Journal of Process Control 129 (2023) 103067

Contents lists available at ScienceDirect

Journal of Process Control

journal homepage: www.elsevier.com/locate/jprocont

Optimal control of mineral processing plants using constrained model predictive static programming

Zander M. Noome [a], Johan D. le Roux [a,*], Radhakant Padhi [b]

[a] Department of Electrical, Electronic, and Computer Engineering, University of Pretoria, Pretoria, South Africa
[b] Department of Aerospace Engineering, Indian Institute of Science, Bangalore 560012, India

ARTICLE INFO

Article history:
Received 29 November 2022
Received in revised form 11 August 2023
Accepted 21 August 2023
Available online 4 September 2023

Keywords:
Flotation
Grinding mills
Mineral processing
Model predictive control
Model predictive static programming
Optimal control
Process control

ABSTRACT

The model predictive static programming (MPSP) technique, which is extended recently to incorporate applicable state and control constraints, operates on the philosophy of nonlinear model predictive control (NMPC). However, it reduces the problem into a lower-dimensional problem of control variables alone, thereby enhancing computational efficiency significantly. Because of this, problems with larger dimensions and/or increased complexity can be solved using MPSP without changing the computational infrastructure. In this paper, the MPSP technique with applicable constraints is applied to two challenging control problems in the mineral processing industry: (i) a single-stage grinding mill circuit model, and (ii) a four-cell flotation circuit model. The results are compared with a conventional nonlinear MPC approach. Comparison studies show that constrained MPSP executes much faster than constrained MPC with similar/improved performance. Therefore, it can be considered a potential optimal control candidate for mineral processing plants.

1. Introduction

The use of process control in industrial plants is an efficient way to maintain consistent product quality, improve throughput, optimize 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 [1,2]. NMPC is ideal for processes with relatively slow dynamics since it can be computationally intensive to apply online [3,4].

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. [5], 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 [6]. 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 [7]. Other fast NMPC schemes include a real-time iterations (RTI) algorithm to reduce the computational time of the NMPC algorithm [7–9]. 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 [9].

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 [7,10].

Explicit NMPC is another technique that is used to decrease the computational time of NMPC controllers [11]. 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) [12]. Chen et al. [13] and Zhang et al. [14] have used reinforcement learning to approximate the polyhedral regions of explicit MPC into a function for the explicit

* Corresponding author.
E-mail address: derik.leroux@up.ac.za (J.D. le Roux).

https://doi.org/10.1016/j.jprocont.2023.103067

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 [15]. Faroni et al. [16] 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 [12,17].

The computational time of the MPC controllers can also be decreased by defining termination constraints of the optimization algorithm [12]. 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. [18], 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. [19] 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 [20] developed a different approach to NMPC known as Model Predictive Static Programming (MPSP). MPSP combines two different philosophies: NMPC and Approximate Dynamic Programming [21]. The novelty of MPSP is that it converts the dynamic optimization problem to an equivalent small-dimensional state optimization problem in terms of the manipulated variable alone. The MPSP algebra relies on the recursive calculation of sensitivity matrices which relates the error of the output to the error of the control at each time step. After the conversion, assuming a quadratic cost function, the problem can be solved as a standard quadratic programming problem. 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 [22]. The simulations handled inequality state constraints by using a slack variable approach which transforms the inequality constraints into an unconstrained problem.

Li et al. [23] 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. [24] 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 in [25]. 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 [26–30].

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 [31].

The contribution of this article is the application of the constrained MPSP of Kumar et al. [24] in simulation to a grinding mill circuit as well as a flotation circuit respectively. The computational performance of the constrained MPSP is compared to the application of constrained NMPC for the same plant models and disturbances. Whereas Le Roux et al. [31] considered unconstrained MPSP only for a grinding mill circuit, this article evaluates in simulation, constrained MPSP (which is more relevant for practical problems) as applied to a grinding mill circuit and a four-cell flotation circuit. It extends earlier preliminary work where a limited simulation scenario evaluated state and input constrained MPSP for the grinding mill circuit [32].

The article is organized as follows: Section 2 describes the MPSP method and the NMPC method. Section 3.1 is the first example of MPSP and NMPC applied to a grinding mill circuit, and Section 3.2 is the second example where MPSP and NMPC is applied to a flotation circuit. Section 4 concludes the article.

2. Model Predictive Static Programming

The recently extended version of MPSP [24], which is applicable to tracking reference signals and can satisfy state and input constraints, is applied in this paper. A summary of the method described by Kumar et al. [24] is provided below.

A nonlinear system is written in discrete form as,

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

where $X_k \in \Re^n$, $U_k \in \Re^m$ and $Y_k \in \Re^p$ represent the states, inputs and outputs of the system respectively. The subscript k represents the time step and the superscript i represents the iteration index. The key aim here is to calculate a control history U_k^{i+1}, $k = 1, 2, \ldots, N$, so that the output Y_k^{i+1} will converge to the desired output Y_k^* for $k = 1, 2, \ldots, N$.

2.1. Analysis for output and state deviation

The output deviation dY_k^i can be written in terms of the state and input deviations at time steps $(k-1)$, $(k-2)$, ... , until the first time step, as follows,

$$dY_k^i = \sum_{j=1}^{k-1} \left[B_j^k\right]^i dU_j^i, \tag{2}$$

where $[B_j^k]^i$ is the sensitivity matrix, and is defined as,

$$\begin{aligned} [B_j^k]^i &= \left[\frac{\partial Y_k}{\partial X_k}\right]_{\left(X_k^i, U_k^i\right)} \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]_{\left(X_{k-1}^i, U_{k-1}^i\right)} \cdots \left[\frac{\partial F_j}{\partial X_j}\right]_{\left(X_j^i, U_j^i\right)} \\ \left[B_k^k\right]^i &= \left[\frac{\partial Y_k}{\partial U_k}\right]_{\left(X_k^i, U_k^i\right)}. \end{aligned} \tag{3}$$

Similarly, the state deviation can be written as,

$$dX_k^i = \sum_{j=1}^{k-1} \left[A_j^k\right]^i dU_j^i, \tag{4}$$

where,

$$\left[A_j^k\right]^i = \left[\frac{\partial F_{k-1}}{\partial X_{k-1}}\right]_{\left(X_{k-1}^i, U_{k-1}^i\right)} \cdots \left[\frac{\partial F_{j+1}}{\partial X_{j+1}}\right]_{\left(X_{j+1}^i, U_{j+1}^i\right)} \left[\frac{\partial F_j}{\partial U_j}\right]_{\left(X_j^i, U_j^i\right)}. \tag{5}$$

It should be noted that (2) represents the sensitivity of the output dY_k^i at the kth iteration with respect to the input changes dU_j^i at all the previous grid points ($j = 1, 2, \ldots, k-1$). Calculating $[B_j^k]^i$ and $[A_j^k]^i$ for all $k = 1, 2, 3, \ldots, N$ where N represents the control and prediction horizon, can be computationally demanding. Note that the following recursive computation reduces the computational cost significantly.

$$\left.\begin{aligned}
\left[\phi_k^k\right]^i &= I_{n\times n} \\
\left[\phi_j^k\right]^i &= \left[\phi_{j+1}^k\right]^i \left[\frac{\partial F_j}{\partial X_j}\right]_{\left(X_j^i, U_j^i\right)} \\
\left[A_j^k\right]^i &= \left[\phi_{j+1}^k\right]^i \left[\frac{\partial F_j}{\partial U_j}\right]_{\left(X_j^i, U_j^i\right)} \\
\left[B_j^k\right]^i &= \left[\frac{\partial Y_k}{\partial X_k}\right]_{\left(X_k^i, U_k^i\right)} \left[A_j^k\right]^i
\end{aligned}\right\} \quad \forall j < k$$
$$\begin{aligned}
\left[B_j^k\right]^i &= \left[\frac{\partial Y_k}{\partial U_k}\right]\left(X_k^i, U_k^i\right) && \forall j = k \\
\left[A_j^k\right]^i &= [0]_{n\times m} && \forall j \geq k \\
\left[B_j^k\right]^i &= [0]_{p\times m} && \forall j > k
\end{aligned} \tag{6}$$

Detailed derivation of these expressions is given by Kumar et al. [24].

2.2. Cost function

The analysis in Section 2.1 led to an under constrained system of equations. One way of solving this system is to optimize a cost function. The cost function chosen for each ith iteration is,

$$\begin{aligned} J^i = {} & \frac{1}{2}\sum_{k=2}^{N}\left(\Delta Y_k^i - \Delta Y_k^*\right)^T Q \left(\Delta Y_k^i - \Delta Y_k^*\right) \\ & + \frac{1}{2}\sum_{k=1}^{N-1}\left(\Delta U_k^i\right)^T R \left(\Delta U_k^i\right) \end{aligned} \tag{7}$$

where $\Delta Y_k^* = Y_k^* - Y_k^i$ is the output error with respect to the desired output Y_k^*, Q is the output weighting matrix and R is the input deviation weighting matrix. Minimizing this cost function ensures that the measured output remains close to the desired output at each grid point for the next iteration ($Y_k^{i+1} \to Y_k^*,\ \forall\, k = 1, 2, 3, \ldots, N$). Assuming small output and input deviations, (7) can be rewritten as,

$$\begin{aligned} J^i = {} & \frac{1}{2}\sum_{k=2}^{N}\left(dY_k^i - \Delta Y_k^*\right)^T Q \left(dY_k^i - \Delta Y_k^*\right) \\ & + \frac{1}{2}\sum_{k=1}^{N-1}\left(dU_k^i\right)^T R \left(dU_k^i\right). \end{aligned} \tag{8}$$

2.3. Constrained MPSP

A condensed form of the cost function in (8) can be obtained by substituting (2) and converting it into a vector form,

$$\begin{aligned} J^i = {} & \frac{1}{2}(\delta U^i)^T \left(R_k + ([B]^i)^T Q_k [B]^i\right) \delta U^i \\ & - (\delta U^i)^T \left([B]^i\right)^T Q_k (\Delta Y^{*i}) \\ & + \frac{1}{2}(\Delta Y^{*i})^T Q_k \Delta Y^{*i} \end{aligned} \tag{9}$$

where Q_k, R_k, ΔY^{*i} and $[B]^i$ are,

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

The state and input constraints applied in (9) are,

$$\begin{bmatrix} [A]^i \\ -[A^i] \\ I \\ -I \end{bmatrix} \delta U^i \leq \begin{bmatrix} X^{UB} - X^i \\ X^i - X^{LB} \\ U^{UB} - U^i \\ U^i - U^{LB} \end{bmatrix}, \tag{10}$$

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 the inputs. The matrix $[A]^i$ is the same as represented in (5) but in matrix form,

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

Finally, δU^i is the small control deviations,

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

Output constraints can be added by using the definition in (2) to obtain,

$$\begin{bmatrix} [B]^i \\ -[B^i] \end{bmatrix} \delta U^i \leq \begin{bmatrix} Y^{UB} - Y^i \\ Y^i - Y^{LB} \end{bmatrix}, \tag{11}$$

where Y^{UB} and Y^{LB} are the upper and lower output constraints.

The cost function in (9) and the constraints in (10) and (11) can be solved with any standard quadratic programming (QP) problem solver [24]. If, for example, sequential quadratic programming (SQP) is used to solve the static optimization problem, the approach inherits the stability and convergence properties of SQP. The initial control for MPSP needs to be a feasible solution, which is not a difficult task for regulation problems as it can be assumed to be a steady-state solution without the presence of disturbances. In the case of an infeasible solution, SQP can be adapted by introducing additional variables to define a higher dimensional subproblem consistent with constraints [33, 34], or infeasible interior point algorithms can be considered [35]. The quadratic programming optimization algorithm used for the MPSP method is *quadprog* [36], which uses the dual method described in Goldfarb and Idnani [33].

2.4. Nonlinear model predictive control summary

The constrained NMPC can be formulated as,

$$\begin{aligned}\min_{U_k,U_{k+1},\ldots,U_{k+N_c}} J(X_k,U_k) = \frac{1}{2}\sum_{k=1}^{N_p}\left(Y_k - Y_k^*\right)^T Q\left(Y_k - Y_k^*\right) \\ + \frac{1}{2}\sum_{k=1}^{N_c}\left(U_{k+1} - U_k\right)^T R\left(U_{k+1} - U_k\right)\end{aligned} \tag{12}$$

s.t.

$$\begin{aligned}X_{k+1} &= F_k(X_k, U_k)\\ Y_k &= h_k(X_k)\\ U^{LB} &\le U \le U^{UB}\\ X^{LB} &\le X \le X^{UB}\\ Y^{LB} &\le Y \le Y^{UB}\end{aligned} \tag{13}$$

where N_p is the prediction horizon, N_c is the control horizon, Y_k^* is the desired outputs, U_k is the input, X_k is the states of the plant, U^{LB} and U^{UB} are the lower and upper bounds of the input constraints, X^{LB} and X^{UB} are the lower and upper bounds of the state constraints, Y^{LB} and Y^{UB} are the lower and upper bounds of the output constraints, and R and Q are the input deviation weighting matrices and output error weighting matrices respectively.

The objective function (12) is difficult to solve for nonlinear plant models because it is difficult for the optimization algorithm to differentiate between local and global minima. Linearizing the nonlinear plant model about a nominal operation point is an effective way to reduce the objective function complexity [37]. The linearization of the plant model reduces computational time significantly because the objective function can be condensed into a convex quadratic programming problem [6]. The MPSP method can be seen as a linearization algorithm.

The minimization problem in (12) can be solved using an appropriate numerical optimization routine which can solve nonlinear problems [34]. A standard NMPC approach with warm-starting and move-blocking is implemented in both simulation case studies to act as a baseline for the comparison of the MPSP controller. The optimization algorithm used is the *Least Squares Subproblems* method described in Kraft [38].

3. Simulation case studies

The simulations in both case studies 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.

3.1. Example 1: A grinding mill circuit

The constrained MPSP controller in Section 2 and the constrained NMPC controller in Section 2.4 are applied in simulation to a grinding mill circuit. A single-stage grinding mill circuit is shown in Fig. 1 [39,40]. The variables in Fig. 1 are described in Table 1. An overview of the process can be found in Le Roux et al. [40].

3.1.1. Grinding mill circuit process model

A brief overview of the model of the grinding mill circuit in Fig. 1 is given below. A complete description of the plant model is given by Le Roux et al. [40]. The model nomenclature is given in Table 2. The variable and parameter values were taken from Le Roux et al. [31].

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 [–]

The state-space model of the grinding mill circuit is,

$$\begin{aligned}\dot{x}_{mw} &= u_{MIW} - \frac{d_q\varphi x_{mw}x_{mw}}{x_{ms}+x_{mw}} + Q_{cwu}\\ \dot{x}_{ms} &= \frac{u_{MFO}}{\rho_o}(1-\alpha_r) - \frac{d_q\varphi x_{mw}x_{ms}}{x_{ms}+x_{mw}} + Q_{csu} + \\ &\quad \frac{\varphi P_{\text{mill}}}{\rho_o K_{RC}}\left(\frac{x_{mr}}{x_{mr}+x_{ms}}\right)\\ \dot{x}_{mf} &= \frac{u_{MFO}}{\rho_o}\alpha_f - \frac{d_q\varphi x_{mw}x_{mf}}{x_{ms}+x_{mw}} + Q_{cfu} + \frac{P_{\text{mill}}}{\rho_o K_{FP}}\\ \dot{x}_{mr} &= \frac{u_{MFO}}{\rho_o}\alpha_r - \frac{P_{\text{mill}}\varphi}{\rho_o K_{RC}}\left(\frac{x_{mr}}{x_{mr}+x_{ms}}\right)\\ \dot{x}_{mb} &= \frac{u_{MFB}}{\rho_b} - \frac{P_{\text{mill}}\varphi}{K_{BC}}\left(\frac{x_{mb}}{\rho_o(x_{mr}+x_{ms})+\rho_b x_{mb}}\right)\\ \dot{x}_{sw} &= \frac{d_q\varphi x_{mw}x_{mw}}{x_{ms}+x_{mw}} - \frac{u_{CFF}x_{sw}}{x_{sw}+x_{ss}} + u_{SFW}\\ \dot{x}_{ss} &= \frac{d_q\varphi x_{mw}x_{ms}}{x_{ms}+x_{mw}} - \frac{u_{CFF}x_{sw}}{x_{sw}+x_{ss}}\\ \dot{x}_{sf} &= \frac{d_q\varphi x_{mw}x_{mf}}{x_{ms}+x_{mw}} - \frac{u_{CFF}x_{sf}}{x_{sw}+x_{ss}}\end{aligned} \tag{14}$$

where u_{MIW}, u_{MFO} and u_{MFB} are the flow rates of the mill inlet water, the feed ore and the feed balls respectively, u_{SFW} is the sump water dilution flow-rate, 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.

The outputs are,

$$\begin{aligned}y_{JT} &= \tfrac{x_{mw}+x_{ms}+x_{mr}+x_{mb}}{v_{\text{mill}}}\\ y_{SVOL} &= x_{ss}+x_{sw}\\ y_{PSE} &= \tfrac{Q_{cfo}}{Q_{cso}},\end{aligned} \tag{15}$$

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, Y_{JT} is the volume fraction filled in the grinding mill, Y_{SVOL} is the volume of the sump and Y_{PSE} is the particle size distribution of the hydrocyclone overflow.

The intermediate variables required in (14) for the mill are,

$$\begin{aligned}\varphi &= \begin{cases}\sqrt{1-\left(\varepsilon_c^{-1}-1\right)\frac{x_s}{x_w}}; & \frac{x_s}{x_w} \le \left(\varepsilon_0^{-1}-1\right)^{-1}\\ 0; & \frac{x_s}{x_w} > \left(\varepsilon_0^{-1}-1\right)^{-1}\end{cases}\\ P_{\text{mill}} &= P_{\max}\left\{1-\delta_v\left(\frac{x_{mw}+x_{mr}+x_{ms}+x_{mb}}{v_{\text{mill}}\ v_{P_{\max}}}-1\right)^2 \right.\\ &\quad \left. -\ \delta_s\left(\frac{\varphi}{\varphi_N}-1\right)^2\right\}\phi_c,\end{aligned} \tag{16}$$

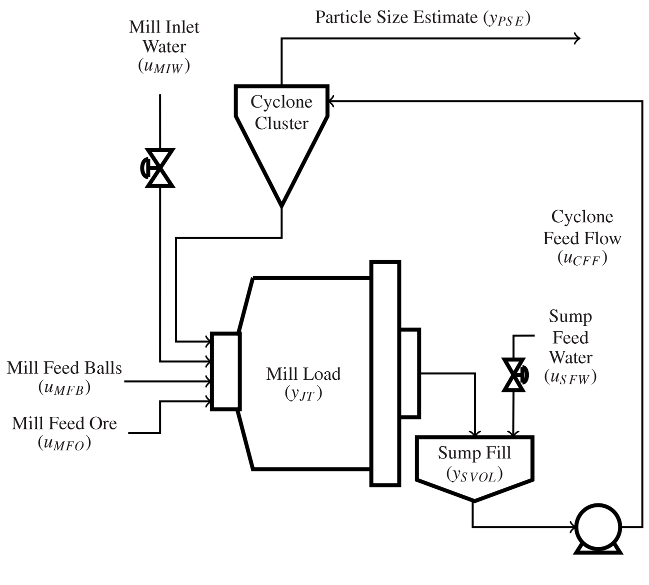
Mill Inlet Water (u_{MIW})
Particle Size Estimate (y_{PSE})
Cyclone Cluster
Cyclone Feed Flow (u_{CFF})
Sump Feed Water (u_{SFW})
Mill Feed Balls (u_{MFB})
Mill Feed Ore (u_{MFO})
Mill Load (y_{JT})
Sump Fill (y_{SVOL})

Fig. 1. A single-stage grinding mill circuit [39].

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 (14) and (15) for the hydrocyclone are,

$$
\begin{aligned}
Q_{ccu} =& \frac{u_{CFF}\left(x_{ss}-x_{sf}\right)}{x_{sw}+x_{ss}}\left(1-C_1\exp\left(\frac{-u_{CFF}}{\varepsilon_c}\right)\right)\times \\
&\left(1-\left(\frac{x_{ss}}{C_2\left(x_{sw}+x_{ss}\right)}\right)^{C_3}\right)\left(1-\left(\frac{x_{sf}}{x_{ss}}\right)^{C_3}\right) \\
F_u =& 0.6-\left(0.6-\frac{x_{ss}}{x_{sw}+x_{ss}}\right)\exp\left(\frac{-Q_{ccu}}{\alpha_{su}\varepsilon_c}\right) \\
Q_{cwu} =& \frac{x_{sw}\left(Q_{ccu}-F_uQ_{ccu}\right)}{F_ux_{sw}+F_ux_{sf}-x_{sf}} \\
Q_{cfu} =& \frac{x_{sf}\left(Q_{ccu}-F_uQ_{ccu}\right)}{F_ux_{sw}+F_ux_{sf}-x_{sf}} \\
Q_{csu} =& q_{ccu}+\frac{x_{sf}\left(Q_{ccu}-F_uQ_{ccu}\right)}{F_ux_{sw}+F_ux_{sf}-x_{sf}} \\
Q_{cso} =& \frac{u_{CFF}x_{ss}}{x_{ss}+x_{sw}}-Q_{csu} \\
Q_{cfo} =& \frac{u_{CFF}x_{sf}}{x_{ss}+x_{sw}}-Q_{cfu}.
\end{aligned}
\tag{17}
$$

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.

3.1.2. Grinding mill circuit simulation

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 [5]. The nonlinear state-space description of the circuit in (14) 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 [39,41,42]. Since the observer design falls outside the scope of this study, full-state feedback is assumed.

Table 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	111.9	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 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,

$$
\begin{aligned}
X_0 &= \left[x_{mw}, x_{ms}, x_{mf}, x_{mr}, x_{mb}, x_{sw}, x_{ss}, x_{sf}\right]^T \\
&= [3.78,\ 3.45,\ 1.08,\ 1.86,\ 9.23,\ 3.79,\ 2.11,\ 0.66]^T
\end{aligned}
\tag{18}
$$

$$
U_0 = [u_{MFO},\ u_{SFW},\ u_{CFF}]^T = [66.9,\ 67.1,\ 267]^T \tag{19}
$$

$$
Y_{sp} = \left[y_{JT},\ y_{SVOL},\ y_{PSE}\right]^T = [0.31,\ 5.90,\ 0.60]^T \tag{20}
$$

The input and output constraints for the grinding mill are,

$$
\begin{bmatrix}0\\0\\100\end{bmatrix} \le \begin{bmatrix}u_{MFO}\\u_{SFW}\\u_{CFF}\end{bmatrix} \le \begin{bmatrix}100\\150\\500\end{bmatrix}
$$

$$
\begin{bmatrix}0.25\\1.0\\0.5\end{bmatrix} \le \begin{bmatrix}y_{JT}\\y_{SVOL}\\y_{PSE}\end{bmatrix} \le \begin{bmatrix}0.45\\8.0\\0.8\end{bmatrix}
$$

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.

MPSP configuration

Two MPSP controllers were simulated. The first controller, $MPSP_{36}$, has a control/prediction horizon of $N = 36$ and the second controller, $MPSP_{72}$, has a control/prediction horizon of

Table 3
Iteration time results of the MPSP and the NMPC simulations for the grinding mill circuit.

Simulated Controller	$\bar{x}$ [s]	σ [s]
$MPSP_{36}$	0.670	0.544
$MPSP_{72}$	1.735	1.260
$NMPC_{B3}$	0.869	0.581
$NMPC_{B1}$	4.878	3.167

$N = 72$ according to (8). The $MPSP_{72}$ controller uses a sampling time of $T_s = 5$ to obtain similar results as the $MPSP_{36}$ controller but with an increased number of control variables.

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 [31],

$$Q_{mpsp} = \text{diag}([36.22, 0.013, 1510])$$
$$R_{mpsp} = 10^{-3}\,\text{diag}([3.481, 0.218, 0.218]),$$

where Q_{mpsp} and R_{mpsp} are the weighting matrices for the MPSP objective function in (9). The MPSP algorithm terminates if the algorithm has been executed 15 times, or if either of the conditions below are met,

$$\frac{\left\|Y_k^i(1) - Y_k^*(1)\right\|_2}{\left\|Y_k^*(1)\right\|_2} < 0.5$$
$$\frac{\left\|Y_k^i(2) - Y_k^*(2)\right\|_2}{\left\|Y_k^*(2)\right\|_2} < 0.5$$
$$\frac{\left\|Y_k^i(3) - Y_k^*(3)\right\|_2}{\left\|Y_k^*(3)\right\|_2} < 0.1,$$

where $Y_k^i(p)$ refers to the pth entry in the output vector Y_k^i.

The MPSP optimization problem is solved using a strictly convex QP solver.

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 [12].

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,

$$Q_{nmpc} = \text{diag}([5.489, 0.015, 496.7])$$
$$R_{nmpc} = 10^{-3}\,\text{diag}([3.481, 0.218, 0.218]),$$

where Q_{nmpc} and R_{nmpc} are the weighting matrices for the NMPC objective function in (12). The NMPC algorithm terminates when the sequential programming problem optimization routine has 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 setpoint 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.

Simulation results and discussion

The results of the simulation are shown in Figs. 2 to 3. The time to calculate a new control step for each kth time step was measured in the simulation and is shown in Table 3, where $\bar{x}$ and σ represent the mean and standard deviation of the iteration times of the simulations respectively. The Integral Absolute Error (IAE) performance indicator is shown in Table 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 $MPSP_{36}$ and $NMPC_{B3}$ controllers are the least computationally demanding algorithms as shown in Fig. 4.

All the controller computational times do not increase when any constraints are active, but the computational time of the $NMPC_{B3}$, $MPSP_{36}$ and $MPSP_{72}$ controller increases 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 $NMPC_{B1}$ because the computational time stays constant at approximately 7.9 s during the simulation until plant disturbances are removed.

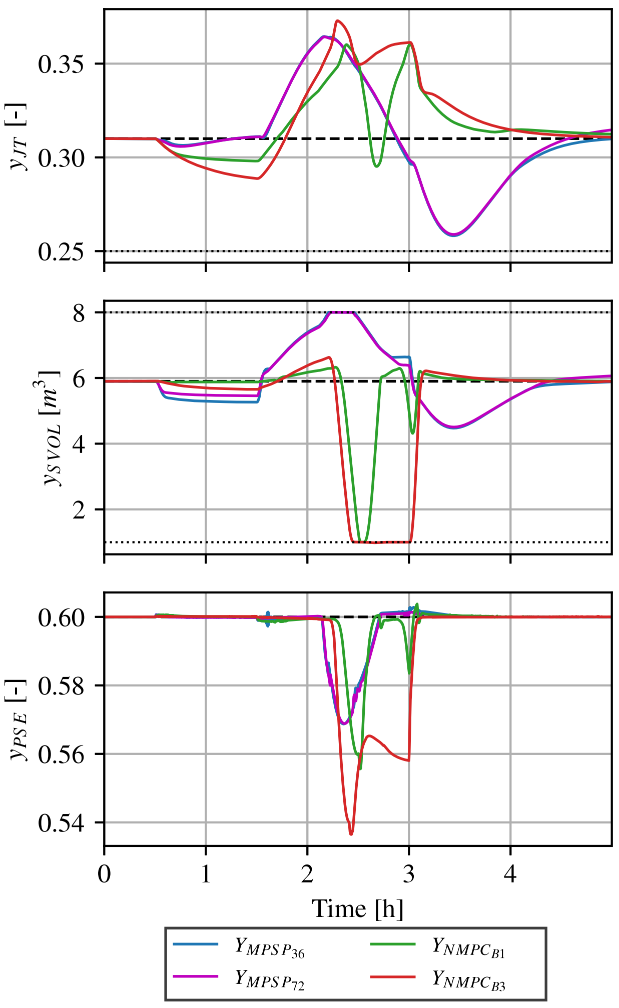

Fig. 2. MPSP and NMPC simulation outputs of the grinding mill circuit. The nominal conditions of each output is shown in (20) and is represented by the dashed line.

Table 4
IAE performance of the MPSP and the NMPC simulations for the grinding mill circuit.

Simulated Controller	IAE(Y_{JT})	IAE(Y_{SVOL})	IAE(Y_{PSE})
$MPSP_{36}$	315.6	13040	43.40
$MPSP_{72}$	313.6	12144	42.23
$NMPC_{B3}$	310.8	14820	115.18
$NMPC_{B1}$	213.2	5795	36.37

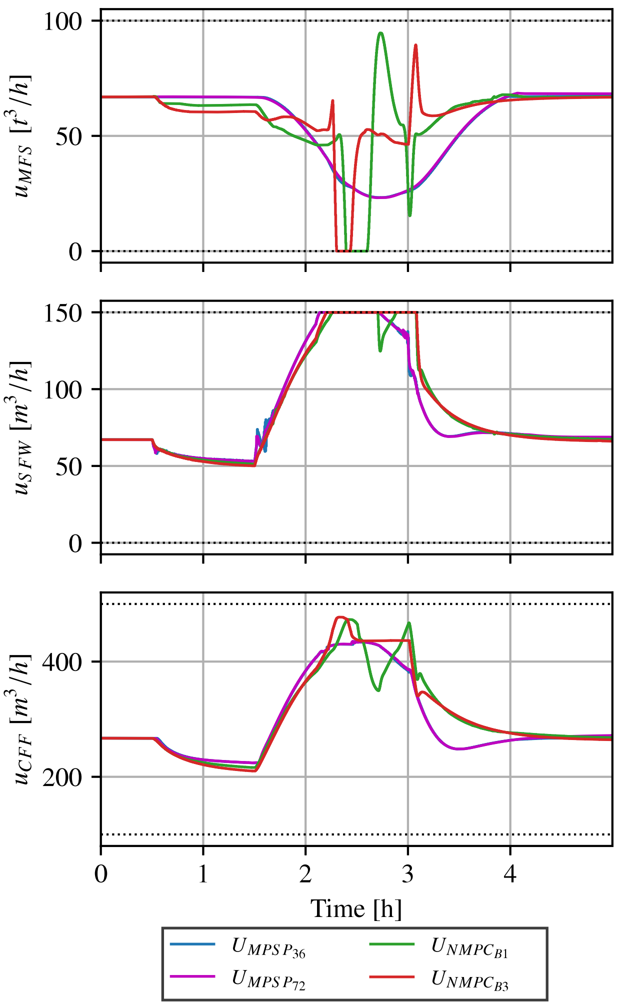

Fig. 3. MPSP and NMPC simulation inputs to the grinding mill circuit. The nominal conditions of each input is shown in (19).

The reason for the computational time of the $NMPC_{B1}$ controller staying constant during the simulations is because 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 $MPSP_{36}$ and the $NMPC_{B3}$ controller is minimal. The $NMPC_{B3}$ controller is on average 200 ms slower than the $MPSP_{36}$ controller as seen in Table 3. The $MPSP_{72}$ controller is on average 2.6 times slower than the $MPSP_{36}$ controller.

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}.

The number of decision variables of the optimization algorithm of the $NMPC_{B1}$ controller is equal to the control horizon of $N_c = 12$. The number of decision variables of the optimization algorithm of the $MPSP_{72}$ controller is equal to the control/prediction horizon of $N = 72$. The $MPSP_{72}$ controller has more decision variables compared to the $NMPC_{B1}$ controller by a factor of 6 and calculates a new input on average 2.8 times faster than $NMPC_{B1}$.

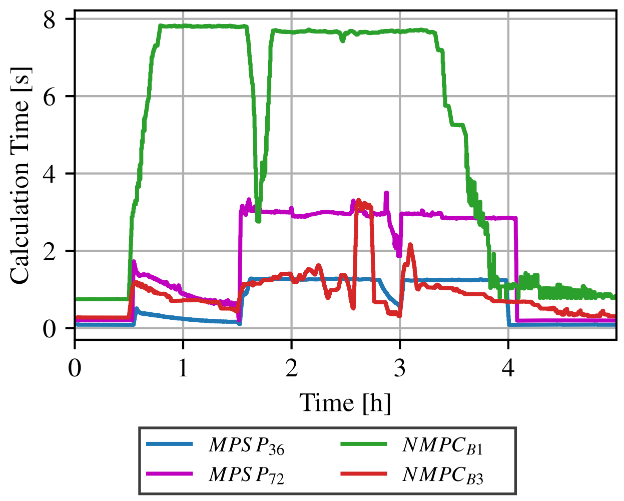

Fig. 4. The time necessary to calculate a new input for the respective controllers.

The controller that performs the best when comparing the IAE values in Table 4 is the $NMPC_{B1}$ controller. 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.

3.2. Example 2: A flotation circuit

The constrained MPSP controller in Section 2 and the constrained NMPC controller in Section 2.4 are applied in simulation to a four-cell flotation circuit.

3.2.1. Flotation process description

A four-cell flotation circuit is shown in Fig. 5 [43]. The variables in Fig. 5 are described in Table 5. Only a brief overview of the process is given below.

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 is shown in Fig. 5. 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}.

3.2.2. Flotation process model

A brief overview of the model of the flotation circuit in Fig. 5 is given below. A complete description is given by Oosthuizen et al. [43]. The model nomenclature is given in Table 5. The variable and parameters values were taken from Oosthuizen et al. [43].

The state-space continuous model for each cell k is,

$$\begin{aligned}
\dot{D}_{BF_k} &= \frac{K_{BS_{Jg}} J_{g_k} + K_{BS_\lambda} \lambda_{air_k} - D_{BF_k}}{\lambda_{air_k}} \\
\dot{\alpha}_k &= \frac{K_{\alpha_{BF}} D_{BF_k} + K_{\alpha_{Jg}} J_{g_k} - \alpha_k}{\lambda_{air_k}} \\
\dot{L}_k &= \left(Q_{F_k} - Q_{T_k} - Q_{C_k}\right)/A_k \\
\dot{M}^i_k &= \dot{M}^i_{F_k} - \dot{M}^i_{T_k} - \dot{M}^i_{C_k}
\end{aligned} \tag{21}$$

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^i_k [kg] is the mass of

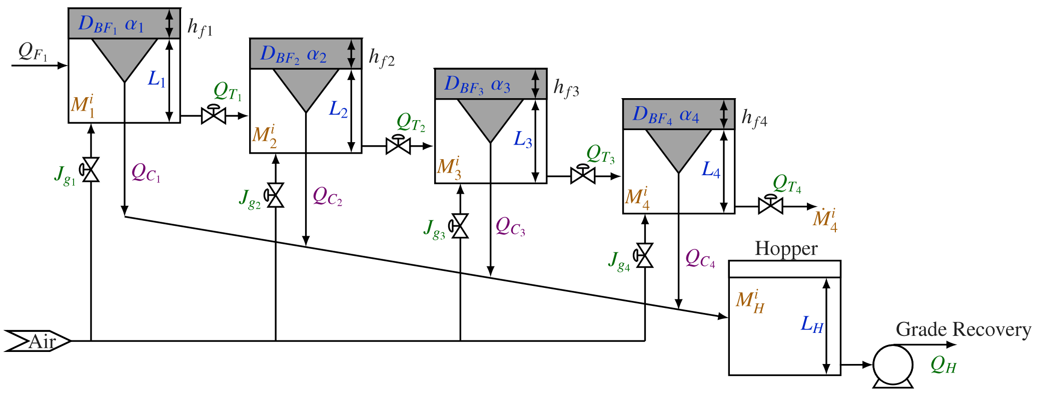

Fig. 5. A four-cell flotation bank with a hopper.

Table 5
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
$K_{BS_{Jg}}$	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
$K_{\alpha_{Jg1}}$	7.20	The effect of the superficial gas velocities on the air recovery in cell 1
$K_{\alpha_{Jg2}}$	7.30	The effect of the superficial gas velocities on the air recovery in cell 2
$K_{\alpha_{Jg3}}$	7.00	The effect of the superficial gas velocities on the air recovery in cell 3
$K_{\alpha_{Jg4}}$	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 $i = 1$

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.

The water recovery model for the flotation circuit is modelled as,

$$\frac{Q_{C_k}}{A_k} = \begin{cases} \frac{18.54\mu C_{PB} J_{g_k}^2}{\rho g D_{BF_k}^2}(1-\alpha_k)\alpha_k & 0 < \alpha_k < 0.5 \\ \frac{18.54\mu C_{PB} J_{g_k}^2}{4\rho g D_{BF_k}^2} & \alpha_k \geqslant 0.5 \end{cases} \tag{22}$$

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,

$$J_{g_k} = 100\frac{Q_{air_k}}{A_k}. \tag{23}$$

The masses for the concentrate are calculated from entrainment and true flotation models. The entrainment model defines the concentrate mass flow-rate as [44],

$$\dot{M}_{C_k}^i = \frac{K^i M_k^i J_{g_k}\alpha_k}{D_{BP_k}} + Ent_{Frac}^i \frac{M_k^i}{A_k L_k} Q_{C_k} \tag{24}$$

where K^i is the flotation rate constant for material class i. The entrainment factor is,

$$Ent_{Frac}^i = \frac{\ln\left(d_{ptr}^i\right) - \ln\left(d_{pmin}\right)}{\ln\left(d_{pmax}\right) - \ln\left(d_{pmin}\right)} \tag{25}$$

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

$$d_{ptr}^i = \sqrt[3]{\frac{\ln(0.5) J_{g_k}{}^2}{K_{ent}^i h_{f_k}}} \tag{26}$$

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

$$K_{ent}^i = \left[\frac{1}{3}\frac{g\left(\rho_s^i - \rho\right)}{18\mu}\right]^{1.5} \frac{\sqrt{\frac{\rho g}{3\mu C_{PB}}(\sqrt{3} - \pi/2)P_e}}{\sqrt{\alpha_k\left(1-\alpha_k\right)}}, \tag{27}$$

where ρ_s^i is the solids particle density of the material class i. The state-space equation for the hopper is,

$$\begin{aligned} \dot{L}_H &= \frac{Q_{C_1} + Q_{C_2} + \cdots + Q_{C_N} - Q_H}{A_H} \\ \dot{M}_H^i &= \sum_{k=1}^{N} \dot{M}_{C_k}^i - \frac{M_H^i}{L_H A_H} Q_H \end{aligned} \tag{28}$$

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,

$$Grade_H = \frac{M_{D_H}^0}{\sum_{i=0}^{n} M_H^i} \tag{29}$$

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

3.2.3. 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. 6 [45]. 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. 6 the following controller is implemented using NMPC. The lead controller is used to compare constrained MPSP to NMPC in simulation.

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 [45,46].

The nonlinear state-space description of the flotation circuit in (21) is simulated using the Runge–Kutta fourth-order method.

All model states are observable and can be estimated using an appropriate observer [43]. Since the observer design falls outside the scope of this study, full-state feedback is assumed.

The initial state conditions of each cell are,

$$\begin{aligned}
X_{c1}^0 &= \left[D_{BF_1}, \alpha_1, L_1, M_1^0, M_1^1\right]^T \\
&= [1.80, 0.41, 1.30, 53.87, 7007]^T, \\
X_{c2}^0 &= \left[D_{BF_2}, \alpha_2, L_2, M_2^0, M_2^1\right]^T \\
&= [1.45, 0.27, 1.30, 29.58, 7025]^T, \\
X_{c3}^0 &= \left[D_{BF_3}, \alpha_3, L_3, M_3^0, M_3^1\right]^T \\
&= [1.66, 0.17, 1.30, 19.84, 7032]^T, \\
X_{c4}^0 &= \left[D_{BF_4}, \alpha_4, L_4, M_4^0, M_4^1\right]^T \\
&= [1.40, 0.17, 1.30, 13.44, 7027]^T, \\
X_H^0 &= \left[L_H, M_H^0, M_H^1\right]^T \\
&= [1.40, 97.62, 7.31]^T.
\end{aligned} \tag{30}$$

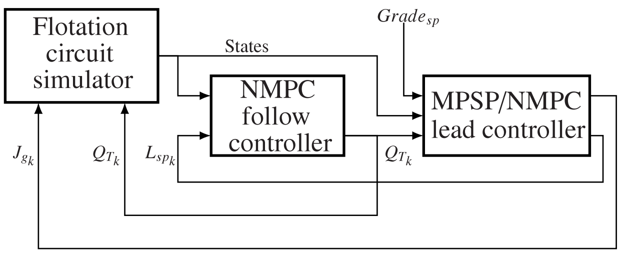

Fig. 6. Lead-follow controller configuration for the flotation circuit.

The initial input conditions of the following and leading controllers are,

$$\begin{aligned}
U_{follow}^0 &= \left[Q_{T_1},\ Q_{T_2},\ Q_{T_3},\ Q_{T_4},\ Q_H\right]^T \\
&= [728.38,\ 727.62,\ 727.29,\ 727.01,\ 3.00]^T, \\
U_{lead}^0 &= \left[J_{g1},\ J_{g2},\ J_{g3},\ J_{g4},\right. \\
&\qquad \left. L_1,\ L_2,\ L_3,\ L_4,\ L_H\right]^T \\
&= [8.4,\ 8.5,\ 8.1,\ 7.7, \\
&\qquad 1.30,\ 1.30,\ 1.30,\ 1.30, 1.00]^T,
\end{aligned} \tag{31}$$

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

The input ranges for the simulations are,

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

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.

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 $N_B = 5$ [12]. Warm-starting is used for the initial guess input of the optimization problem. The weighting matrices for the NMPC follow controller are chosen as,

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

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}.

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,

$$Q_{mpsp} = Q_{Grade} = 100$$
$$R^{lead}_{mpsp} = \mathrm{diag}([R_{Jg_1},\ R_{Jg_2},\ R_{Jg_3},\ R_{Jg_4},\ R_{L1_{sp}}, R_{L2_{sp}},\ R_{L3_{sp}},\ R_{L4_{sp}},\ R_{LH_{sp}}]),$$

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 executed 10 times, or if the following conditions are met,

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

The same strictly convex QP solver that was used in the grinding mill circuit simulation is used for the flotation circuit simulation.

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 (12) are chosen as,

$$Q_{nmpc} = Q_{Grade} = 100$$
$$R^{lead}_{nmpc} = \mathrm{diag}([R_{Jg_1},\ R_{Jg_2},\ R_{Jg_3},\ R_{Jg_4},\ R_{L1_{sp}}, R_{L2_{sp}},\ R_{L3_{sp}},\ R_{L4_{sp}},\ R_{LH_{sp}}])$$

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 executed a maximum of 10 times, or the algorithm converged between iterations within a tolerance of 1×10^{-4}.

Simulation results and discussion

All the simulation results of the NMPC and the MPSP controllers are shown in Figs. 7 to 11. The grade and recovery of the flotation circuit are shown in Fig. 7. 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. 7 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 [45]. NMPC settles at a higher recovery than MPSP.

The cell level results are shown in Fig. 8, 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. 8 in conjunction with Fig. 9 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 of the disturbance. This causes the MPSP controller to have better state constraint performance than the NMPC controller.

Fig. 10 shows the froth bubble diameter size D_{BF_k} of each cell k. Both the MPSP and the NMPC controllers can reject the state

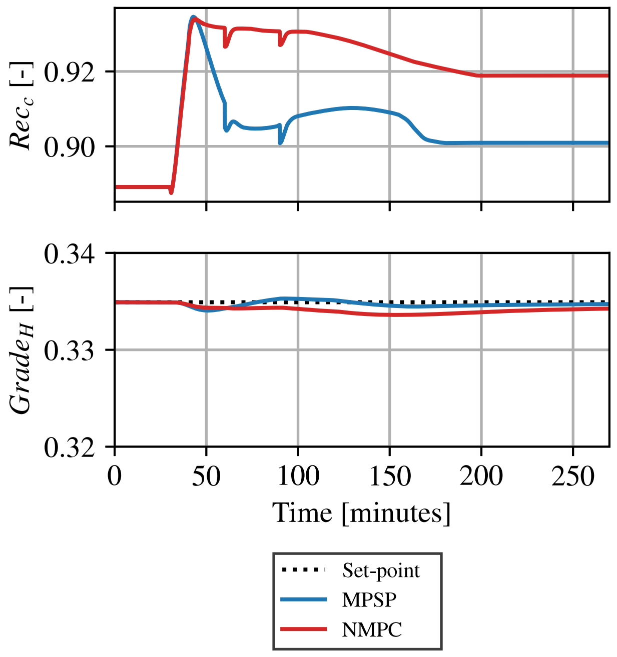

Fig. 7. Grade and recovery of the flotation circuit.

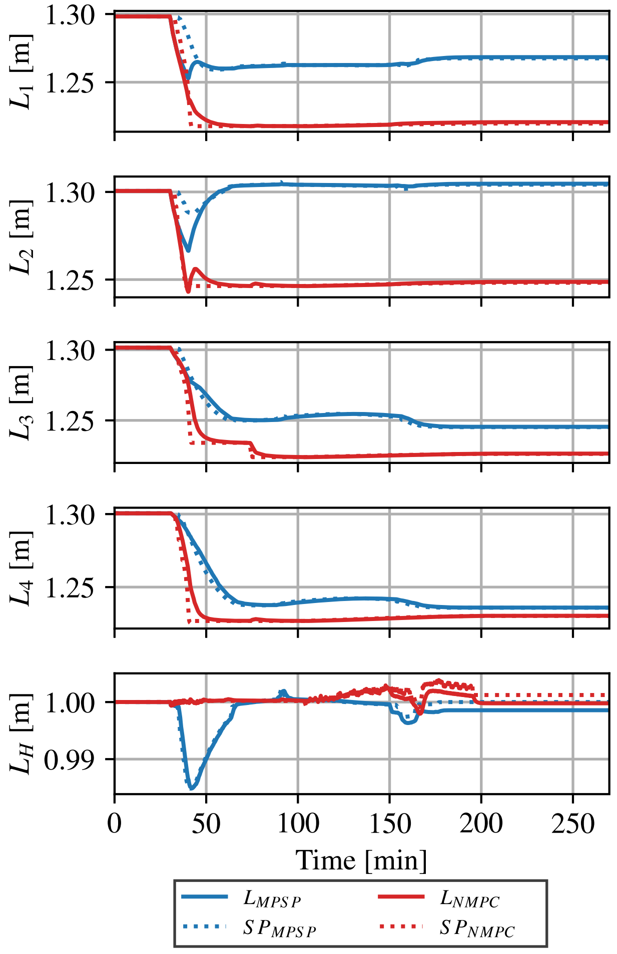

Fig. 8. Flotation cell levels. Legend: L_{NMPC} is the output level for the NMPC controller, L_{MPSP} is the output levels of the MPSP controller, SP_{NMPC} is the input set-point of the NMPC controller to the following controller and SP_{MPSP} is the input set-point of the MPSP controller to the following controller. The nominal conditions of each cell is shown in (30).

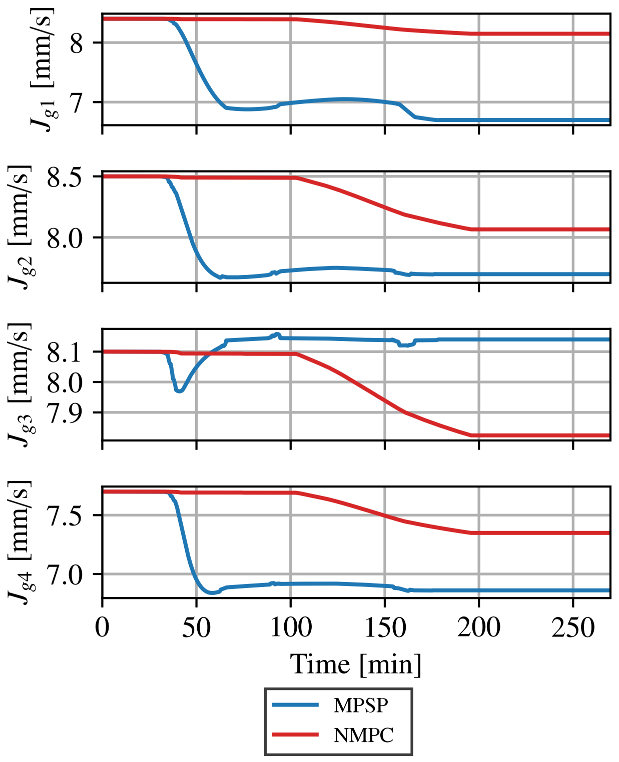

Fig. 9. Superficial gas velocity. The nominal conditions of each cell is shown in (31).

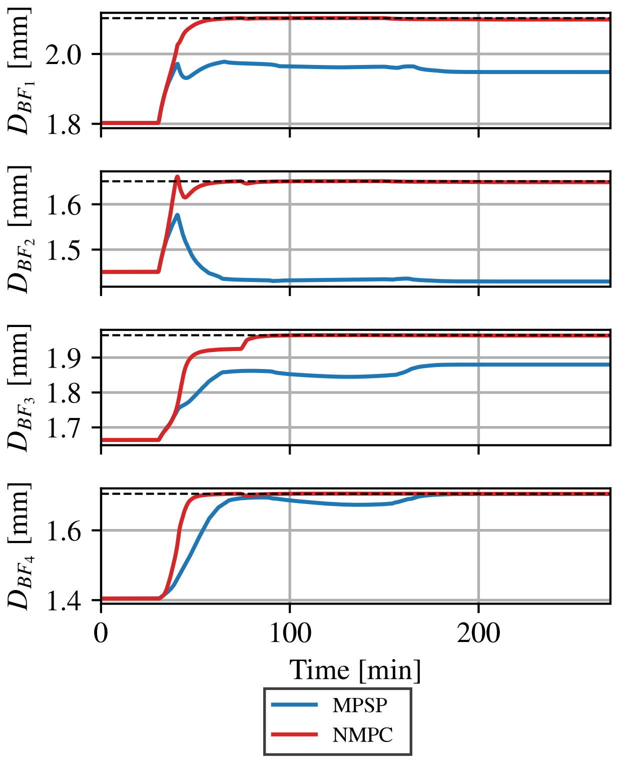

Fig. 10. Froth bubble size diameter of each cell. The nominal conditions of each cell is shown in (30).

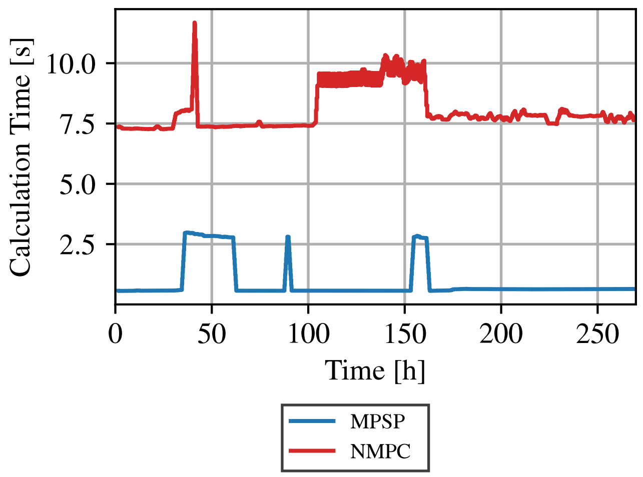

Fig. 11. The time necessary to calculate a new input of the controllers for the flotation circuit.

Table 6
Iteration time and performance results of the MPSP and the NMPC simulations for the flotation circuit.

Controller	$\bar{x}$ [s]	σ [s]	Max [s]	IAE($Grade_H$)
NMPC	8.029	1.419	13.88	12.227
MPSP	0.907	0.773	3.060	4.423

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 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 for the lead controllers are shown in Fig. 11 and Table 6. The MPSP controller is on average 8.85 times faster than the NMPC 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.

The sampling time for each controller is $T_s = 10$ s. As shown in Fig. 11, whereas the NMPC controller is able to calculate a new input within 7.5 s on average, 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. Fig. 11 indicates that the MPSP algorithm is more computationally efficient.

Industrial flotation circuits often run at a sampling rate of $T_s = 60$ s, even though this may be too slow to capture all the dynamics of the process [45,46]. 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.

4. Conclusion

In the case of the grinding mill circuit example in Section 3.1 none of the disturbances were known. Both the MPSP and NMPC algorithms were able to stay within the constraints of the plant. The MPSP controller which uses the QP problem solver is computationally faster than the NMPC controllers.

In the case of the flotation circuit example in Section 3.2 all the disturbances were measured. Both the MPSP and NMPC algorithms were able to accurately predict the state and output trajectory. None of the state constraints were violated for either controller. The MPSP controller calculated a new control move faster than the NMPC controller for each time-step in the simulation. The number of optimization variables increased significantly between the two case studies which does not lead to a significant increase of computational time for the MPSP controller.

As shown in this paper, constrained MPSP, as applied in simulation to a grinding mill circuit and a four-cell flotation, has similar performance but a faster computational time compared to an NMPC controller. Even though MPSP operates on the philosophy of MPC, it reduces the original problem into a lower-dimensional problem of control variables alone, thereby enhancing computational efficiency significantly. Because of this, problems with larger dimensions and/or increased complexity can be solved using MPSP without changing the computational infrastructure. Hence, it can be a preferred fast MPC approach for mineral processing plants. Potential for future work is to evaluate constrained MPSP to other fast MPC approaches [47].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

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

References

[1] D.Q. Mayne, Model predictive control: Recent developments and future promise, Automatica 50 (12) (2014) 2967–2986.
[2] P.S.G. Cisneros, S. Voss, H. Werner, Efficient nonlinear model predictive control via quasi-LPV representation, in: 2016 IEEE 55th Conference on Decision and Control, CDC, 2016, pp. 3216–3221.
[3] A. Bemporad, M. Morari, V. Dua, E.N. Pistikopoulos, The explicit linear quadratic regulator for constrained systems, Automatica 38 (1) (2002) 3–20.
[4] M. Schwenzer, M. Ay, T. Bergs, D. Abel, Review on model predictive control: An engineering perspective, Int. J. Adv. Manuf. Technol. 117 (5–6) (2021) 1327–1349.
[5] L.C. Coetzee, I.K. Craig, E.C. Kerrigan, Robust nonlinear model predictive control of a run-of-mine ore milling circuit, IEEE Trans. Control Syst. Technol. 18 (1) (2010) 222–229.
[6] D. Kouzoupis, R. Quirynen, J.V. Frasch, M. Diehl, Block condensing for fast nonlinear MPC with the dual Newton strategy, IFAC-PapersOnLine 48 (23) (2015) 26–31.
[7] I.J. Wolf, W. Marquardt, Fast NMPC schemes for regulatory and economic NMPC – a review, J. Process Control 44 (2016) 162–183.
[8] M. Diehl, H.G. Bock, J.P. Schlöder, A real-time iteration scheme for nonlinear optimization in optimal feedback control, SIAM J. Control Optim. 43 (5) (2005) 1714–1736.
[9] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, M. Diehl, From linear to nonlinear MPC: bridging the gap via the real-time iteration, Internat. J. Control 93 (1) (2020) 62–80.
[10] V.M. Zavala, L.T. Biegler, The advanced-step NMPC controller: Optimality, stability and robustness, Automatica 45 (1) (2009) 86–93.
[11] E.N. Pistikopoulos, Perspectives in multiparametric programming and explicit model predictive control, AICHE J. 55 (8) (2009) 1918–1925.
[12] Y. Wang, S. Boyd, Fast model predictive control using online optimization, IEEE Trans. Control Syst. Technol. 18 (2) (2010) 267–278.
[13] S. Chen, K. Saulnier, N. Atanasov, D.D. Lee, V. Kumar, G.J. Pappas, M. Morari, Approximating explicit model predictive control using constrained neural networks, in: 2018 Annual American Control Conference, ACC, 2018, pp. 1520–1527.
[14] X. Zhang, M. Bujarbaruah, F. Borrelli, Near-optimal rapid MPC using neural networks: A primal-dual policy learning framework, IEEE Trans. Control Syst. Technol. 29 (5) (2021) 2102–2114.
[15] R. Cagienard, P. Grieder, E.C. Kerrigan, M. Morari, Move blocking strategies in receding horizon control, J. Process Control 17 (6) (2007) 563–570.
[16] M. Faroni, M. Beschi, M. Berenguel, A. Visioli, Fast MPC with staircase parametrization of the inputs: Continuous input blocking, in: 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, 2017, pp. 1–8.
[17] S.J. Wright, Applying New Optimization Algorithms to More Predictive Control, Technical Report, Argonne National Lab.(ANL), Argonne, IL (United States), 1996.
[18] G. Hou, L. Gong, C. Huang, J. Zhang, Fuzzy modeling and fast model predictive control of gas turbine system, Energy 200 (2020) 117465.
[19] K. Kunz, S.M. Huck, T.H. Summers, Fast model predictive control of miniature helicopters, in: 2013 European Control Conference, ECC, 2013, pp. 1377–1382.
[20] R. Padhi, M. Kothari, Model predictive static programming: A computationally efficient technique for suboptimal control design, Int. J. Innov. Comput. Inf. Control 5 (2) (2009) 399–411.
[21] P. Kumar, R. Padhi, Extension of model predictive static programming for reference command tracking, IFAC Proc. Vol. 47 (2014) 855–861.
[22] N.G. Bhitre, R. Padhi, State constrained model predictive static programming: A slack variable approach, IFAC Proc. Vol. 47 (1) (2014) 832–839.
[23] Y. Li, H. Zhou, W. Chen, Three-dimensional impact time and angle control guidance based on MPSP, Int. J. Aerosp. Eng. 2019 (2019) 1–16.
[24] P. Kumar, B.B. Anoohya, R. Padhi, Model predictive static programming for optimal command tracking: A fast model predictive control paradigm, J. Dyn. Syst. Meas. Control 141 (2019) 021014.
[25] D. Biswas, S. Ghosh, S. Sengupta, S. Mukhopadhyay, Energy management of a parallel hybrid electric vehicle using model predictive static programming, Energy 250 (2022) 123505.
[26] A.K. Tripathi, R. Padhi, Autonomous landing for UAVs using T-MPSP guidance and dynamic inversion autopilot, IFAC-PapersOnLine 49 (1) (2016) 18–23.
[27] B. Zhang, S. Tang, B. Pan, Multi-constrained suboptimal powered descent guidance for lunar pinpoint soft landing, Aerosp. Sci. Technol. 48 (2016) 203–213.
[28] F. Bin, G. Hang, C. Kang, W. Xingyu, Y. Jie, Aero-thermal heating constrained midcourse guidance using state-constrained model predictive static programming method, J. Syst. Eng. Electron. 29 (6) (2018) 1263.
[29] H. Hong, A. Maity, F. Holzapfel, S. Tang, Model predictive convex programming for constrained vehicle guidance, IEEE Trans. Aerosp. Electron. Syst. 55 (5) (2019) 2487–2500.
[30] Y. Wang, H. Hong, S. Tang, Geometric control with model predictive static programming on SO(3), ACTA Astronaut. 159 (2019) 471–479.
[31] J.D. Le Roux, R. Padhi, I.K. Craig, Optimal control of grinding mill circuit using model predictive static programming: A new nonlinear MPC paradigm, J. Process Control 24 (12) (2014) 29–40.
[32] Z.M. Noome, J.D. Le Roux, Controlling a grinding mill circuit using constrained model predictive static programming, IFAC-PapersOnLine 55 (21) (2022) 49–54.
[33] D. Goldfarb, A. Idnani, A numerically stable dual method for solving strictly convex quadratric programs, Math. Program. 27 (1983) 1–33.
[34] J. Nocedal, S.J. Wright, Numerical Optimization, second ed., Springer, 2006.
[35] S. Mizuno, Infeasible-interior-point algorithms, in: T. Terlaky (Ed.), Interior Point Methods of Mathematical Programming, Springer US, Boston, MA, 1996, pp. 159–187.
[36] B.A. Turlach, A. Weingessel, quadprog: Functions to solve quadratic programming problems, 2011, R package version 1.5-4.
[37] N.S. Nise, Control Systems Engineering, John Wiley & Sons, 2020, pp. 136–141.
[38] D. Kraft, A Software Package for Sequential Quadratic Programming, DFVLR-FB 88-28, 1988, pp. 24–25.
[39] J.D. Le Roux, C.W. Steyn, Validation of a dynamic nonlinear grinding circuit model for process control, Miner. Eng. 187 (2022) 107780.
[40] J.D. Le Roux, I.K. Craig, D.G. Hulbert, A. Hinde, Analysis and validation of a run-of-mine ore grinding mill circuit model for process control, Miner. Eng. 43–44 (2013) 121–134.

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

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

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

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

[45] D.J. Oosthuizen, A Dynamic Flotation Model for Online Estimation, Control and Optimisation (Ph.D. thesis), University of Pretoria, 2023.

[46] K. Hadler, C. Smith, J. Cilliers, Flotation performance improvement by air recovery optimisation on roughers and scavengers, in: XXV International Mineral Processing Congress 2010, Vol. 3, IMPC 2010, 2010, pp. 1917–1924.

[47] M. Diehl, H.J. Ferreau, N. Haverbeke, Efficient numerical methods for nonlinear MPC and moving horizon estimation, in: L. Magni, D.M. Raimondo, F. Allgöwer (Eds.), Nonlinear Model Predictive Control: Towards New Challenging Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 391–417.