Development of a model predictive controller for a milling circuit

by B. Muller and P.L. de Vaal*

Synopsis

Milling circuits are known to be difficult to control. In order to address these difficulties, it was decided to develop and test a model predictive controller (MPC) for a milling circuit. A milling simulator was written using Microsoft Visual C++ and used to design and test the MPC controller. Due to the large number of tuning parameters in a controller of this type, an optimizing algorithm was written that automatically tunes the controller. Good, robust control was obtained when the controller was tuned using this algorithm. Control was further improved by combining feedback and feedforward control and by using quadratic programming in the MPC algorithm. The movement of the manipulated variables was reduced by using a novel technique that allows the controlled variable to vary within a specified region around the setpoint. An optimiser controller was also successfully implemented to control the circuit at its maximum throughput.

Introduction

The control of a milling circuit is a difficult problem due to a number of factors. Among them are the multivariable, interactive nature of the system, the widely differing dynamics and the fact that the process is very non-linear and difficult to model. The aim of this paper is to design and test a model predictive controller for a milling circuit. Problem areas were identified and the model predictive controller adapted to try and accommodate these difficulties. Since the throughput of the milling circuit is a parabolic function of the mill load, an optimizer controller was also written to control the milling circuit at the maximum throughput.

The milling circuit

Milling circuits are used to grind mining material to a fine product. The particle size that leaves this circuit is a measure of the *quality* of the product produced by the milling circuit. This quality is important, because it determines the ability of downstream processes to extract maximum possible benefit (for example gold recovery from the ore). A milling circuit will operate in an optimized manner when it produces *high quality* product at the maximum possible rate and efficiency (Craig *et al.*1992). The first and primary objective of a controller will therefore be to control the circuit in such a way that the required setpoint for the particle size is achieved. Another control objective will be to maximize the throughput of the milling circuit.

In order to design a controller, the dynamics of the process have to be investigated. A milling circuit is an interactive multivariable process (Craig and Hulbert1986). This means that there are a number of input and output variables that affect each other. A simplified diagram of a typical milling circuit is shown in Figure 1. This specific circuit has six plant inputs (u) and five plant outputs (y).

Often, no distinction is made between the feed of the fine material (u_3) and the feed of the coarse material (u_4). When these two inputs are combined, the plant will have only five inputs.

Circuit dynamics

The throughput of the milling circuit is a non-linear function. When plotted against the load in the mill (mass inside the mill), a parabolic function is obtained with a maximum throughput near a load of 70%. When the power used by the mill is plotted against the mill load, more or less the same parabolic function with a maximum near a load of 70% is obtained. The power used by the mill is therefore often used as an indication of the throughput of the mill (Craig *et al.*1992). The response of the power can be approximated as a parabolic function of the load according to the following Equation:

* *Department of Chemical Engineering, University of Pretoria, Pretoria 0002.*

 SA ISSN 0038–223X/3.00 + 0.00. Paper received Sep. 2000; revised paper received Oct. 2000.

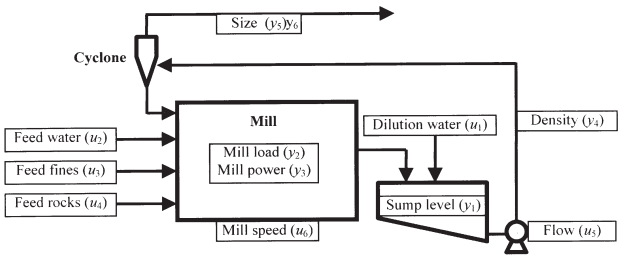

Figure 1—Milling circuit

$$Power = Power_{\max} - \alpha\left(Load_{base} - Load\right)^2 \qquad [1]$$

The problem with this relationship is that it is not fixed. The power/load relationship is only on average a parabolic function. When looking at the individual points, it may deviate substantially from the parabolic function. The load where the power reaches its optimum ($Load_{base}$) and the maximum power ($Power_{max}$) may also shift around over time. This is caused by disturbances such as changing feed-ore size, changing feed composition and other unmeasured dynamics. The power (and therefore also the throughput) can be controlled by using an optimizer controller that changes the setpoint of the load (which is controlled by the MPC controller) such that the power is kept at its optimum value. A block flow diagram for this kind of cascade control is given in Figure 2.

Difficulties in controlling a milling circuit

The control of a milling circuit is made difficult due to the following characteristics:

- Interaction between the inputs and the outputs, since it is a multivariable process
- The process has widely differing time constants
- Large dead times for certain input-output pairs
- The system contains a number of integrators
- It is a non-linear process with many disturbances and unmodelled dynamics
- The process changes with time as the circuit ages
- The throughput is a parabolic function of the load and should be controlled at its maximum.

Since model predictive control can address most of these difficulties quite well, it was decided to design and test a model predictive controller for a milling circuit.

Model predictive control

Model predictive control (MPC) uses a dynamic model of the plant to predict the future actions of the manipulated variables on the plant output. The future moves of the manipulated variables can then be determined by minimizing the difference between the setpoint and the predicted output. These optimization calculations are performed during each time step using the latest measurements. Only the first calculated move of the manipulated variable is used, since at each time step the manipulated variables are recalculated.

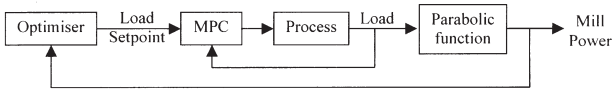

Figure 2—Control of mill power

The algorithm executed during each time step n by the MPC is as follows (Seborg *et al.*1989):

- The present and future control actions ($m_n, \ldots m_{n+k}$) are determined by minimizing the difference between the predicted process output ($\hat{c}_{n+1}$) *and the target* (y_{sp}). *A* quadratic error function is normally used
- Only the first computed control action (m_n) is implemented
- At time n+1 a new measurement becomes available and the whole procedure is repeated.

Step response model

A typical response of a system to a unit step change in the input (m) is shown in Figure 3. The system had an initial value of c_0 and the step change was applied at time $t = 0$.

In general, the change in the input of a system can be expressed as a sequence of step changes ($\Delta m_i = m_i - m_{i-1}$). The step response coefficients are then used to calculate the changes in the outputs due to these input step changes (Seborg *et al.*1989).

Basic model predictive control algorithm

The algorithm will be introduced by looking at the situation encountered each time the model predictive feedback controller is executed. This time step will be called n (see Figure 4). It is assumed that the manipulated variable has been changed in the past, influencing the controlled variable. The current value of the controlled variable ($\hat{c}_n$) can be calculated using the past changes of the manipulated variable (Δ_m) by making use of the step response model. This value is compared to the current measured value of the controlled variable (c_m) to determine the feedback error. The future values of the controlled variables (c_j^f) as influenced by the past changes in the manipulated variables can also be

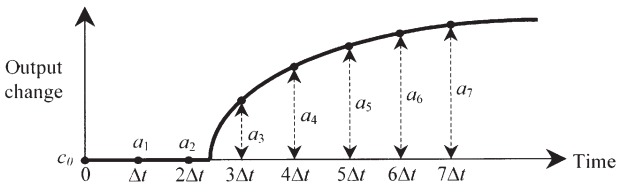

Figure 3—Step response coefficients

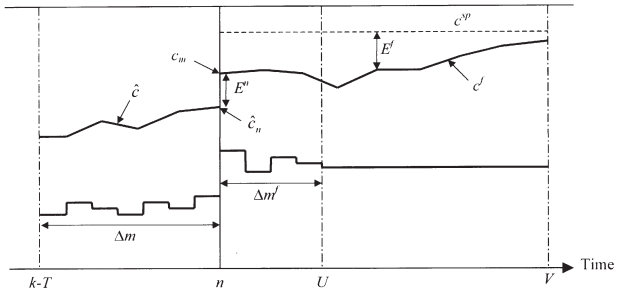

Figure 4 — Dynamic response of manipulated and controlled variables

calculated. The error is assumed to remain constant and therefore added at each future time instant.

The future changes in the controlled variables ($\hat{c}$) due to future moves in the manipulated variables (Δm^f) can also be calculated. In MPC, the number of future control actions or control moves that will be calculated during each time step, is called the control horizon (U). The number of future outputs (controlled variables) that will be calculated using the control horizon is called the prediction horizon (V). At each time step n the next U values of the manipulated variable (m) are calculated as well as the next V output directions ($\hat{c}$) caused by the future calculated changes of m.

The objective of the controller is now to minimize the difference between the future controlled variables (due to past and future changes in the manipulated variables) and the setpoint (closed loop error). If the future setpoints are not known, they are to be assumed the same as the current setpoint. The sum of the errors squared objective function is used:

$$OBJ = \sum_{j=1}^{V}\left[c_j^{sp} - \left(c_j^f + \alpha_j^f\right)\right]^2 \qquad [2]$$

The objective function can now be minimized by changing the future manipulated variables (Δm^f), which will only influence the value of $\hat{c}$. An optimum solution can be obtained using the least-squares method. This method minimizes the objective function by differentiating with respect to Δm^f and setting it equal to zero.

The objective function can be expanded by adding the square of the changes in the manipulated variables. This will penalize the movement of the manipulated variables. The two terms in the objective function can furthermore be multiplied with a weighting matrix. The relative values of these two tuning parameters (weighting matrices) will determine the importance placed on the controlled variable and the variability of the manipulated variable. The objective function will now look as follows:

$$OBJ = \sum_{j=1}^{V} W_{1,j}\left[c_j^{sp} - \left(c_j^f + \alpha_j^f\right)\right]^2 + \sum_{i=1}^{U} W_{2,j}\left[\Delta m_i^f\right]^2 \qquad [3]$$

During each time step, only the first control action (Δm_1^f) is implemented. The next output (c_{n+1}) is then observed, the predictions are corrected and the optimization algorithm is applied again. A problem that may occur, especially when the A^TA matrix is ill-conditioned, is that excessively large changes in the manipulated variable may result. This problem can be overcome by modifying the weighting matrices (W).

Tuning of a model predictive controller

The following design parameters can be adjusted to tune a model predictive controller:

- T: Model horizon (one for each controlled output)
- U: Control horizon (one for each manipulated input)
- V: Prediction horizon (one for each controlled output)
- W_1: Weighting matrix for predicted errors (one for each controlled output)
- W_2: Weighting matrix for control moves (one for each manipulated input)
- Δt: Sampling period.

There exists a number of rules according to which initial parameters can be chosen (Prett and Garcia[1988]). Since there is such a large number of parameters, it is best to write an optimization function that automatically tunes the controller. This can be done by simulating the circuit while being controlled and applying step changes to the setpoint of each controller at specific times. The deviation from the setpoints (error) can be calculated using for instance the integral squared error. The parameters can now be changed iteratively to minimize this error.

Process constraints

Almost all practical control problems have constraints on both the manipulated and controlled variables. This is due to physical limitations of the plant equipment. These constraints have to be incorporated in the control strategy to ensure effective control. One way of dealing with process constraints is to adjust the tuning parameters. The selection of tuning parameters will then be an iterative procedure until all the constraints are met. This procedure, however, is not always satisfactory. A more direct approach is to modify the optimisation problem by adding the constraints explicitly. An example of such an approach is quadratic programming, which will ensure that the input and output variables stay within their limits at all times.

Quadratic programming

In real processes, the computed values of the manipulated variables may not be achievable due to physical constraints or limits on the variables. The limits imposed on the controlled variables may also be violated. Three kinds of process constraints are usually encountered (Garcia and Morshedi[1986]):

- Manipulated variable constraints (valve saturation)
- Controlled variable constraints (overshoots past allowable limits must be avoided)
- Associated variables (key process variables not directly controlled, but that must be kept within bounds).

Ideally, the model predictive controller should predict violations and prescribe moves that would keep these variables within bounds. The quadratic objective function includes these limits on the inputs and the outputs in the optimization algorithm to explicitly ensure that all the variables are kept within bounds. The Karush-Kuhn-Tucker conditions (Bazaraa *et al.*[1993]) are used to include the limitations on the variables in the optimization function.

Tuning of a quadratic controller

When using quadratic programming, some more tuning parameters exist in addition to the usual ones described earlier. The projection interval that should be constrained can furthermore be chosen. In practice, only a subset of the V projections are constrained, starting with the lth projection, where $l \geq 1$. This subset of future projections forms a 'constraint window' over which the quadratic controller will prevent constraint violations from occurring. Performance is often improved by moving the 'constraint window' further down the horizon (Garcia and Morshedi[1986]). The reason for this is that any projection violation is handled rigorously by the quadratic program. Severe input moves in the face of non-minimum phase characteristics are sometimes required to correct for violations in the earlier projections.

Developing the MPC-controller

As a proof-of-concept step, a simulator was written which could be used to design an application of the MPC controller. The simulator uses transfer functions to generate the output changes from input changes. Each input is controlled by a local PI-controller. Upper and lower limits/constraints, noise and disturbances can also be added to both the inputs and the outputs and they can be made to drift over time. Non-linearities such as non-linear valve characteristics and valve hysteresis were also included in the simulator. The model used for the simulator could also be made to differ from the model used by the MPC-controller. This would represent model errors and can be used to test the robustness of the controller. The parameters of the model can also be changed while simulating to represent changes in the model over time. The resulting simulator provides a very good representation of a real milling circuit. This makes it possible to design an MPC-controller for any standard milling circuit if transfer functions for the process are available.

The milling circuit simulator and MPC controller were written in Microsoft Visual C++. The most important output variable that needs to be controlled in a milling circuit is the particle size that leaves the cyclone. The throughput is another important variable and needs to be maximized. This is done by maximizing the power consumption of the mill. The other output variables, however, are also important to control, since they will influence the particle size and the power consumption of the mill.

Initially only the *Particle size*, the *Sump level* and the *Mill load* were chosen as controlled variables. The *Dilution water*, *Feed rocks* and *Flow to cyclone* were chosen as the manipulated variables. The *Feed water* and *Mill speed* were treated as disturbance variables.

There are a large number of tuning parameters for the three input and three output variables, since the horizons of each input and output can be specified separately. If the weighting matrices are seen as diagonal matrices with the same diagonal elements for each input-output pair, there are a total of 16 tuning parameters (3 model horizons, 3 control horizons, 3 prediction horizons, 3 input weighting elements, 3 output weighting elements and the time step for the controller). An optimizing algorithm was therefore written to automatically find the best parameters for the controller.

The controller was first tuned for an ideal process with no limits on the inputs or the outputs. The least squares optimization method was used. Good control was obtained, but the control deteriorated when constraints were added, since the movement of the manipulated variables violated these constraints. The controller was retuned with the constraints in place and the performance was improved dramatically.

Next, drift rates to the base levels of the variables were added, as well as some non-linear factors (e.g. valve characteristics). It was found that the drifting of one of the disturbance variables caused some of the controlled variables, like the sump level, to drift away from the setpoint. This was due to the integral action of this disturbance variable on the sump level. This problem was solved in three ways. The first method is to assume that part of the error is caused by integral action instead of assuming a constant error. The fraction of the error that is assumed to be caused by integral action is then treated as another tuning parameter. It was found that the control was improved most when it was assumed that the whole error on the *Sump level* was caused by integral action. The second method to handle the integrating disturbances was to use feedforward control on that disturbance variable. This removed the problem (see Figure 5), but due to the nature of feedforward control it is highly dependent on the accuracy of the model. The third solution was to use that specific disturbance vector as another manipulated variable. Since MPC uses an optimization function for calculating the values of the manipulated variables, the number of manipulated and controlled variables do not need to correspond. This method did not improve control as much as the previous two methods, but it resulted in a more robust controller.

The MPC controller was next compared to a normal PI-controller to give an indication of how well the controller performs. A simple single input, single output PI-controller was designed for comparative purposes. From Figure 6 it is clear that performance of the MPC controller on the particle size was much better than the PI-controller, since the PI-controller showed more interaction than the MPC controller. This interaction can be eliminated using more advanced PI-control techniques. This was done by Hulbert *et al.*1990 by using the inverse Nyquist array technique. This technique eliminated most of the interaction, but at the expense of poor control on the *Sump level*. The MPC controller, when compared to the literature (Hulbert *et al.*1990), therefore showed better control than the advanced PI-control techniques.

The controller was tested further by adding noise to the inputs and the outputs. This did not deteriorate the performance of the controller much when looking at the outputs, but it did cause excessive movement in the manipulated variables, since the controller attempted to control the noise. This was overcome by incorporating a variable region of uncertainty around the setpoint (see Figure 7). The controlled variable is allowed to vary within this region. As long as the controlled variable stays within this region, the error is set to zero and no control action is taken. If the controlled variable moves out of this region, the error is set equal to the distance to the nearest boundary and the control action is calculated accordingly. This reduced the movement of the manipulated variables substantially (see Figure 8) and ensured that the controller did not try to control the noise of the plant.

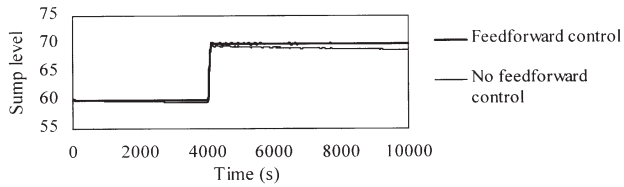

Figure 5—Control with and without feedforward control on *Feed water*

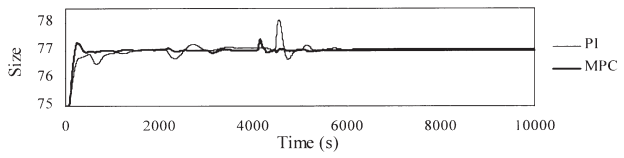

Figure 6—MPC versus PI-control

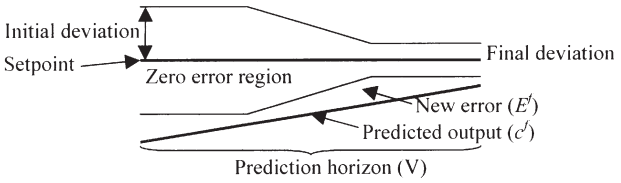

Figure 7—Region of uncertainty

The robustness of the controller was tested by incorporating a 20% as well as a 50% modelling error. This was done by increasing the model parameters of the model used by the controller first by 20% and then by 50%. The model used by the simulator was therefore now different from the model used by the controller. The controller performance deteriorated visibly, as can be seen from Figure 9. It was found, however, that the deviations from the setpoint were of the same order as the noise. This showed that the controller still performed reasonably well, even with a 50% modelling error. When the region of uncertainty was used with a 50% modelling error, control was further improved. This region of uncertainty therefore also increases the robustness of the controller.

Next, a controller was tuned using quadratic programming. This time the controller could be tuned without the limits, since quadratic programming incorporates the limits in the optimization function. Good control was obtained, but the calculation time was much longer compared with using the least squares method. Figure 10 shows the difference in the performance between the least squares method and quadratic programming when the same control parameters were used.

An optimizer controller was written to control the power of the mill at its optimum, since it is a parabolic function of the mill load. This optimizer changed the setpoint of the mill load either up or down by a certain amount and recorded the change in the power. Average values over the time interval of the optimizer were used to eliminate the effect of noise. When the power increased, the mill load setpoint continued to increase in that direction. When the power decreases, the mill load setpoint is changed in the opposite direction. This worked quite well and the results are shown in Figure 11. It is clear that the setpoint of the mill load is continuously changed to keep the mill power at the maximum value (the mill power increased steadily because the simulator was chosen such that the maximum power increases over time).

Conclusions and recommendations

It was found that good, robust control could be obtained for a milling circuit using model predictive control. The constraints on the inputs and outputs adversely affected the control. It is therefore recommended that the controller be tuned with the constraints in place while the maximum expected setpoint changes are made to the process. This will ensure that the process will be able to handle setpoint changes up to these expected values without violating the constraints. Another method will be to use quadratic programming, which has been shown to perform better than the least squares methods.

Control was further improved by using feedforward control, although this will only work well as long as the feedforward model is accurate. Noise was handled by using a variable region of uncertainty. This mechanism successfully prevented excessive movement of the manipulated variables. Lastly, an optimizer was successfully implemented that controlled the milling circuit at maximum power consumption and therefore at maximum throughput by changing the setpoint of the mill load.

Acknowledgement

This research was made possible by a sponsorship and co-operation from Mintek, which is gratefully acknowledged.

References

BAZARAA, M.S., SHERALI, H.D., and SHETTY, C.M. *Nonlinear Programming Theory and Algorithms*, Second Edition, John Wiley & Sons, Inc., New York. 1993.

CRAIG, I.K. and HULBERT, D.G. The Transfer Function Matrix of the run-of-mine (R.O.M.) Milling Circuit at Vaal Reefs 7 Shaft, Clarified in terms of Actual Circuit Dynamics, Mintek report, no. C640M 1986.

CRAIG, I.K., HULBERT, D.G., METZNER, G., and MOULT, S.P. Optimized multivariable control of an industrial run-of-mine milling circuit, *Journal of the South African Institute of Mining and Metallurgy*, vol. 92, no. 6, 1992. pp. 169–176.

GARCIA, C.E. and MORSHEDI, A.M. Quadratic Programming solution of Dynamic Matrix Control (QDMC), *Chemical Engineering Communications*, vol. 46, 1986. pp. 73–87.

HULBERT, D.G., CRAIG, I.K., COETZEE, M.L., and TUDOR, D. Multivariable Control of a Run-of-mine Milling Circuit, *Journal of the South African Institute of Mining and Metallurgy*, vol. 90, no. 7, 1990. pp. 173–181.

PRETT, D.M. and GARCIA, C.E, *Fundamental Process Control,* Butterworths, Boston. 1988.

SEBORG, EDGAR and MELLICHAMP. *Process Dynamics and Control,* Wiley. 1989. ◆

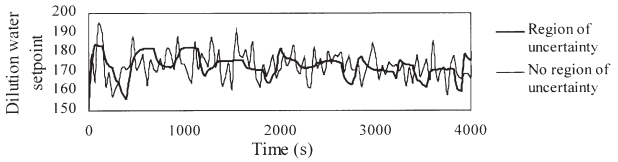

Figure 8—Setpoint changes with and without the region of uncertainty

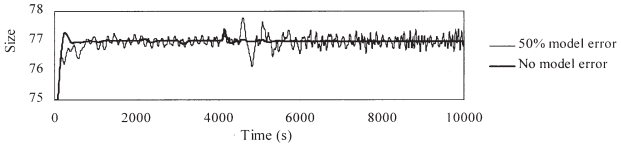

Figure 9—Control of particle size with 50% modelling error

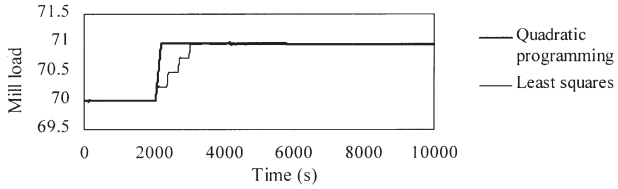

Figure 10—Quadratic programming versus least squares

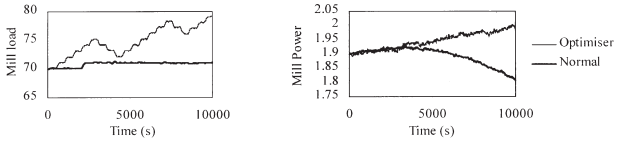

Figure 11—Outputs with and without the power optimizer controller

SAIMM DIARY

Mining

COLLOQUIUM
Mining project financing
26 November 1999, Mövenpick Indaba Hotel, Fourways

COLLOQUIUM
Massive mining of underground orebodies
21–22 February 2001, Mintek, Randburg

COLLOQUIUM
Mining methods for the new millennium
22 February 2001, Mintek, Randburg

Extraction Metallurgy

COLLOQUIUM
Chemicals in the mining industry
13 October 2000, Rustenburg Platinum Mines

COLLOQUIUM
Shotcrete and membrane support
April 2001, Mintek, Randburg

COLLOQUIUM
Nickel, cobalt, coal and zinc
16–18 May 2001, Victoria Falls, Zimbabwe

INTERNATIONAL CONFERENCES

RaSiM 5—Dynamic rock mass response to mining
17–19 September 2001, Sandton, JOHANNESBURG

The 6th International Symposium on Mine Mechanization and Automation
25–28 September 2001, Sandton, JOHANNESBURG

XIV International Coal Preparation Congress and Exhibition
11–15 March 2002, Sandton, JOHANNESBURG

For further information, please contact:
The Secretariat, SAIMM, P.O. Box 61127, MARSHALLTOWN 2107
Tel.: (011) 834-1273/7, Fax: (011) 838-5923 or 833-8156, e-mail: saimm@saimm.co.za