Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 55-21 (2022) 49–54

# Controlling a Grinding Mill Circuit using Constrained Model Predictive Static Programming ⋆

Zander M. Noome * Johan D. le Roux *,1

* *Department of Electrical, Electronic, and Computer Engineering, University of Pretoria, Pretoria, South Africa.*

**Abstract:** A constrained Model Predictive Static Programming (MPSP) method is implemented in simulation to a single-stage grinding mill circuit model. The results are compared to a constrained Nonlinear Model Predictive Control (NMPC) method. Both the constrained MPSP and NMPC controllers were able to track the desired output set-points without exceeding any constraints. The comparison shows that the constrained MPSP has a faster computational time than that of the NMPC controller with similar performance. Therefore, constrained MPSP shows promise as a model-based controller for large processes where computational time limits the use of NMPC.

*Keywords:* Computational time, Model Predictive Static Programming (MPSP), Nonlinear Model Predictive Control (NMPC), grinding mill, industrial processes.

## 1. INTRODUCTION

The use of process control on industrial plants is an efficient way to maintain consistent product quality, improve throughput, optimise power usage, and ensure safe process operation. Most industrial processes are complex and have multi-variable inputs and outputs, which make them very difficult to control. Nonlinear Model Predictive Control (NMPC) can be used to control multi-variable problems in the process industry (Bemporad et al., 2002). These NMPCs are ideal for slow processes because they use online computation and can take several minutes to calculate (Bemporad et al., 2002). For example, a robust NMPC was implemented in simulation on a grinding mill circuit in Coetzee et al. (2010), but the computation time was too long for practical implementation. To produce a practically viable controller, it is necessary to reduce the computational time of the model predictive controller (MPC) without compromising the performance.

Different techniques are available to improve the computational time of NMPC. These include Fast NMPC by Wang and Boyd (2010), and Explicit NMPC by 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). The Fast MPC method is done by exploiting the quadratic program that is used to obtain a new control input. Some of these methods include move-blocking and warm-starting (Wang and Boyd, 2010).

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). 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 NMPC (le Roux et al., 2014).

Recently, Kumar et al. (2019) adapted the MPSP method to include state and input constraints. The contribution of this article is the application of the constrained MPSP of Kumar et al. (2019) in simulation to a grinding mill circuit and comparing it to various constrained NMPC controllers.

## 2. MODEL PREDICTIVE STATIC PROGRAMMING

A nonlinear system is written in discrete form as,

$$\begin{aligned} X^i_{k+1} &= F_k\left(X^i_k, U^i_k\right) \\ Y^i_k &= h_k\left(X^i_k\right), \end{aligned} \tag{1}$$

where $X_k \,\epsilon\, \Re^n$, $U_k \,\epsilon\, \Re^m$ and $Y_k \,\epsilon\, \Re^p$ represents the states, inputs and output of the system respectively (Kumar and Padhi, 2014). The subscript $k$ represents the time step and the superscript $i$ represents the iteration index. The aim of the MPSP method is to calculate a control history $U^{i+1}_k$, $k = 1, 2, ..., N$, so that the output $Y^{i+1}_k$ 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 history for the inputs $U_k$ are poor.

⋆ This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Numbers: 137769).

[1] Corresponding author: derik.leroux@up.ac.za

10.1016/j.ifacol.2022.09.242

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

$$\begin{aligned} X_k^{i+1} &\triangleq X_k^i + \Delta X_k^i \\ U_k^{i+1} &\triangleq U_k^i + \Delta U_k^i \\ Y_k^{i+1} &\triangleq Y_k^i + \Delta Y_k^i. \end{aligned} \quad (2)$$

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,

$$\begin{aligned} Y_k^{i+1} &= h(X_k^{i+1}) = h(X_k^i + \Delta X_k^i) \\ &\approx Y_k^i + \left[\frac{\partial Y_k}{\partial X_k}\right]_{X_k^i} \Delta X_k^i. \end{aligned} \quad (3)$$

Substituting $Y_k^{i+1}$ from (2) into (3) and making $\Delta Y_k^i$ the subject, the following equation is obtained,

$$\Delta Y_k^i = Y_k^{i+1} - Y_k^i \approx \left[\frac{\partial Y_k}{\partial X_k}\right]_{X_k^i} \Delta X_k^i. \quad (4)$$

Performing the small error approximation to $X_{k+1}^{i+1}$ while using the states and inputs defined in (2), then,

$$\begin{aligned} X_{k+1}^{i+1} &= F_k\left(X_k^{i+1}, U_k^{i+1}\right) \\ &= F_k\left(X_k^i + \Delta X_k^i, U_k^i + \Delta U_k^i\right) \\ &\approx X_{k+1}^i + \left[\frac{\partial F_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} \Delta X_k^i \\ &\quad + \left[\frac{\partial F_k}{\partial U_k}\right]_{(X_k^i, U_k^i)} \Delta U_k^i \\ \Delta X_{k+1}^i &\approx \left[\frac{\partial F_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} \Delta X_k^i + \left[\frac{\partial F_k}{\partial U_k}\right]_{(X_k^i, U_k^i)} \Delta U_k^i. \end{aligned} \quad (5)$$

Assuming that the states, inputs and outputs all have small deviations, (4) and (5) can be rewritten as,

$$\begin{aligned} dY_k^i &= \left[\frac{\partial Y_k}{\partial X_k}\right]_{X_k^i} dX_k^i \\ dX_{k+1}^i &= \left[\frac{\partial F_k}{\partial X_k}\right]_{(X_k^i, U_k^i)} dX_k^i + \left[\frac{\partial F_k}{\partial U_k}\right]_{(X_k^i, U_k^i)} dU_k^i, \end{aligned} \quad (6)$$

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

$$dY_k^i = \left[A^k\right]^i dX_1^i + \left[B_1^k\right]^i dU_1^i + \cdots + \left[B_{k_1}^k\right]^i dU_{k-1}^i \quad (7)$$

where $\left[A^k\right]^i$ is defined as,

$$\begin{aligned} \left[A^k\right]^i = \left[\frac{\partial Y_k}{\partial X_k}\right]_{(X_k^i)} & \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]_{(X_{k-1}^i, U_{k-1}^i)} \\ & \cdots \times \left[\frac{\partial F_1}{\partial X_1}\right]_{(X_1^i, U_1^i)} \end{aligned} \quad (8)$$

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

$$\begin{aligned} [B_j^k]^i = \left[\frac{\partial Y_k}{\partial X_k}\right]_{(X_k^i)} & \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]_{(X_{k-1}^i, U_{k-1}^i)} \\ \cdots \times \left[\frac{\partial F_{j+1}}{\partial X_{j+1}}\right]_{(X_{j+1}^i, U_{j+1}^i)} & \left[\frac{\partial F_j}{\partial X_j}\right]_{(X_j^i, U_j^i)}. \end{aligned} \quad (9)$$

The state error at the first time step $dX_1$ in (7) is zero because the states are known at that time. This means that the output error can reduce to,

$$dY_k^i = \sum_{j=1}^{k-1} \left[B_j^k\right]^i dU_j^i. \quad (10)$$

While deriving (10) it is clear that the output error 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 (10) 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 $[B_j^k]^i$ for all $k = 2, 3, ..., N$ where $N$ represents the control and prediction horizon, can be computationally heavy. The following recursive algorithm can be used to reduce the computational cost,

$$\begin{aligned} [A_k^k]^i &= I_{n \times n} \\ [A_j^k]^i &= [A_{j+1}^k]^i \left[\frac{dF_j}{dX_j}\right]_{(X_j^i, U_j^i)} \\ [B_j^k]^i &= \left[\frac{dY_k}{dX_k}\right]_{(X_k^i)} [A_{j+1}^k]^i \left[\frac{dF_j}{dU_j}\right]_{(X_j^i, U_j^i)} \end{aligned} \quad (11)$$

where $j = (k-1), (k-2), ..., 1$.

### 2.1 Cost function

In (10) 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,

$$\begin{aligned} J^i = \frac{1}{2} \sum_{k=2}^{N} & \left(\Delta Y_k^i - \Delta Y_k^*\right)^T Q_{mpsp} \left(\Delta Y_k^i - \Delta Y_k^*\right) \\ & + \frac{1}{2} \sum_{k=1}^{N-1} \left(\Delta U_k^i\right)^T R_{mpsp} \left(\Delta U_k^i\right) \end{aligned} \quad (12)$$

where $\Delta Y_k^* = Y_k^* - Y_k^i$ is the output error with respect to the desired output $Y_k^*$, $Q_{mpsp}$ is the output weighting matrix and $R_{mpsp}$ is the input deviation weighting matrix. Minimizing the cost function in (12) 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, (12) can be written as,

$$\begin{aligned} J^i = \frac{1}{2} \sum_{k=2}^{N} & \left(dY_k^i - \Delta Y_k^*\right)^T Q_{mpsp} \left(dY_k^i - \Delta Y_k^*\right) \\ & + \frac{1}{2} \sum_{k=1}^{N-1} \left(dU_k^i\right)^T R_{mpsp} \left(dU_k^i\right). \end{aligned} \quad (13)$$

### 2.2 Constrained MPSP

Constrained MPSP, where constraints are added to the system, uses a compact form of the cost function in (12) by substituting (10) and simplifying to get,

$$
\begin{aligned}
J^i = \frac{1}{2}(\delta U^i)^T \left(R_{mpsp} + ([B]^i)^T Q_{mpsp}[B]^i\right) \delta U^i \\
- (\delta U^i)^T \left([B]^i\right)^T Q_{mpsp}(\Delta Y^{*i}) \\
+ \frac{1}{2}(\Delta Y^{*i})^T Q_{mpsp} \Delta Y^{*i}
\end{aligned} \tag{14}
$$

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

$$
\begin{aligned}
Q_{mpsp} &\triangleq \mathrm{diag}([Q_1], [Q_2],\ ...,\ [Q_N]) \\
R_{mpsp} &\triangleq \mathrm{diag}([R_1], [R_2],\ ...,\ [R_N]) \\
\Delta Y^{*i} &\triangleq \left[(\Delta Y_1^{*i})^T \ \ (\Delta Y_2^{*i})^T \ \ \cdots (\Delta Y_N^{*i})^T\right] \\
[B]^i &\triangleq \begin{bmatrix} [B_1^1]^i & [B_2^1]^i & \dots & [B_N^1]^i \\ [B_1^2]^i & [B_2^2]^i & \dots & [B_N^2]^i \\ \vdots & \vdots & \ddots & \vdots \\ [B_1^N]^i & [B_2^N]^i & \dots & [B_N^N]^i \end{bmatrix}.
\end{aligned}
$$

The state and input constraints applied in (14) are,

$$
\begin{bmatrix} [A]^i \\ -[A^i] \\ I \\ -I \end{bmatrix} \delta U^i \leq \begin{bmatrix} X^{UB} - X^i \\ X^i - X^{LB} \\ U^{UB} - U^i \\ U^i - U^{LB} \end{bmatrix}, \tag{15}
$$

where $X^{UB}$ and $X^{LB}$ are the upper and lower bound constraints of the states and $U^{UB}$ and $U^{LB}$ are the upper and lower bound constraints of all the control inputs. The matrix $[A]^i$ is the same as represented in (8) 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, $\delta U^i$ is the small control errors,

$$
\delta U^i \triangleq \left[(dU_1^i)^T \ \ (dU_2^i)^T \ \ \cdots \ \ (dU_N^i)^T\right].
$$

Output constraints can be added by using the definition in (10) to obtain,

$$
\begin{bmatrix} [B]^i \\ -[B^i] \end{bmatrix} \delta U^i \leq \begin{bmatrix} Y^{UB} - Y^i \\ Y^i - Y^{LB} \end{bmatrix}, \tag{16}
$$

where $Y^{UB}$ and $Y^{LB}$ are the upper and lower output constraints. The cost function in (14) and the constraints in (15) and (16) can be solved with any standard quadratic programming solving method (Kumar et al., 2019).

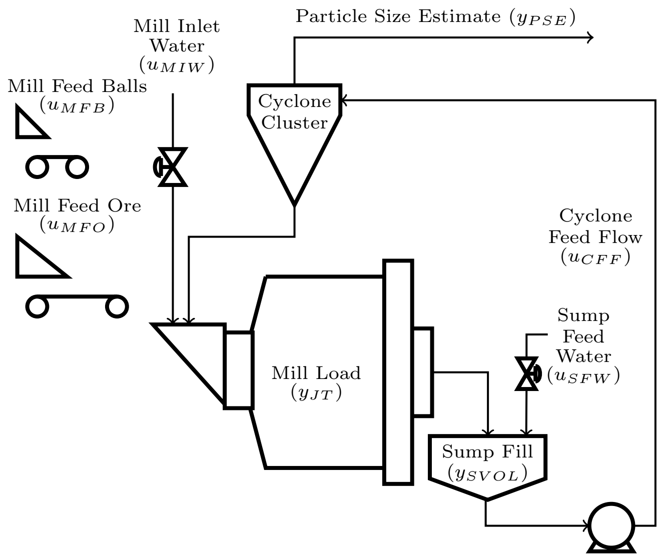
## 3. NONLINEAR MODEL PREDICTIVE CONTROL

The constrained NMPC can be formulated as,

$$
\begin{aligned}
\min_{U_k} J(X_k, U_k) = \frac{1}{2}\sum_{k=1}^{N_p} (Y_k - Y_k^*)^T Q_{mpc} (Y_k - Y_k^*) \\
+ \frac{1}{2}\sum_{k=1}^{N_c} (U_{k+1} - U_k)^T R_{mpc} (U_{k+1} - U_k)
\end{aligned} \tag{17}
$$

s.t.

$$
\begin{aligned}
X_{k+1} &= F_k (X_k, U_k) \\
Y_k &= h_k (X_k) \\
U^{LB} &\leq U \leq U^{UB} \\
X^{LB} &\leq X \leq X^{UB} \\
Y^{LB} &\leq Y \leq Y^{UB}
\end{aligned} \tag{18}
$$



Particle Size Estimate ($y_{PSE}$)
Mill Inlet Water ($u_{MIW}$)
Mill Feed Balls ($u_{MFB}$)
Cyclone Cluster
Mill Feed Ore ($u_{MFO}$)
Cyclone Feed Flow ($u_{CFF}$)
Sump Feed Water ($u_{SFW}$)
Mill Load ($y_{JT}$)
Sump Fill ($y_{SVOL}$)

Fig. 1. A single-stage grinding mill circuit.

where $N_p$ is the prediction horizon, $N_c$ is the control horizon, $Y_k^*$ is the desired output and $k$ represents the time steps. The minimization problem in (17) can be solved using an appropriate numerical optimization routine (Nocedal and Wright, 2006).

## 4. GRINDING MILL CIRCUIT MODEL

The constrained MPSP controller in 2 and the constrained NMPC controller in 3 are applied in simulation to the single-stage grinding mill circuit shown in Fig. 1 (Le Roux et al., 2013). Only a brief summary of the circuit is given below.

The mill, sump and hydrocyclone are the three main elements of the grinding mill circuit in Fig. 1. The variables in Fig. 1 are described in Table 1. The mill has four inputs: underflow from the hydrocyclone, mill inlet water ($u_{MIW}$), mill feed solids ($u_{MFO}$) and mill feed balls ($u_{MFB}$). These four inputs get mixed in the mill to form a slurry. The mill load of the grinding mill model is represented as a fraction of the charge inside the mill $y_{JT}$. 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 to 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 $\mu$m is given by $y_{PSE}$. The hydrocyclone underflow contains the large particles which return to the mill for further breakage.

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).

Table 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 [-] |

### 4.1 Model equations

The nonlinear state-space continuous model of the grinding mill is (Le Roux et al., 2013),

$$\begin{aligned}
\dot{x}_{mw} &= u_{MIW} - \frac{d_q \varphi x_{mw} x_{mw}}{x_{ms}+x_{mw}} + q_{cwu} \\
\dot{x}_{ms} &= \frac{u_{MFO}}{\rho_o}(1-\alpha_r) - \frac{d_q \varphi x_{mw} x_{ms}}{x_{ms}+x_{mw}} + q_{csu} + \\
&\quad \frac{\varphi P_{mill}}{\rho_o K_r}\left(\frac{x_{mr}}{x_{mr}+x_{ms}}\right) \\
\dot{x}_{mf} &= \frac{u_{MFO}}{\rho_o}\alpha_f - \frac{d_q \varphi x_{mw} x_{mf}}{x_{ms}+x_{mw}} + q_{cfu} + \frac{P_{mill}}{\rho_o K_f} \\
\dot{x}_{mr} &= \frac{u_{MFO}}{\rho_o}\alpha_r - \frac{P_{mill}\varphi}{\rho_o K_r}\left(\frac{x_{mr}}{x_{mr}+x_{ms}}\right) \\
\dot{x}_{mb} &= \frac{u_{MFB}}{\rho_B} - \frac{P_{mill}\varphi}{K_b}\left(\frac{x_{mb}}{\rho_o(x_{mr}+x_{ms})+\rho_B x_{mb}}\right) \\
\dot{x}_{sw} &= \frac{d_q \varphi x_{mw} x_{mw}}{x_{ms}+x_{mw}} - \frac{u_{CFF} x_{sw}}{x_{sw}+x_{ss}} + u_{SFW} \\
\dot{x}_{ss} &= \frac{d_q \varphi x_{mw} x_{ms}}{x_{ms}+x_{mw}} - \frac{u_{CFF} x_{sw}}{x_{sw}+x_{ss}} \\
\dot{x}_{sf} &= \frac{d_q \varphi x_{mw} x_{mf}}{x_{ms}+x_{mw}} - \frac{u_{CFF} x_{sf}}{x_{sw}+x_{ss}}
\end{aligned} \tag{19}$$

where $x_{mw}$, $x_{ms}$, $x_{mf}$, $x_{mr}$ and $x_{mb}$ are the volume of water, solids, fines, rocks and balls inside the mill respectively, $x_{sw}$, $x_{ss}$ and $x_{sf}$ are the water, solids and fines inside the sump respectively, and $q_{cwu}$, $q_{csu}$ and $q_{cfu}$ are the cyclone water, solids and fines underflow respectively. The model parameters are described in Table 2. The outputs are,

$$\begin{aligned}
y_{JT} &= \tfrac{x_{mw}+x_{ms}+x_{mr}+x_{mb}}{v_{mill}} \\
y_{SVOL} &= x_{ss}+x_{sw} \\
y_{PSE} &= \tfrac{q_{cfo}}{q_{cso}},
\end{aligned} \tag{20}$$

where $q_{cfo}$ and $q_{cso}$ are the volumetric flow-rates of the fines and the solids at the overflow of the hydrocyclone respectively.

The intermediate variables required in (19) for the mill are defined as,

$$\begin{aligned}
\varphi &= \begin{cases} \sqrt{1-\left(\varepsilon_0^{-1}-1\right)\frac{x_s}{x_w}}; & \frac{x_s}{x_w} \le \left(\varepsilon_0^{-1}-1\right)^{-1} \\ 0; & \frac{x_s}{x_w} > \left(\varepsilon_0^{-1}-1\right)^{-1} \end{cases} \\
P_{\text{mill}} &= P_{\max}\left\{1-\delta_{Pv}Z_x^2 - \delta_{Ps}Z_r^2\right\}\alpha_c \\
Z_x &= \frac{x_{mw}+x_{mr}+x_{ms}+x_{mb}}{v_{mill}v_{P_{max}}} - 1 \\
Z_r &= \frac{\varphi}{\varphi_{P_{max}}} - 1,
\end{aligned} \tag{21}$$

Table 2. Circuit parameter descriptions.

| Parameter | Value | Description |
|---|---|---|
| $\alpha_f$ | 0.055 | Fraction fines in the ore |
| $\alpha_r$ | 0.465 | Fraction rock in the ore |
| $\alpha_c$ | 0.72 | Fraction of critical mill speed |
| $\alpha_{su}$ | 1.50 | Parameter related to fraction solids in underflow |
| $C_1$ | 0.6 | Constant |
| $C_2$ | 0.7 | Constant |
| $C_3$ | 4.0 | Constant |
| $\delta_{ps}$ | 2.90 | Power-change parameter for fraction solids in the mill |
| $\delta_{pv}$ | 2.90 | Power-change parameter for the volume of mill filled |
| $\rho_B$ | 7.85 | Density of steel balls [t/m$^3$] |
| $\rho_S$ | 3.2 | Density of feed ore [t/m$^3$] |
| $\varepsilon_c$ | 111.85 | Maximum fraction solids by volume of slurry at 0 slurry flow |
| $\varepsilon_{sv}$ | 0.6 | Parameter related to coarse split [m$^3$/h] |
| $K_b$ | 90.0 | Steel abrasion factor [kWh/t] |
| $K_f$ | 31.31 | Power needed per tonne of fines produced [kWh/t] |
| $K_r$ | 8.06 | Rock abrasion factor [kWh/t] |
| $\varphi_{P_{max}}$ | 0.57 | Rheology factor for maximum mill power draw |
| $P_{max}$ | 1670 | Maximum mill motor power draw [kW] |
| $v_{mill}$ | 59.12 | Mill volume [m$^3$] |
| $v_{P_{max}}$ | 0.34 | Fraction of mill volume filled for maximum power draw |
| $d_q$ | 84.50 | Volumetric flow per "flowing volume" driving force [h$^{-1}$] |
| $\chi_P$ | 0 | Cross-term for maximum power draw |

where $\varphi$ is a rheology factor, $P_{mill}$ is the power draw of the grinding mill, $Z_x$ is the effect of the mill charge on the power draw, and $Z_r$ is the effect of the rheology of the mill charge on the power draw. The intermediate variables required in (19) and (20) for the hydrocyclone are defined as,

$$\begin{aligned}
q_{ccu} =& \frac{u_{CFF}(x_{ss}-x_{sf})}{x_{sw}+x_{ss}}\left(1-C_1\exp\left(\frac{-u_{CFF}}{\varepsilon_c}\right)\right)\times \\
& \left(1-\left(\frac{x_{ss}}{C_2(x_{sw}+x_{ss})}\right)^{C_3}\right)\left(1-\left(\frac{x_{sf}}{x_{ss}}\right)^{C_3}\right) \\
F_u =& 0.6-\left(0.6-\frac{x_{ss}}{x_{sw}+x_{ss}}\right)\exp\left(\frac{-q_{ccu}}{\alpha_{su}\varepsilon_c}\right) \\
q_{cwu} =& \frac{x_{sw}(q_{ccu}-F_u q_{ccu})}{F_u x_{sw}+F_u x_{sf}-x_{sf}} \\
q_{cfu} =& \frac{x_{sf}(q_{ccu}-F_u q_{ccu})}{F_u x_{sw}+F_u x_{sf}-x_{sf}} \\
q_{csu} =& q_{ccu}+\frac{x_{sf}(q_{ccu}-F_u q_{ccu})}{F_u x_{sw}+F_u x_{sf}-x_{sf}} \\
q_{cso} =& \frac{u_{CFF}x_{ss}}{x_{ss}+x_{sw}} - q_{csu} \\
q_{cfo} =& \frac{u_{CFF}x_{sf}}{x_{ss}+x_{sw}} - q_{cfu}
\end{aligned} \tag{22}$$

## 5. SIMULATION

In this section, constrained MPSP and constrained NMPC are applied to the grinding mill circuit and the results are compared.

### 5.1 Simulation settings

The comparison made use of the following general settings:

- The simulation time is 5 h.
- The sampling time is $T_s = 10\,\text{s}$.
- Full state feedback is assumed.
- The nonlinear state-space description of the circuit in (18) is discretized using the Runge-Kutta fourth order method.
- The ball feed-rate $u_{MFB}$ is kept as 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,

$$X_0 = [x_{mw}, x_{ms}, x_{mf}, x_{mr}, x_{mb}, x_{sw}, x_{ss}, x_{sf}]^T = [3.78,\ 3.45,\ 1.08,\ 1.86,\ 9.23,\ 3.79,\ 2.11,\ 0.66]^T, \tag{23}$$

$$U_0 = [u_{MFO},\ u_{SFW},\ u_{CFF}]^T = [66.9,\ 67.1,\ 267]^T \tag{24}$$

and

$$Y_{sp} = [y_{JT},\ y_{SVOL},\ y_{PSE}]^T = [0.31,\ 5.90,\ 0.60]^T. \tag{25}$$

- The input ranges for the simulations were ,

$$[u_{MFO},\ u_{SFW},\ u_{CFF}]_{LB}{}^T = [0,\ 0,\ 100]^T$$
$$[u_{MFO},\ u_{SFW},\ u_{CFF}]_{UB}{}^T = [100,\ 150,\ 500]^T. \tag{26}$$

- The output constraints for the simulation were chosen as,

$$[y_{JT},\ y_{SVOL},\ y_{PSE}]_{LB}{}^T = [0.25,\ 1.0,\ 0.5]^T$$
$$[y_{JT},\ y_{SVOL},\ y_{PSE}]_{UB}{}^T = [0.45,\ 8.0,\ 0.8]^T. \tag{27}$$

- State noise is added to the feedback as a normal Gaussian distribution of $\mathcal{N}(0,\ (0.01X_0)^2)$.
- The MPSP and NMPC objective functions in (14) and (17) are solved using sequential least squares quadratic programming.

The desired set-points are kept constant at the nominal values of the plant. Disturbances are introduced to activate the constraints of the controllers:

- A change in the mill feed size distribution is made by increasing the fraction of rocks in the ore fed to the mill, $\alpha_r$, with 50 % of its nominal value from $t = 0.5\,\text{h}$ to $t = 2.1\,\text{h}$.
- A change in the ore hardness is made by increasing the power needed per ton of fines produced, $\phi_f$, with 100 % of its nominal value from $t$ =1.5 h to $t$ =4 h.

For the constrained MPSP simulation the following applies:

- As shown in (13), the prediction horizon and control horizon are equal for MPSP. In this case, $N = 36$.
- The weighting matrices for the MPSP controller were chosen to normalize the unputs and outputs and prioritize sepoint following of $y_{PSE}$ le Roux et al. (2014), such that,

$$Q_{mpsp} = \text{diag}([1448.91, 1, 9669.44])$$
$$R_{mpsp} = 10^{-3}\,\text{diag}([3.48, 0.22, 0.22]). \tag{28}$$

Table 3. Iteration time results of the MPSP simulation and the NMPC simulations.

| Simulated Controller | $\bar{x}$ [s] | $\sigma$ [s] | Maximum [s] |
|---|---|---|---|
| $MPSP$ | 3.187 | 1.681 | 6.860 |
| $NMPC_{Nc36b3}$ | 5.252 | 0.463 | 5.557 |
| $NMPC_{Nc12b1}$ | 5.200 | 0.301 | 5.517 |
| $NMPC_{Nc12b3}$ | 1.983 | 0.130 | 2.487 |

- The MPSP algorithm terminates for each iterative step if the algorithm has executed 10 times, or if the conditions below are met,

$$\frac{\left\|Y_k^i - Y_k^*\right\|_2}{\|Y_k^*\|} < [0.005,\ 0.005,\ 0.001]^T$$
$$\frac{\left\|U_k^{i+1} - U_k^*\right\|_2}{\|U_k^*\|} < [0.01,\ 0.01,\ 0.01]^T. \tag{29}$$

Multiple NMPC controller configurations are simulated. For the constrained NMPC simulations the following settings apply:

- The prediction horizon is $N_p = 36$ and the control horizons are changed between $N_c = 36$ and $N_c = 12$ respectively.
- Move-blocking of $N_B = 3$ was applied to two of the controllers.
- All of the NMPC controllers used warm-starting. (The previous control calculation is used as the initial guess value for the optimisation routine (Wang and Boyd, 2010))
- The weighting matrices for each of the NMPC controllers in (17) were obtained by scaling the MPSP weightings according to the control horizon and the number of control moves blocked.

$$Q_{mpc} = Q_{mpsp}$$
$$R_{mpc} = R_{mpsp} \times \frac{N_p N_c}{N_B}. \tag{30}$$

- The NMPC algorithm terminates at each iterative step when the optimization routine has executed a maximum of 10 times, or the algorithm converged to within a tolerance of 0.001.

### 5.2 Results

The simulation was done in Python. The results of the simulation are shown in Figs. 2 to 4. The time it took to calculate a new control step for each $k$-th time step was measured in the simulation and is shown in Table 3, where $\bar{x}$ and $\sigma$ represent the mean and standard deviation of the iteration times of the simulations respectively. The NMPC simulation results show a subscript of "$NcCCbB$", where $CC$ represent the control horizon and $B$ represents the number of move-blocking. The input and output constraints are adhered to. The simulation was executed on an Intel(R) Core(TM) i5-8400 (6 Core) 2.80 Ghz processor with 20 GB RAM running on Microsoft Windows 10 Home operating system.

## 6. DISCUSSION

The simulation results show that the constrained MPSP and NMPC controllers can reject disturbances with the same efficiency. The $NMPC_{Nc12b3}$ controller is the least

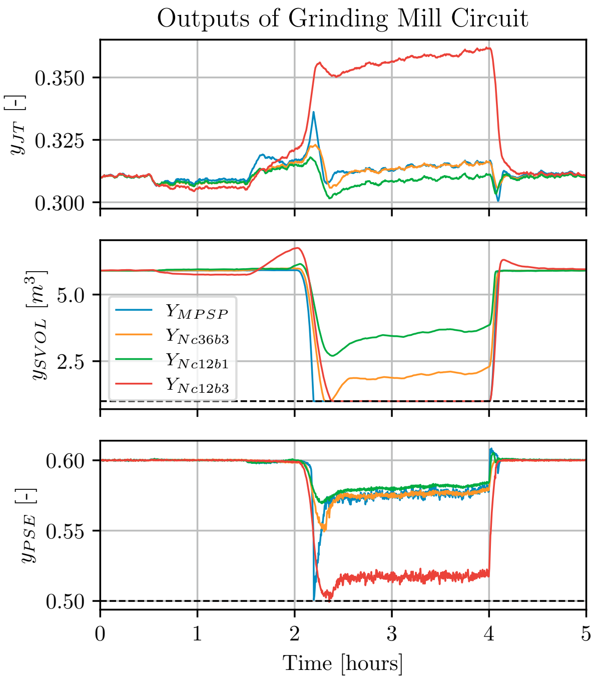


Fig. 2. NMPC and MPSP simulation outputs of the grinding mill circuit with state noise.

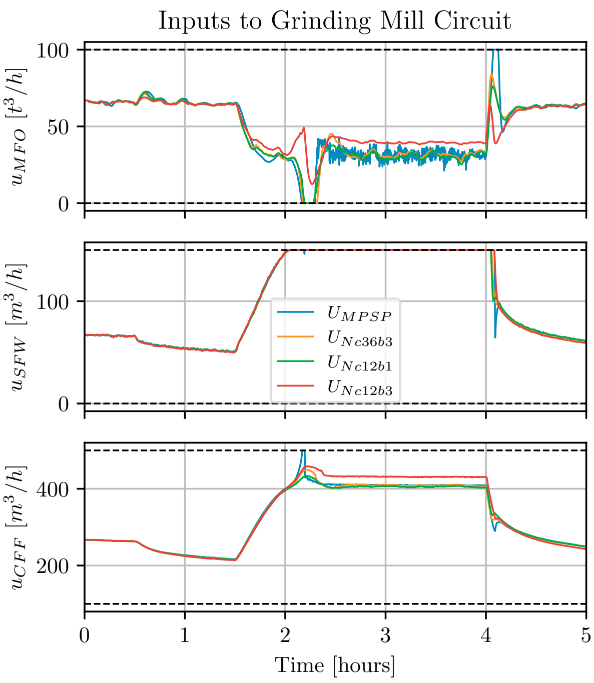


Fig. 3. NMPC and MPSP simulation inputs to the grinding mill circuit with state noise.

computationally heavy algorithm, but has the worst performance. The $NMPC_{Nc12b1}$ has the best output results of all the controllers, but is 2.622 times slower than the $NMPC_{Nc12b3}$ controller. The MPSP controller shows better performance than the $NMPC_{Nc12b3}$ controller, but is 1.607 times slower. The MPSP controller has more aggressive control moves than any of the NMPC controllers.

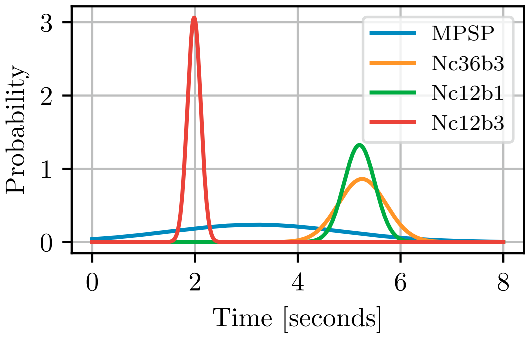


Fig. 4. Normal distribution of the time necessary to calculate a new input for the respective controllers.

The results in Fig. 4 indicate that the time necessary to calculate an MPSP control move is less consistent than for NMPC.

## 7. CONCLUSION

The constrained MPSP method is a viable option for use in the mineral processing industry where the computational time of constrained NMPC is too long for practical implementation without compromising performance. Future work may investigate the robustness of constrained MPSP.

## REFERENCES

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

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 T. Contr. Syst. T.*, 18(1), 222–229.

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.*, 141(2).

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

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

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.

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

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

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

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