feedback is assumed.

The ball feed-rate u_{MFB} is kept at a constant ratio with respect to the volume of the mill filled with charge y_{JT}, such that u_{MFB}/y_{JT} = 16.7. The mill water inlet u_{MIW} is kept in a ratio of 7 % with the mill feed ore u_{MFO}.

The nominal and initial values of the plant are

$$\begin{aligned} X_0 &= [x_{mw}, x_{ms}, x_{mf}, x_{mr}, x_{mb}, x_{sw}, x_{ss}, x_{sf}]^T \\ &= [3.78,\ 3.45,\ 1.08,\ 1.86,\ 9.23,\ 3.79,\ 2.11,\ 0.66]^T \end{aligned} \tag{4.5}$$

$$U_0 = [u_{MFO},\ u_{SFW},\ u_{CFF}]^T = [66.9,\ 67.1,\ 267]^T \tag{4.6}$$

$$Y_{sp} = [y_{JT},\ y_{SVOL},\ y_{PSE}]^T = [0.31,\ 5.90,\ 0.60]^T . \tag{4.7}$$

The input and output constraints for the grinding mill are

$$\begin{aligned} \begin{bmatrix} 0 \\ 0 \\ 100 \end{bmatrix} \leq \begin{bmatrix} u_{MFO} \\ u_{SFW} \\ u_{CFF} \end{bmatrix} \leq \begin{bmatrix} 100 \\ 150 \\ 500 \end{bmatrix} \\ \begin{bmatrix} 0.25 \\ 1.0 \\ 0.5 \end{bmatrix} \leq \begin{bmatrix} y_{JT} \\ y_{SVOL} \\ y_{PSE} \end{bmatrix} \leq \begin{bmatrix} 0.45 \\ 8.0 \\ 0.8 \end{bmatrix} . \end{aligned} \tag{4.8}$$

The desired set-points are kept constant at the nominal values of the plant.

The first disturbance introduced to the grinding mill is a change in the mill feed size distribution by increasing the fraction of rocks in the ore fed to the mill, α_r, with 50 % of its nominal value from $t = 0.5$ h to $t = 2.1$ h. The second disturbance introduced to the circuit is a change in the ore hardness by increasing the fines production factor, K_{FP}, with 60 % of its nominal value from t =1.5 h to t =3 h. The increase in K_{FP} is introduced as a step-change in the simulation whereas it would generally change gradually over time in an industrial plant.

The simulations were done in Python. The simulations were executed on an Intel(R) Core(TM) i5-8400 (6 Core) 2.80 GHz processor with 20 GB RAM running a Microsoft Windows 10 operating system.

4.3.2 MPSP configuration

Two MPSP controller configurations were simulated, where the prediction and control horizon is equal such that $N = 36$ according to (2.17).

The weighting matrices for the MPSP controller were chosen to normalize the inputs and outputs and prioritize set-point following of y_{PSE}, such that

$$\begin{aligned} Q_{mpsp} &= \text{diag}([36.2227, 0.025, 1510.85]) \\ R_{mpsp} &= 10^{-3}\,\text{diag}([3.481, 0.218, 0.218]), \end{aligned} \tag{4.9}$$

where Q_{mpsp} and R_{mpsp} are the weighting matrices for the MPSP objective function in (2.24) (Le Roux et al., 2014). The MPSP algorithm terminates if the control sequence has been executed 15 times, or if any of the conditions

$$\begin{aligned} \frac{\left\|Y_k^i(1) - Y_k^*(1)\right\|_2}{\left\|Y_k^*(1)\right\|_2} &< 0.5 \\ \frac{\left\|Y_k^i(2) - Y_k^*(2)\right\|_2}{\left\|Y_k^*(2)\right\|_2} &< 0.5 \\ \frac{\left\|Y_k^i(3) - Y_k^*(3)\right\|_2}{\left\|Y_k^*(3)\right\|_2} &< 0.1, \end{aligned} \tag{4.10}$$

where $Y_k^i(p)$ refers to the p-th entry in the output vector Y_k^i are met.

The first MPSP controller, $MPSP_{NLP}$, uses the same NLP solver as the NMPC controller. The second MPSP controller, $MPSP_{QP}$, uses a strictly convex QP solver.

4.3.3 NMPC configuration

Two NMPC controller configurations were simulated, where the prediction horizon is $N_p = 36$ and the control horizon is $N_c = 12$ for both controllers. The first controller, $NMPC_{B1}$, does not make use of move-blocking and the second controller, $NMPC_{B3}$, makes use of move-blocking of $N_B = 3$. Both the NMPC controllers use warm-starting (Wang and Boyd, 2010).

Similar to MPSP, the weighting matrices for the NMPC controller were chosen to normalize the inputs and outputs and prioritize set-point following of y_{PSE}, such that

$$\begin{aligned} Q_{nmpc} &= \text{diag}([5.489, 0.015, 496.694]) \\ R_{nmpc} &= 10^{-3}\,\text{diag}([3.481, 0.218, 0.218]), \end{aligned} \tag{4.11}$$

where Q_{nmpc} and R_{nmpc} are the weighting matrices for the NMPC objective function in (2.1). The NMPC algorithm terminates when the sequential programming problem optimization routine has been executed a maximum of 15 times, or the algorithm converged between iterations within a tolerance of 0.01.

The weighting matrices for the NMPC and MPSP controllers can be chosen to apply larger penalties to input deviations and output set-point tracking errors. For these larger weights, it may mean that none of the constraints are reached for the operating conditions in the simulations shown below. Because the aim is to evaluate the algorithms when constraints are active, the weighting matrices were not updated to apply larger penalties.

4.4 RESULTS AND DISCUSSION

The results of the simulation are shown in Figs. 4.2 to 4.3. The time to calculate a new control step for each k-th time step was measured in the simulation and is shown in Table 4.3, where $\bar{x}$ and σ represent the mean and standard deviation of the iteration times of the simulations respectively. The average set-point deviation values were calculated to give an indication of the performance of a controller and is shown in Table 4.4 for the tracking error of each output Y_{JT}, Y_{SVOL} and Y_{PSE}.

The simulation results show that the constrained MPSP and NMPC controllers can reject disturbances with the same efficiency. The $MPSP_{QP}$ and $NMPC_{B3}$ controllers are the least computationally demanding algorithms as shown in Fig. 4.4 and Fig. 4.5. When the sump volume y_{SVOL} is at the upper constraint from $t = 2.3$ h to $t \approx 2.5$ h, the calculation time of the $MPSP_{NLP}$ controller increases slightly. All the controller computational times do not increase when any constraints are active, but the computational time of the $MPSP_{QP}$, $MPSP_{NLP}$ and $NMPC_{B3}$ controllers increase significantly when new disturbances are added to the system as can be seen at $t = 0.5$ h, $t = 1.5$ h and $t = 3$ h. This is not the case for the $NMPC_{B1}$ controller because the computational time stays constant at approximately 7.9 s during the simulation until plant disturbances are removed. The reason for the computational time of the $MPSP_{B1}$ controller staying constant during the simulations is that the controller does not converge to an optimal solution before 15 iterations of the NLP solver.

The large increase in computational time when the first disturbance is introduced at $t = 0.5$ h increases the overall standard deviation of all the controllers. The differences in the computational times between the $MPSP_{QP}$ and the $NMPC_{B3}$ controller are minimal compared to the sampling time. The $MPSP_{QP}$ controller is on average 201 ms faster than the $NMPC_{B3}$ controller as seen in Table 4.3.

The computational time for both the MPSP controllers decreases significantly after all the disturbances are removed at $t = 4$ h. The reason for the computational time decrease is that the MPSP controllers converges faster to the optimal solution for their respective objective functions than the NMPC controllers.

The computational time difference between the two NMPC controllers is significant. The computational time increases by a factor of 5.6 when there is no move-blocking implemented on the NMPC controller. The output results of the $NMPC_{B1}$ and $NMPC_{B3}$ controllers differ because the increased number of control moves adds more weight to the input deviation in the objective function for the same weighting matrices Q_{nmpc} and R_{nmpc}.

Table 4.3. Iteration time results of the MPSP and the NMPC simulations for the grinding mill circuit.

Simulated Controller	$\bar{x}$ [s]	σ [s]
$MPSP_{QP}$	0.670	0.544
$MPSP_{NLP}$	1.313	1.098
$NMPC_{B3}$	0.869	0.581
$NMPC_{B1}$	4.878	3.168

Table 4.4. Average set-point deviation percentage of the MPSP and the NMPC simulations for the grinding mill circuit.

Simulated Controller	$\bar{\sigma}(Y_{JT})$ [%]	$\bar{\sigma}(Y_{SVOL})$ [%]	$\bar{\sigma}(Y_{PSE})$ [%]
$MPSP_{QP}$	5.656	12.278	0.402
$MPSP_{NLP}$	5.605	12.244	0.357
$NMPC_{B3}$	3.821	5.456	0.337
$NMPC_{B1}$	5.570	13.955	1.066

The controller that performs the best when comparing all the average set-point deviation values in Table 4.4 is the $NMPC_{B1}$ controller. Both the *MPSP* controllers achieve similar output results for Y_{PSE} compared to the $NMPC_{B1}$ controller and similar output results for Y_JT and Y_{SVOL} compared to the $NMPC_{B3}$ controller. The controller that performs the worst is the $NMPC_{B3}$ controller, which has an average set-point deviation of 13.955 % for the sump volume Y_{SVOL}.

All the simulated controllers are viable options for controlling a grinding mill as the calculation time stays below the sampling time of $T_s = 10$ s. The $NMPC_{B1}$ controller has the best performance with the slowest average computational time, and the $MPSP_{QP}$ controller has similar performance with respect to Y_{PSE} but with a much faster average computational time.

The input and output differences between the two MPSP controllers are minimal. Thus, the use of the $MPSP_{QP}$ controller is recommended as the same results are obtained but with a faster computational time.

4.5 CHAPTER SUMMARY

Multiple NMPC and MPSP controllers were implemented on a single-stage grinding mill circuit. The simulation showed that the MPSP controller using the QP solver has faster or similar computational time to the NMPC controller which implemented move blocking.

The plant results between an MPSP controller using a quadratic programming problem solver and an MPSP controller using a nonlinear programming problem solver were the same. This can be an indication that for each time step the optimal solution to the controller objective function was found because they achieved the same results using different optimization algorithms.

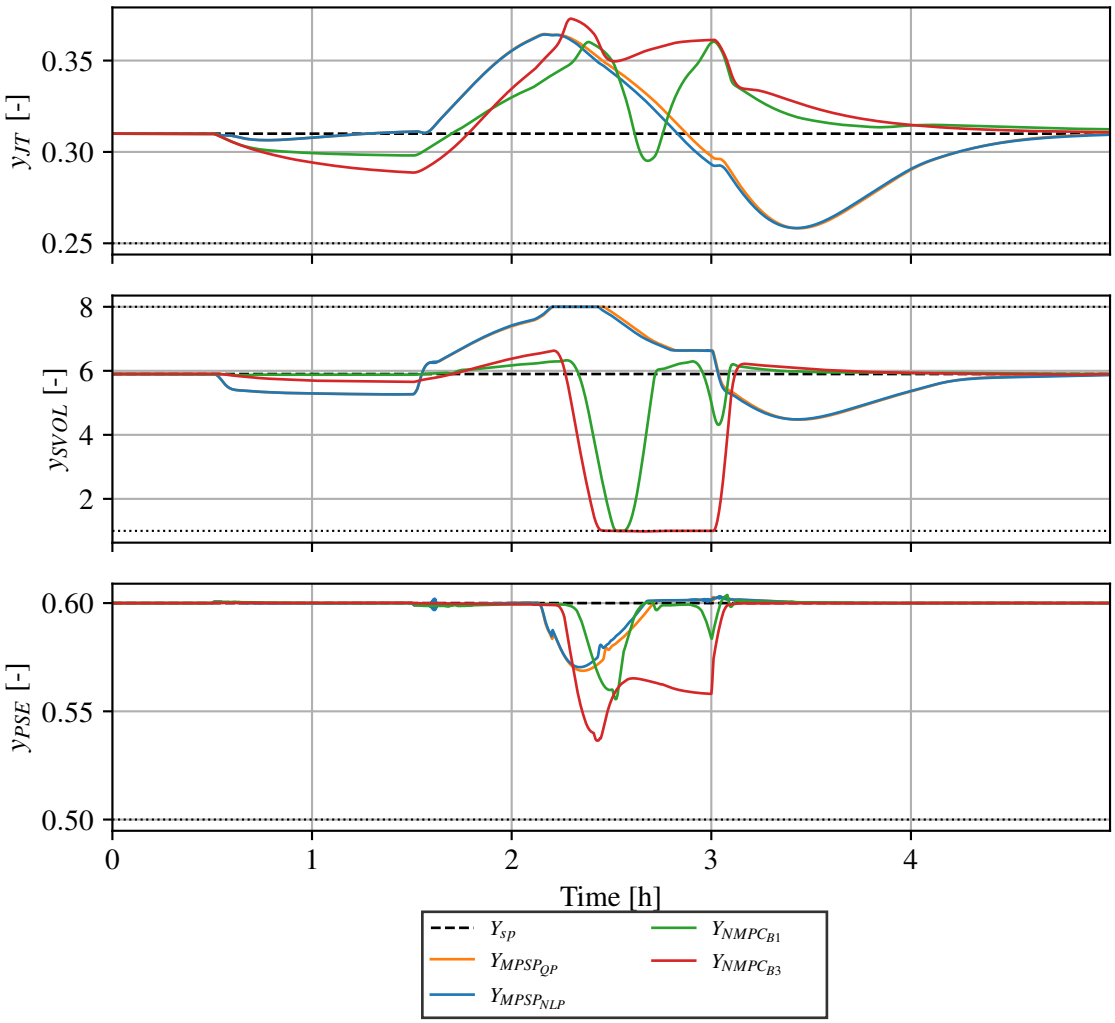

Figure 4.2. MPSP and NMPC simulation outputs of the grinding mill circuit. Y_{sp} is the desired set-point, $Y_{NMPC_{B1}}$ is the output of the NMPC controller with no move-blocking, $Y_{NMPC_{B3}}$ is the output from the NMPC controller with a move-blocking of 3, $Y_{MPSP_{NLP}}$ is the MPSP controller output using the nonlinear optimization routine and $Y_{MPSP_{QP}}$ is the output of the MPSP controller using the quadratic programming solver. The nominal condition of each output is shown in (4.7).

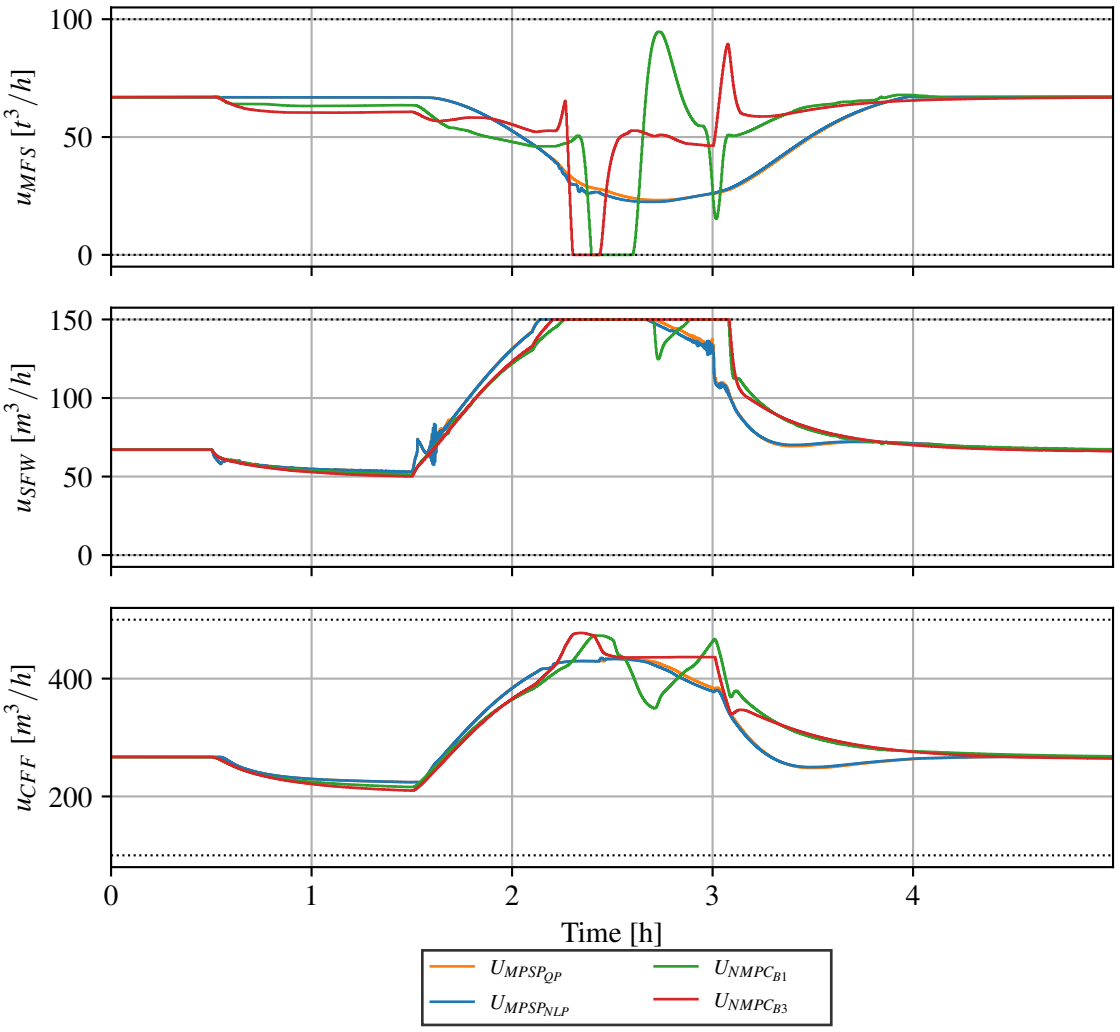

Figure 4.3. MPSP and NMPC simulation inputs to the grinding mill circuit. $U_{NMPC_{B1}}$ is the inputs from the NMPC controller with no move-blocking, $U_{NMPC_{B3}}$ is the inputs of the NMPC controller with a move-blocking of 3, $U_{MPSP_{NLP}}$ is the MPSP controller inputs using the nonlinear optimization routine and $U_{MPSP_{QP}}$ is the inputs of the MPSP controller using the quadratic programming solver. The nominal condition of each input is shown in (4.6).

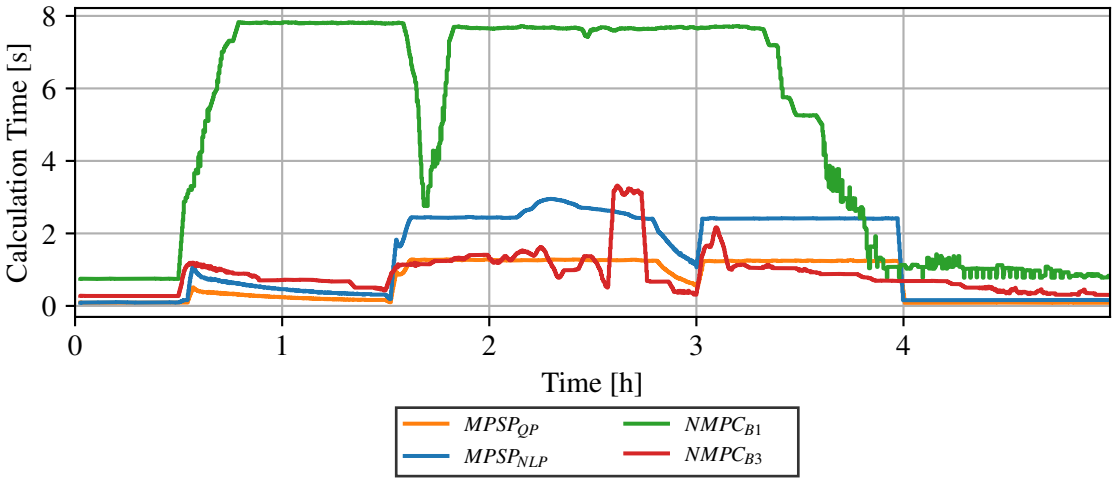

Figure 4.4. The time necessary to calculate a new input for the respective controllers. $NMPC_{B1}$ is the NMPC controller with no move-blocking, $NMPC_{B3}$ is the NMPC controller with a move-blocking of 3, $MPSP_{NLP}$ is the MPSP controller using a nonlinear programming solver and $MPSP_{QP}$ is the MPSP controller using a quadratic programming solver.

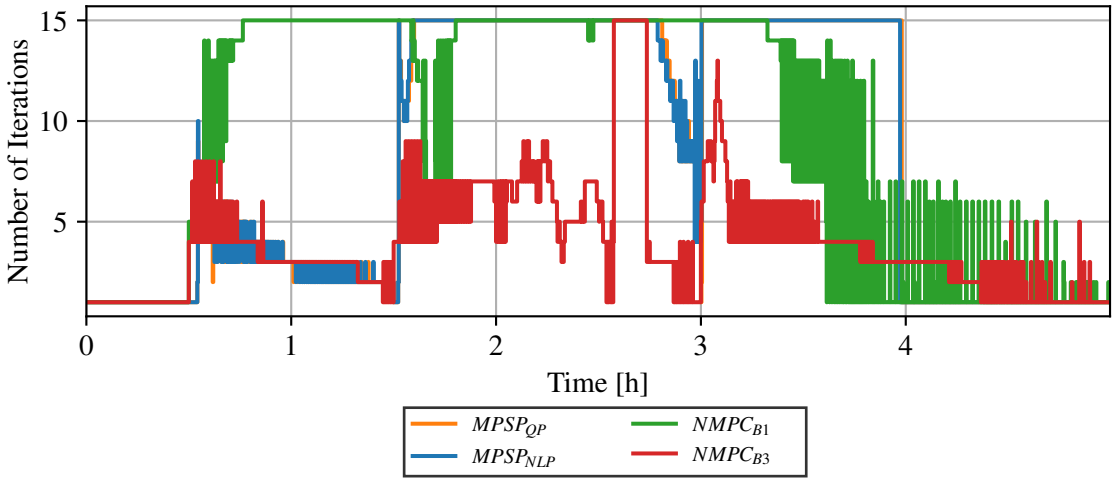

Figure 4.5. The number of iterations used to calculate a new input for the respective controllers. $NMPC_{B1}$ is the NMPC controller with no move-blocking, $NMPC_{B3}$ is the NMPC controller with a move-blocking of 3, $MPSP_{NLP}$ is the MPSP controller using a nonlinear programming solver and $MPSP_{QP}$ is the MPSP controller using a quadratic programming solver.

CHAPTER 5 MPSP APPLIED TO A FLOTATION CIRCUIT

5.1 CHAPTER OVERVIEW

The constrained MPSP controller in Section 2.2 and the constrained NMPC controller in Section 2.1 are applied in simulation to a four-cell flotation circuit. In this chapter, the process description and the chosen nonlinear plant models of each cell and the hopper of the flotation circuit are given. In Section 5.3 the simulation configuration is discussed, which includes the definition of the input and state constraints of the flotation circuit, plant disturbances, and the specific controller configurations. A leader-follower controller strategy is used to create a centralized MPSP or NMPC controller. The best-performing controllers of the simulations in the previous Section 4.3 are used.

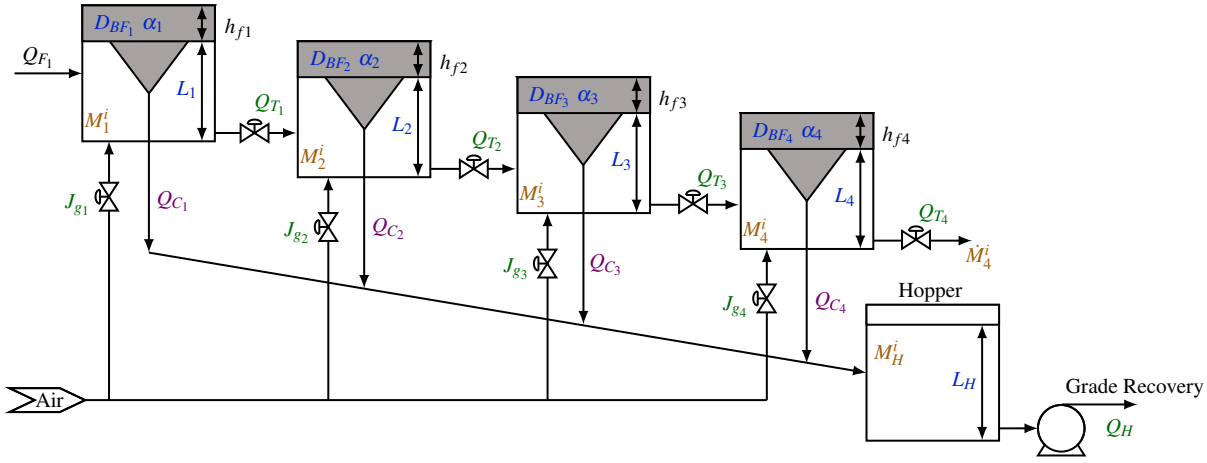

Figure 5.1. A four-cell with a hopper flotation circuit. Adapted from Oosthuizen et al. (2021), with permission.

5.2 FLOTATION PROCESS DESCRIPTION

A four-cell flotation circuit is shown in Fig. 5.1 (Oosthuizen et al., 2021). The variables in Fig. 5.1 are described in Table 5.1. Only a brief overview of the process is given below.

Flotation is a method of separating hydrophobic and hydrophilic particles. Reagents, along with other synthetic compounds, and water are added to the finely crushed metal ores in flotation cells (Mondal et al., 2021). The cells are then agitated with air, which causes the separation of the desired materials from the ores which float to the top of the cell to form a froth. The overflow of the froth is the concentrate that consists of the desired material.

There are four flotation cells where the tailings of each cell is denoted by Q_{T_k} and the concentrate flow is denoted by Q_{C_k}, where $k = 1,2,3,4$ represents the respective flotation cells. The cell tailings feed into the next flotation cell downstream as shown in Fig. 5.1. The concentrate of each cell is collected in the concentrate hopper which is then pumped to further mineral extraction processes. The main inputs to the flotation circuit are the aeration rates Q_{Air_k}, the tailings flow-rate of each cell Q_{T_k} and the concentrate flow-rate out of the hopper Q_H. The level of each cell is denoted as L_k and the level of the hopper is denoted by L_H. The froth depth of each cell is indicated as h_{fk}.

5.2.1 Flotation process model

A brief overview of the model of the flotation circuit in Fig. 5.1 is given below. A complete description is given by Oosthuizen et al. (2021). The model nomenclature is given in Table 5.1. The variable and parameter values were taken from Oosthuizen et al. (2021).

The state-space continuous model for each cell k is

$$\begin{aligned}
\dot{D}_{BF_k} &= \frac{K_{BS_{J_g}} J_{g_k} + K_{BS_\lambda} \lambda_{air_k} - D_{BF_k}}{\lambda_{air_k}} \\
\dot{\alpha}_k &= \frac{K_{\alpha_{BF}} D_{BF_k} + K_{\alpha_{J_g}} J_{g_k} - \alpha_k}{\lambda_{air_k}} \\
\dot{L}_k &= \left(Q_{F_k} - Q_{T_k} - Q_{C_k}\right)/A_k \\
\dot{M}_k^i &= \dot{M}_{F_k}^i - \dot{M}_{T_k}^i - \dot{M}_{C_k}^i,
\end{aligned} \tag{5.1}$$

where D_{BF_k} [mm] is the top froth bubble size, α_k is the air recovery, L_k [m] is the level of the pulp, M_k^i [kg] is the mass of material class i in cell k, and $\dot{M}_{F_k}^i$, $\dot{M}_{T_k}^i$, and $\dot{M}_{C_k}^i$ [kg/h] are the feed, tailings and concentrate mass flow-rates. The variable Q_{F_k} [m^3/h] represents the input flow-rate to the specific cell k and is the tailings of the previous cell $k-1$. The surface area of a cell is denoted by A_k.

Table 5.1. Flotation circuit nomenclature.

Parameter	Value	Description
A_k	8.20	The cross-sectional area of cell k [m^2]
A_H	2.00	The cross-sectional area of the hopper [m^2]
J_{g_k}	–	Superficial gas velocity for cell k [mm/s]
Q_{air_k}	–	Volumetric air flow-rate to cell k [m^3/h]
h_{f_k}	–	Froth depth [mm]
α_k	–	Air recovery for cell k
D_{BF_k}	–	Mean top of froth bubble diameter for cell k [mm]
ρ_s^0	3000	Solid particle density for desired material class $i = 0$ [kg/m^3]
ρ_s^1	1800	Solid particle density for gangue material class $i = 1$ [kg/m^3]
ρ	1000	Fluid density [kg/m^3]
μ	0.001	Fluid viscosity [Pa.s]
g	9.81	Gravitational acceleration [m/s^2]
d_{pmin}	10	Particle minimum diameter [μm]
d_{pmax}	150	Particle maximum diameter [μm]
C_{PB}	50	Plateau border drag coefficient
P_e	0.15	Dispersion Peclet number
$K_{BS_{J_g}}$	0.05	The effect of the superficial gas velocities on the mean top of froth bubble diameter
K_{BS_λ}	0.03	The effect of the average froth residence time on the mean top of froth bubble diameter
$K_{\alpha_{BF}}$	-0.002	The effect of the mean top of froth bubble diameter on the air recovery
$K_{\alpha_{J_{g1}}}$	7.20	The effect of the superficial gas velocities on the air recovery in cell 1
$K_{\alpha_{J_{g2}}}$	7.30	The effect of the superficial gas velocities on the air recovery in cell 2
$K_{\alpha_{J_{g3}}}$	7.00	The effect of the superficial gas velocities on the air recovery in cell 3
$K_{\alpha_{J_{g4}}}$	6.63	The effect of the superficial gas velocities on the air recovery in cell 4
K^0	2.30	The flotation rate-constant for desired material class $i = 0$
K^1	0.0002	The flotation rate-constant for gangue material class $i = 1$

The water recovery model for the flotation circuit is modelled as

$$\frac{Q_{C_k}}{A_k} = \begin{cases} \frac{18.54\mu C_{PB} J_{g_k}^2}{\rho g D_{BF_k}^2}(1-\alpha_k)\alpha_k & 0 < \alpha_k < 0.5 \\ \frac{18.54\mu C_{PB} J_{g_k}^2}{4\rho g D_{BF_k}^2} & \alpha_k \geqslant 0.5, \end{cases} \tag{5.2}$$

where ρ, g, μ and C_{PB} represent the fluid density, gravitational constant, fluid viscosity and the Plateau border drag coefficient respectively.

The superficial gas velocity J_{g_k} is calculated as

$$J_{g_k} = 100\frac{Q_{air_k}}{A_k}. \tag{5.3}$$

The masses for the concentrate are calculated from entrainment and true flotation models. The entrainment model defines the concentrate mass flow-rate as

$$\dot{M}_{C_k}^i = \frac{K^i M_k^i J_{g_k} \alpha_k}{D_{BP_k}} + Ent_{Frac}^i \frac{M_k^i}{A_k L_k} Q_{C_k}, \tag{5.4}$$

where K^i is the flotation rate constant for material class i (Neethling and Cilliers, 2009). The entrainment factor is

$$Ent_{Frac}^i = \frac{\ln\left(d_{ptr}^i\right) - \ln\left(d_{pmin}\right)}{\ln\left(d_{pmax}\right) - \ln\left(d_{pmin}\right)}, \tag{5.5}$$

and d_{pmin} is the minimum particle diameter, and d_{pmax} is the maximum particle diameter. The particle diameter is

$$d_{ptr}^i = \sqrt[3]{\frac{\ln(0.5) J_{g_k{}^2}}{K_{ent}^i h_{f_k}}}, \tag{5.6}$$

and the constant K_{ent}^i is defined as

$$K_{ent}^i = \left[\frac{1}{3}\frac{g\left(\rho_s^i - \rho\right)}{18\mu}\right]^{1.5} \frac{\sqrt{\frac{\rho g}{3\mu C_{PB}}(\sqrt{3} - \pi/2)P_e}}{\sqrt{\alpha_k(1-\alpha_k)}}, \tag{5.7}$$

where ρ_s^i is the solid's particle density of the material class i. The state-space equation for the hopper is

$$\begin{aligned} \dot{L}_H &= \frac{Q_{C_1} + Q_{C_2} + \ldots + Q_{C_N} - Q_H}{A_H} \\ \dot{M}_H^i &= \sum_{k=1}^{N} \dot{M}_{C_k}^i - \frac{M_H^i}{L_H A_H} Q_H, \end{aligned} \tag{5.8}$$

where N is the number of cells from which the concentrate overflows to the hopper, M_H^i is the mass of material class i in the hopper and A_H is the surface area of the hopper. The grade in the hopper is calculated as the ratio between the desired material mass $M_{D_H}^0$ and the total masses inside the concentrate hopper as

$$Grade_H = \frac{M_{D_H}^0}{\sum_{i=0}^{n} M_H^i}, \tag{5.9}$$

where n is the total number of material classes i. For this study, only two mineral classes are used, i.e., $i = 0$ for the desired mineral and $i = 1$ for the gangue.

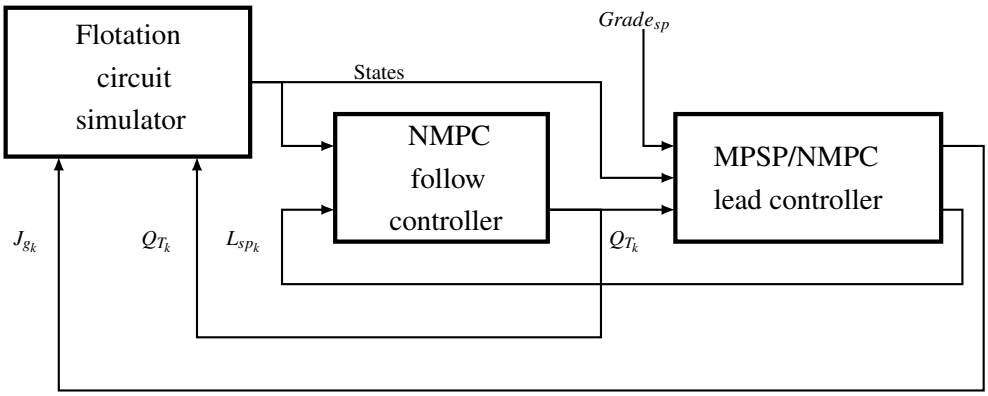

Figure 5.2. Lead-follow controller configuration for the flotation circuit.

5.3 FLOTATION CIRCUIT SIMULATION

To compare the performance of constrained MPSP and NMPC as applied to the flotation circuit, a leader-follower controller is used as shown in Fig. 5.2 (Oosthuizen, 2023). The follower controller is used to control the desired levels of the cells $L_{k_{sp}}$ and the hopper $L_{H_{sp}}$ by manipulating Q_{T_k} and Q_H, whereas the leader controller is used to keep the grade of the flotation circuit at a desired set-point by manipulating the level set-point as well as the superficial gas velocity J_{g_k}. For this configuration, the leader controller does not have any output constraint, but only input and state constraints. In Fig. 5.2 the following controller is implemented using NMPC. The lead controller is used to compare constrained MPSP to NMPC in simulation.

5.3.1 Simulation configuration

To compare the performance of the MPSP controller and the NMPC as applied to the flotation circuit, the following general configuration is used.

The simulation duration is 270 min. Since the residence time for each flotation cell is approximately 60 s, the sampling time for each controller was set at $T_s = 10$ s to ensure the fast dynamics in the froth are captured by the NMPC (Oosthuizen, 2023; Hadler et al., 2010).

The nonlinear state-space description of the flotation circuit in (5.1) is simulated using the Runge-Kutta fourth-order method.

All model states are observable and can be estimated using an appropriate observer (Oosthuizen et al., 2021). Since the observer design falls outside the scope of this study, full-state feedback is

assumed.

The initial state conditions of each cell are

$$\begin{aligned} X_{c1}^0 &= \left[D_{BF_1}, \alpha_1, L_1, M_1^0, M_1^1\right]^T \\ &= [1.80, 0.41, 1.30, 53.87, 7007.32]^T \\ X_{c2}^0 &= \left[D_{BF_2}, \alpha_2, L_2, M_2^0, M_2^1\right]^T \\ &= [1.45, 0.27, 1.30, 29.58, 7024.61]^T \\ X_{c3}^0 &= \left[D_{BF_3}, \alpha_3, L_3, M_3^0, M_3^1\right]^T \\ &= [1.66, 0.17, 1.30, 19.84, 7031.61]^T \\ X_{c4}^0 &= \left[D_{BF_4}, \alpha_4, L_4, M_4^0, M_4^1\right]^T \\ &= [1.40, 0.17, 1.30, 13.44, 7027.26]^T \\ X_H^0 &= \left[L_H, M_H^0, M_H^1\right]^T \\ &= [1.40, 97.62, 7.31]^T. \end{aligned} \tag{5.10}$$

The initial input conditions of the following and leading controllers are

$$\begin{aligned} U_{follow}^0 &= [Q_{T_1},\ Q_{T_2},\ Q_{T_3},\ Q_{T_4},\ Q_H]^T \\ &= [728.38,\ 727.62,\ 727.29,\ 727.01,\ 3.00]^T \\ U_{lead}^0 &= [J_{g1},\ J_{g2},\ J_{g3},\ J_{g4},\ L_1,\ L_2,\ L_3,\ L_4,\ L_H]^T \\ &= [8.4,\ 8.5,\ 8.1,\ 7.7,\ 1.30,\ 1.30,\ 1.30,\ 1.30, 1.00]^T, \end{aligned} \tag{5.11}$$

where the desired grade is $Y_{sp} = Grade_{sp} = 0.33$.

The input ranges for the simulations are

$$\begin{aligned} 4.00 \leq J_{gk} \leq\ 12.00\ \ \forall\, k = 1,2,3,4 \\ 0.01 \leq L_k \leq 1.35\ \ \forall\, k = 1,2,3,4 \\ 0.02 \leq L_H \leq 2.00. \end{aligned} \tag{5.12}$$

The desired set-points are kept constant at the nominal values of the plant.

Measured disturbances are introduced to activate the constraints of the controllers. The input feed flow Q_{F_1} is reduced by 30% of its nominal value in 10 min with a ramp function at $t = 30$ min. The input feed density ρ_{F_1} is reduced by 5% of its nominal value at $t = 60$ min. The input feed grade $G_{F_1}^i$ is

reduced by 5% of its nominal value at $t = 90$ min. The maximum and minimum particle diameter sizes are decreased by 50 % of the initial values at $t = 120$ min. The flotation rate constant K^0, which is the desired material, is increased by 10 % of its nominal value at $t = 150$ min. The flotation rate constant K^1, which is the gangue, is increased by 20 % of its nominal value at $t = 180$ min.

The simulations were done in Python. The simulations were executed on an Intel(R) Core(TM) i5-8400 (6 Core) 2.80 GHz processor with 20 GB RAM running a Microsoft Windows 10 operating system.

5.3.2 NMPC follow controller

The NMPC algorithm is used for the following controller, where the prediction horizon is $N_p = 60$ and the control horizon is $N_c = 20$ with a move blocking of $N_B = 5$ (Wang and Boyd, 2010). Warm-starting is used for the initial guess input of the optimization problem. The weighting matrices for the NMPC follow controller is chosen as

$$\begin{aligned} Q_{nmpc}^{follow} &= \text{diag}([Q_{L_1},\, Q_{L_2},\, Q_{L_3},\, Q_{L_4},\, Q_{L_H}]) \\ R_{nmpc}^{follow} &= \text{diag}([R_{Q_1},\, R_{Q_2},\, R_{Q_3},\, R_{Q_4},\, R_{Q_H}]), \end{aligned} \tag{5.13}$$

where $Q_{L_k} = Q_{L_H} = 1000$ represents the output weighting matrix value and $R_{Q_k} = R_{Q_H} = 0.01$ represents the input weighting matrix value for all cells k and the hopper H.

The NMPC algorithm terminates when the optimization routine has been executed a maximum of 5 times, or the algorithm converged between iterations within a tolerance of 1×10^{-3}.

5.3.3 NMPC lead controller

The settings of the lead NMPC controller are similar to the following NMPC controller in terms of the use of warm starting, $N_p = 60$, $N_c = 20$, and $N_B = 5$.

The weighting matrix for the NMPC controller in (2.1) are chosen as

$$\begin{aligned} Q_{nmpc}^{lead} &= Q_{Grade} = 100 \\ R_{nmpc}^{lead} &= \text{diag}([R_{Jg_1},\, R_{Jg_2},\, R_{Jg_3},\, R_{Jg_4},\, R_{L1_{sp}}, R_{L2_{sp}},\, R_{L3_{sp}},\, R_{L4_{sp}},\, R_{LH_{sp}}]), \end{aligned} \tag{5.14}$$

where $R_{Jg_k} = 0.1$ and $R_{Lk_{sp}} = R_{LH_{sp}} = 1$ for all cells k and the hopper H.

The NMPC algorithm terminates when the optimization routine has been executed a maximum of 10 times, or the algorithm converged between iterations within a tolerance of 1×10^{-4}.

5.3.4 MPSP configuration

The settings of the lead MPSP controller are as follows.

The prediction and control horizon are the same with a value of $N = 60$. The weighting matrices are

$$\begin{aligned} Q_{mpsp} &= Q_{Grade} = 100 \\ R^{lead}_{mpsp} &= \text{diag}([R_{Jg_1},\, R_{Jg_2},\, R_{Jg_3},\, R_{Jg_4},\, R_{L1_{sp}}, R_{L2_{sp}},\, R_{L3_{sp}},\, R_{L4_{sp}},\, R_{LH_{sp}}]), \end{aligned} \tag{5.15}$$

where $R_{Jg_k} = 0.0001$ and $R_{Lk_{sp}} = R_{LH_{sp}} = 0.001$ for all cells k and the hopper H.

The MPSP algorithm terminates for each iterative step if the algorithm has been executed 10 times, or if the condition

$$\frac{\left\| Y^i_k - Y^*_k \right\|_2}{\left\| Y^*_k \right\|_2} < 0.01,$$

is met.

The most computationally efficient MPSP controller in the grinding mill circuit simulation was $MPSP_{QP}$ where the strictly convex programming problem solver was used. The same strictly convex QP solver that was used in the grinding mill circuit simulation is used for the flotation circuit simulation.

5.4 RESULTS AND DISCUSSION

All the simulation results of the NMPC and the MPSP controllers are shown in Figs. 5.3 to 5.7. The grade and recovery of the flotation circuit are shown in Fig. 5.3. The control objective of the lead controller was to keep the flotation circuit grade at a specific set-point. The grade performance of the two controllers are both acceptable with the NMPC controller achieving a final set-point deviation of 1.70 % and the MPSP controller achieving a final set-point deviation of 0.06 %. The recovery shown in Fig. 5.3 is instantaneous and differs from true recovery during transient (non-steady-state) periods. At steady-state, the difference between instantaneous and true recovery is negligible (Oosthuizen, 2023).

Table 5.2. Iteration time and performance results of the MPSP and the NMPC simulations for the flotation circuit.

Simulated Controller	$\bar{x}$ [s]	σ [s]	Maximum [s]	$\bar{\sigma}(Grade_H)$ [%]
NMPC	8.029	1.419	13.876	0.2254
MPSP	0.907	0.773	3.060	0.0815

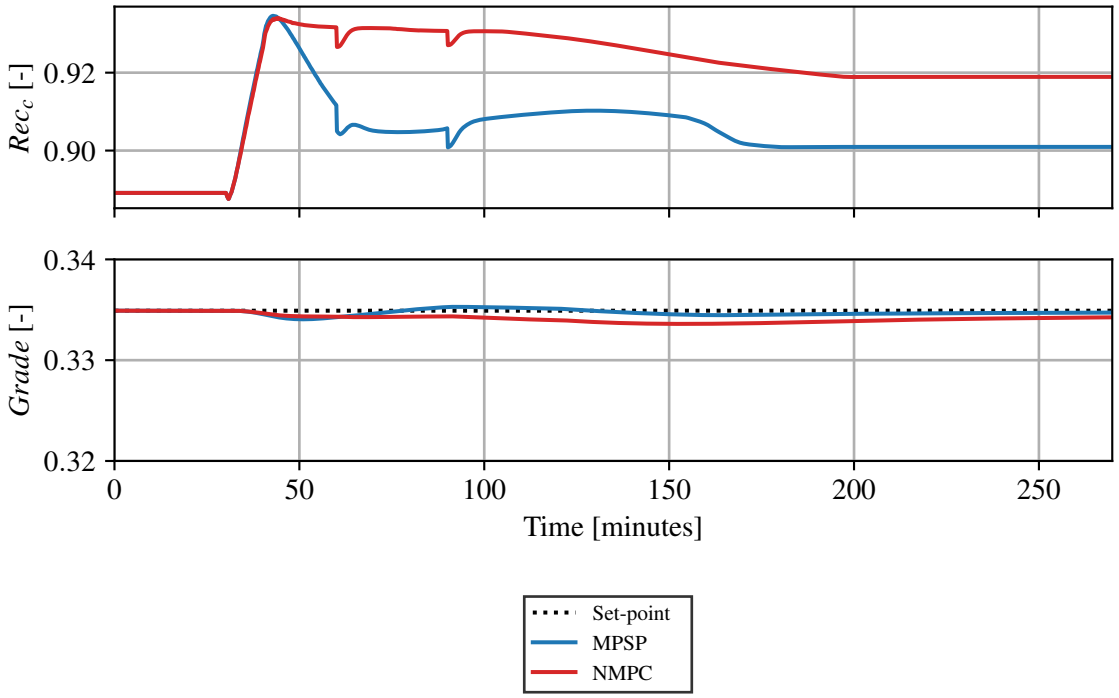

Figure 5.3. Grade and recovery of the flotation circuit.

NMPC settles at a higher recovery than MPSP. The cell level results are shown in Fig. 5.4, where the dotted lines indicate the set-points given from the leading controller to the following level controller, and the solid lines show the output results of the following controller.

The leading NMPC controller manipulates the level set-points more aggressively than the MPSP controller for the first disturbance rejection at $t = 30$ min. Fig. 5.4 in conjunction with Fig. 5.5 shows that the NMPC controller uses the level set-point to reject the first plant disturbances and then uses the superficial gas velocity J_{g_k} to reject the disturbances that occur after $t = 100$ min.

The opposite is true for the MPSP controller. The MPSP controller uses the superficial gas velocities J_{g_k} along with the cell levels L_k to reject all the disturbances. This causes the MPSP controller to have better state constraint performance than the NMPC controller.

Fig. 5.6 shows the froth bubble diameter size D_{BF_k} of each cell k. Both the MPSP and the NMPC controllers can reject the state constraints of the froth bubble size during the first disturbance, but the NMPC controller violates the constraints at $t \approx 40$ min. The possible reason for this state constraint

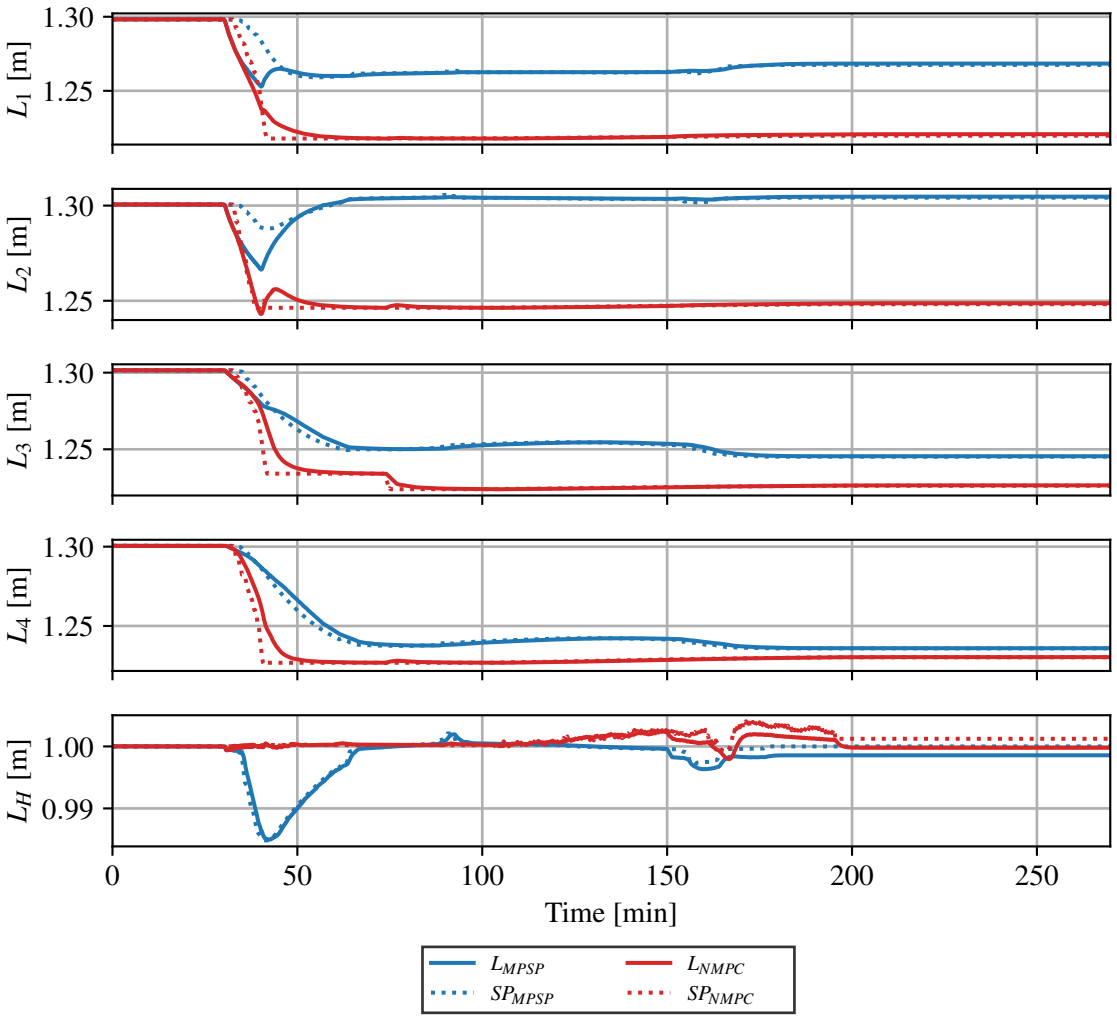

Figure 5.4. Flotation cell levels. L_{NMPC} is the output level for the NMPC controller, L_{MPSP} is the output level of the MPSP controller, SP_{NMPC} is the input set-point of the NMPC controller to the following controller and SP_{MPSP} is the input set-point of the MPSP controller to the following controller.

violation is caused by the input disturbance between the following controller and the NMPC controller on the cell levels L_1, L_3 and L_4.

Both the controllers were able to reject the measured disturbances and control the plant grade to the desired set-point.

The iteration time information and the average set-point deviation results for the lead controllers are shown in Fig. 5.7 and Table 5.2. The MPSP controller is on average 8.85 times faster than the NMPC

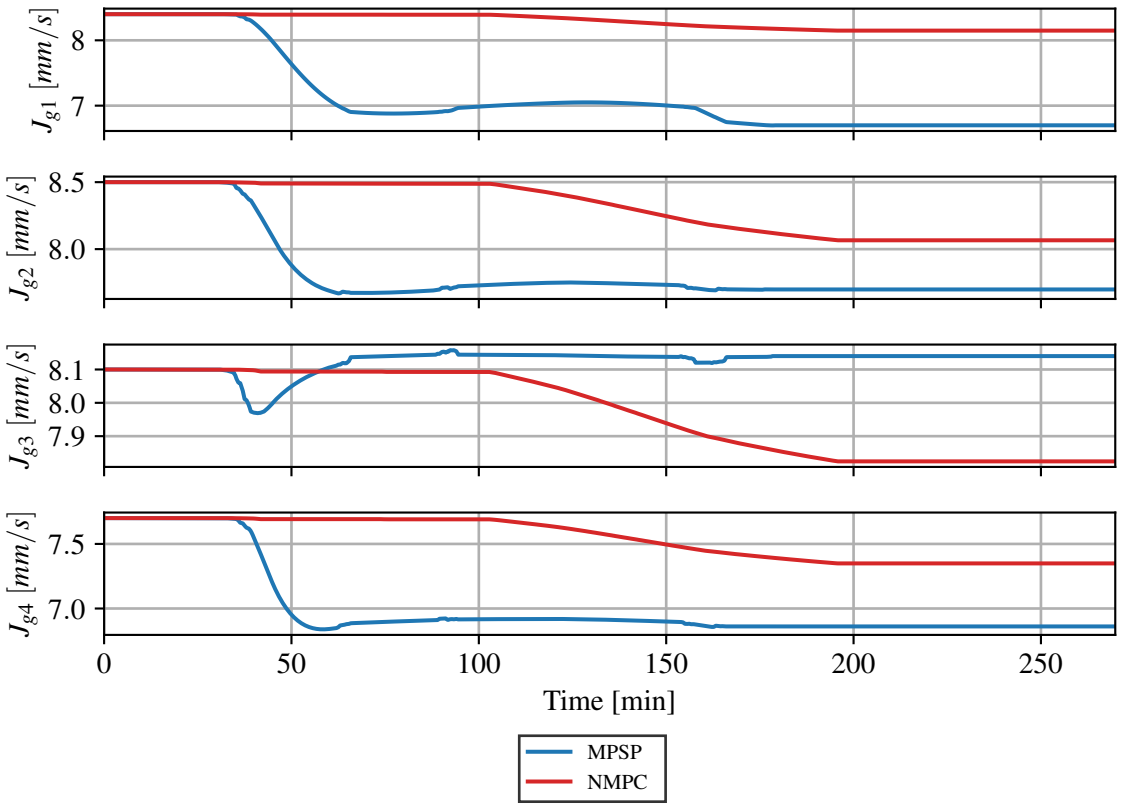

Figure 5.5. Superficial gas velocity.

controller. The standard deviation of the calculation time is smaller for the MPSP controller compared to the NMPC controller. The MPSP controller calculation time significantly increases after the first disturbance is introduced at $t \approx 35$ min. The maximum calculation time of the MPSP controller is 3.06 s, which is when the initial disturbance is introduced to the plant.

Both the lead *NMPC* and *MPSP* controllers were able to control the hopper grade $Grade_H$ within an average set-point deviation of 0.3 %. The *MPSP* controller performs better than the *NMPC* controller when comparing the grade of the flotation system. The average set-point deviation performance of the *MPSP* controller is 0.144 % smaller than the performance of the *NMPC* controller.

The sampling time for each controller is $T_s = 10$ s. As shown in Fig. 5.7, the NMPC controller is able to calculate a new input within 7.5 s on average, but there are instances where it requires more than 10 s to calculate a new input. The MPSP controller is significantly faster and requires at most 3 s to calculate a new input.

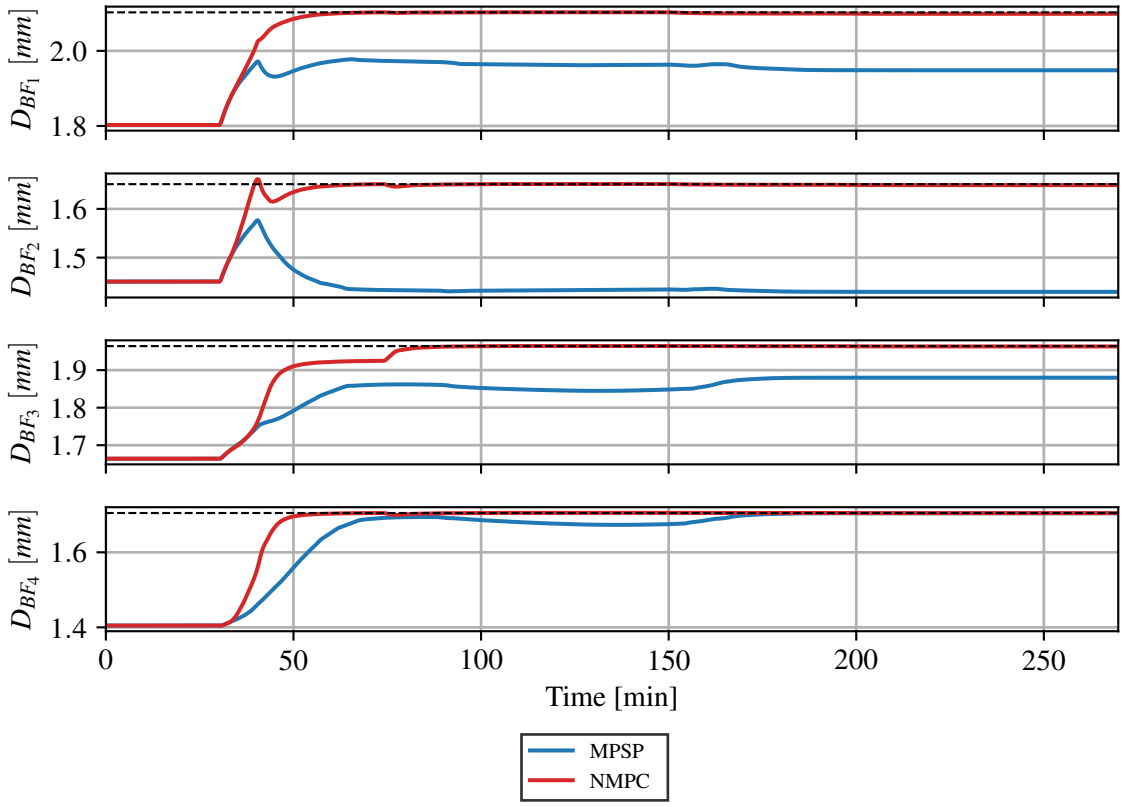

Figure 5.6. Froth bubble size diameter of each cell.

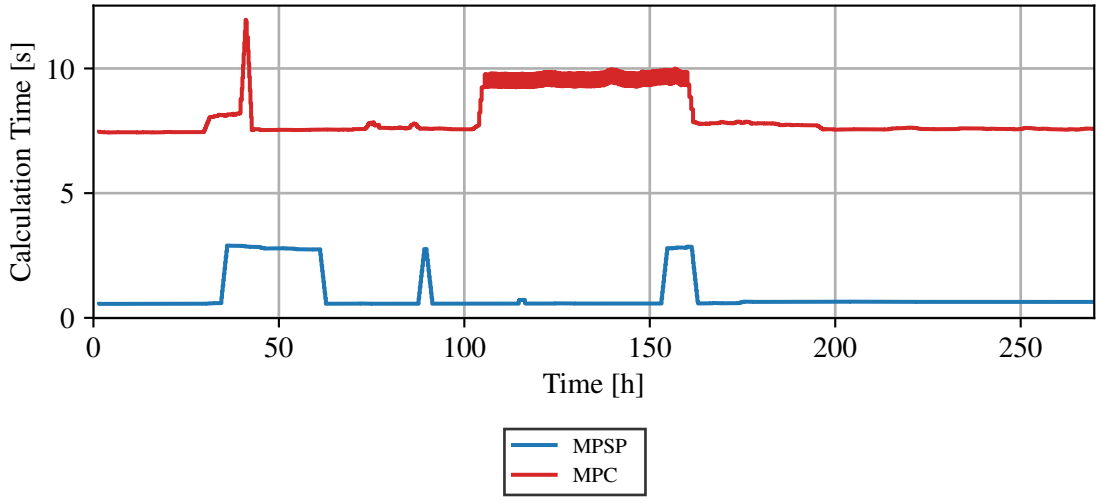

Figure 5.7. The time necessary to calculate a new input of the controllers for the flotation circuit.

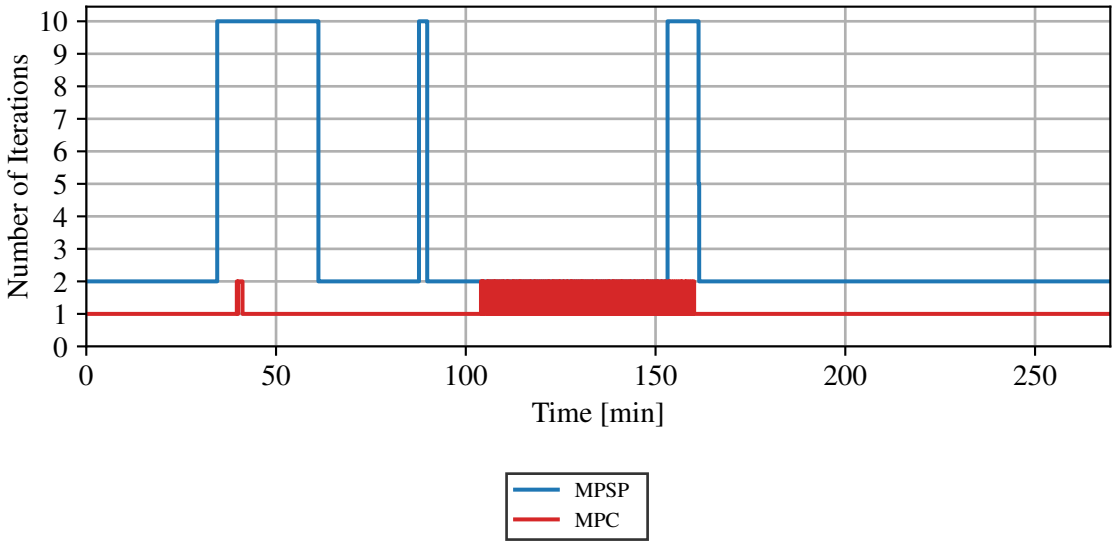

Figure 5.8. The number of iteration steps used to calculate a new input of the controllers for the flotation circuit.

The number of iterations for each time step of a controller is shown in Fig. 5.8. There is a direct correlation between the calculation time and the number of iterations. The NMPC controller uses only one or two optimization iteration for each time step and the computational time thereof is significantly higher than the MPSP controller which has used the maximum of 10 iterations. Fig. 5.7 and Fig. 5.8 together indicate that the MPSP algorithm is more computationally efficient.

Industrial flotation circuits often run at a sampling rate of $T_s = 60\,\text{s}$, even though this may be too slow to capture all the dynamics of the process (Hadler et al., 2010; Oosthuizen, 2023). At this sampling rate, both NMPC and MPSP are viable options where only a single flotation bank is controlled. In the case where multiple flotation banks are combined into a plant-wide MPC strategy, MPSP is the better option. Investigation of MPSP for plant-wide control of a mineral processing plant remains open for further research.

5.5 CHAPTER SUMMARY

Both an MPSP and an NMPC centralized controller were implemented in simulation to a four-cell flotation circuit.

The MPSP controller had a faster computational time throughout the simulation and had better set-point tracking. The two controllers showed very different input, state and output results. The differences

might be caused by the high dimensionality of the plant which contributes to multiple suboptimal solutions in the objective function of each controller.

CHAPTER 6 CONCLUSION

The MPSP controller is a viable replacement for the NMPC controller in the simulated mineral processing circuits. The MPSP controller obtains similar desired results to that of an NMPC controller with the same or faster computational times.

The scalability of the MPSP method could be a big motivation for the use thereof in mineral processes. In the grinding mill simulation, the MPSP objective function results in a strictly convex quadratic programming problem with 180 optimization variables. The computational time for the MPSP method, while disturbances were introduced, was approximately 1 s. In the case of the flotation circuit, the number of optimization variables increased to 540. The average computational time for the flotation circuit was 0.6 s and increased to 2.5 s during disturbances.

In the case of the NMPC method, the scalability does not compare to that of the MPSP method. From the grinding mill circuit to the flotation circuit simulations, the NMPC optimization variables increased from 12 to 36. This led to an increase in the computational time from approximately 2 s to 7.5 s. (The increase in the computational time is not necessarily caused by the increase in optimization variables, but rather by a larger or more complex objective function with multiple suboptimal solutions.)

The main problem with the MPSP method is when the initial conditions of the objective function do not satisfy the plant constraints which leads to an infeasible programming problem. With most MPC and NMPC controllers infeasibility is not a problem, because multiple penalty functions for the constraints can be included in the objective function. This is known as a soft constraint method. The soft constraint method is not as simple to implement on the MPSP controller used in this study, compared to an NMPC controller, because the penalty functions of the constraints will change the

strictly convex QP problem into a discontinuous or NLP optimization problem which leads to larger computational times.

In Kumar et al. (2019) the initial state of the robot did not satisfy the state constraints. They solved the initial infeasibility problem by only applying the constraints after the set-point trajectory was reached.

The use of a slack variable approach might be an ideal way of including state constraints without having infeasibility problem definitions. With the use of a slack variable approach, the number of model differential equations increases (Bhitre and Padhi, 2014). A comparison between the slack variable approach and the constrained MPSP control used in this paper can be done as possible future work.

The MPSP method is a viable model predictive control algorithm which can be used in cases where the computational time of an NMPC controller is too slow. The MPSP method can handle non-linearities, plant constraints and unmeasured disturbances.

REFERENCES

Alhajeri, M. and Soroush, M. (2020). Tuning guidelines for model-predictive control, *Ind. Eng. Chem. Res.* **59**(10): 4177–4191.

Allgower, F., Findeisen, R. and Nagy, Z. K. (2004). Nonlinear model predictive control: From theory to application, *J. Chin. Inst. Chem. Eng.* **35**(3): 299–316.

Bemporad, A., Morari, M., Dua, V. and Pistikopoulos, E. N. (2002). The explicit linear quadratic regulator for constrained systems, *Automatica* **38**(1): 3–20.

Bhitre, N. G. and Padhi, R. (2014). State constrained model predictive static programming: A slack variable approach, *IFAC Proceedings Volumes* **47**(1): 832–839.

Bin, F., Hang, G., Kang, C., Xingyu, W. and Jie, Y. (2018). Aero-thermal heating constrained midcourse guidance using state-constrained model predictive static programming method, *J Syst. Eng. Electron.* **29**(6): 1263.

Biswas, D., Ghosh, S., Sengupta, S. and Mukhopadhyay, S. (2022). Energy management of a parallel hybrid electric vehicle using model predictive static programming, *Energy* **250**: 123505.

Cagienard, R., Grieder, P., Kerrigan, E. C. and Morari, M. (2007). Move blocking strategies in receding horizon control, *J. Process Contr.* **17**(6): 563–570.

Cannon, M. (2004). Efficient nonlinear model predictive control algorithms, *Annu. Rev. Control* **28**(2): 229–237.

Chen, S., Saulnier, K., Atanasov, N., Lee, D. D., Kumar, V., Pappas, G. J. and Morari, M. (2018). Approximating explicit model predictive control using constrained neural networks, *2018 Annual American Control Conference (ACC)*, pp. 1520–1527.

Cisneros, P. S., Voss, S. and Werner, H. (2016). Efficient nonlinear model predictive control via quasi-LPV representation, *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 3216–3221.

Coetzee, L. C., Craig, I. K. and Kerrigan, E. C. (2010). Robust nonlinear model predictive control of a run-of-mine ore milling circuit, *IEEE Trans. Control Syst. Technol.* **18**(1): 222–229.

Diehl, M., Bock, H. G. and Schlöder, J. P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control, *SIAM J. Control Optim.* **43**(5): 1714–1736.

Faroni, M., Beschi, M., Berenguel, M. and Visioli, A. (2017). Fast MPC with staircase parametrization of the inputs: Continuous input blocking, *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8.

Goldfarb, D. and Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratric programs, *Math. Program.* **27**: 1–33.

Gros, S., Zanon, M., Quirynen, R., Bemporad, A. and Diehl, M. (2020). From linear to nonlinear MPC: bridging the gap via the real-time iteration, *Int. J. Control* **93**(1): 62–80.

Hadler, K., Smith, C. and Cilliers, J. (2010). Flotation performance improvement by air recovery optimisation on roughers and scavengers, *XXV Australas. I Min. Met., IMPC 2010*, Vol. 3, pp. 1917–1924.

Hong, H., Maity, A., Holzapfel, F. and Tang, S. (2019). Model predictive convex programming for constrained vehicle guidance, *IEEE Trans. Aerosp. Electron. Syst.* **55**(5): 2487–2500.

Hou, G., Gong, L., Huang, C. and Zhang, J. (2020). Fuzzy modeling and fast model predictive control of gas turbine system, *Energy* **200**: 117465.

Kouzoupis, D., Quirynen, R., Frasch, J. V. and Diehl, M. (2015). Block condensing for fast nonlinear MPC with the dual Newton strategy, *IFAC-PapersOnLine* **48**(23): 26–31.

Kraft, D. (1988). *A software package for sequential quadratic programming*, DFVLR-FB 88-28.

Kumar, P., Anoohya, B. B. and Padhi, R. (2019). Model predictive static programming for optimal command tracking: A fast model predictive control paradigm, *J. Dyn. Syst. Meas. Control.* **141**: 021014.

Kumar, P. and Padhi, R. (2014). Extension of model predictive static programming for reference command tracking, *IFAC Proceedings Volumes* **47**(1): 855–861.

Kunz, K., Huck, S. M. and Summers, T. H. (2013). Fast model predictive control of miniature helicopters, *2013 European Control Conference (ECC)*, pp. 1377–1382.

Le Roux, J. D., Craig, I. K., Hulbert, D. G. and Hinde, A. (2013). Analysis and validation of a run-of-mine ore grinding mill circuit model for process control, *Miner. Eng.* **43–44**: 121–134.

Le Roux, J. D., Olivier, L. E., Naidoo, M. A., Padhi, R. and Craig, I. K. (2016). Throughput and product quality control for a grinding mill circuit using non-linear MPC, *J. Process Contr.* **42**: 35–50.

Le Roux, J. D., Padhi, R. and Craig, I. K. (2014). Optimal control of grinding mill circuit using model predictive static programming: A new nonlinear MPC paradigm, *J. Process Contr.* **24**(12): 29–40.

Le Roux, J. D., Steinboeck, A., Kugi, A. and Craig, I. K. (2017). An EKF observer to estimate semi-autogenous grinding mill hold-ups, *J. Process Contr.* **51**: 27–41.

Le Roux, J. D. and Steyn, C. W. (2022). Validation of a dynamic nonlinear grinding circuit model for process control, *Miner. Eng.* **187**: 107780.

Li, Y., Zhou, H. and Chen, W. (2019). Three-dimensional impact time and angle control guidance based on MPSP, *Int. J. Aerospace Eng.* **2019**: 1–16.

Mayne, D. Q. (2014). Model predictive control: Recent developments and future promise, *Automatica* **50**(12): 2967–2986.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, S., Saboo, A., Fernando, I., Kulal, S., Cimrman, R. and Scopatz, A. (2017). Sympy: symbolic computing in python, *PeerJ Computer Science* **3**: e103.

Mondal, S., Acharjee, A., Mandal, U. and Saha, B. (2021). Froth flotation process and its application, *Vietnam J. Chem* **59**: 417–425.

Neethling, S. and Cilliers, J. (2009). The entrainment factor in froth flotation: Model for particle size and other operating parameter effects, *Int. J. Miner. Process* **93**(2): 141–148.

Nian, R., Liu, J. and Huang, B. (2020). A review on reinforcement learning: Introduction and applications in industrial process control, *Comput. Chem. Eng.* **139**: 1–30.

Nise, N. S. (2020). *Control Systems Engineering*, John Wiley & Sons.

Nocedal, J. and Wright, S. J. (2006). *Numerical optimization*, 2nd edn, Springer.

Oosthuizen, D. J. (2023). *A dynamic flotation model for online estimation, control and optimisation*, PhD thesis, University of Pretoria.

Oosthuizen, D. J., Le Roux, J. D. and Craig, I. K. (2021). A dynamic flotation model to infer process characteristics from online measurements, *Miner. Eng.* **167**: 106878.

Padhi, R. (2008). Model predictive static programming: A promising technique for optimal missile guidance, *Annals of the Indian National Academy of Engineering* **5**: 185–194.

Padhi, R. and Kothari, M. (2009). Model predictive static programming: A computationally efficient technique for suboptimal control design, *Int. J. Innov. Comput. Inf. Control* **5**(2): 399–411.

Pistikopoulos, E. N. (2009). Perspectives in multiparametric programming and explicit model predictive control, *AICHE J.* **55**(8): 1918–1925.

Schwenzer, M., Ay, M., Bergs, T. and Abel, D. (2021). Review on model predictive control: An engineering perspective, *Int. J. Adv. Manuf. Technol.* **117**(5-6): 1327–1349.

Tripathi, A. K. and Padhi, R. (2016). Autonomous landing for UAVs using T-MPSP guidance and dynamic inversion autopilot, *IFAC-PapersOnLine* **49**(1): 18–23.

Wang, Y. and Boyd, S. (2010). Fast model predictive control using online optimization, *IEEE T. Contr. Syst. T.* **18**(2): 267–278.

Wang, Y., Hong, H. and Tang, S. (2019). Geometric control with model predictive static programming on SO(3), *ACTA Astronaut.* **159**: 471–479.

Wolf, I. J. and Marquardt, W. (2016). Fast NMPC schemes for regulatory and economic NMPC – a review, *J. Process Contr.* **44**: 162–183.

Wright, S. J. (1996). Applying new optimization algorithms to more predictive control, *Technical report*, Argonne National Lab.(ANL), Argonne, IL (United States).

Zavala, V. M. and Biegler, L. T. (2009). The advanced-step NMPC controller: Optimality, stability and robustness, *Automatica* **45**(1): 86–93.

Zhang, B., Tang, S. and Pan, B. (2016). Multi-constrained suboptimal powered descent guidance for lunar pinpoint soft landing, *Aerosp. Sci. Technol.* **48**: 203–213.

Zhang, X., Bujarbaruah, M. and Borrelli, F. (2021). Near-optimal rapid MPC using neural networks: A primal-dual policy learning framework, *IEEE T. Contr. Syst. T.* **29**(5): 2102–2114.