Chapter 6: Conclusions and Recommendations

The first step to design and test a model predictive controller for a milling circuit was to develop a comprehensive simulator for a milling circuit. This was successfully done and from the results it is clear that the simulator gave an adequate representation of the behaviour of a milling circuit. The dynamics of each input/output pair were represented by a first order lead/lag transfer function with dead time or an integrator with dead time. The simulator furthermore made provision for limits (minimum and maximum), random filtered noise and disturbances and drift rates on the inputs and outputs. Each input was equipped with a local PI-controller with an actuator that had dynamics and nonlinearities.

Initially, a model predictive controller was developed for a simple simulation without input and output limits, noise, nonlinearities and drifting of the variables (ideal system). The *Sump level*, *Mill load* and *Particle Size* were chosen as controlled variables and the *Dilution water*, *Feed rocks* and *Flow to cyclone* were chosen as manipulated variables. The MPC controller was written in such a manner that the parameters for each input/output pair could be specified separately. This resulted in a large number of tuning parameters. An MPC parameter optimiser was therefore written that automatically tunes the controller. Excellent control was obtained, but the control deteriorated when the effect of limits on the inputs and outputs were applied. This is because some of the inputs specified by the controller were larger than the maximum allowable limits. This problem was overcome by retuning the controller with the limits in place. The weighting factors on the movement of the manipulated variables were subsequently increased, resulting in less movement of the manipulated variables and preventing them from violating the limits. It is therefore recommended that the controller be tuned while applying maximum expected setpoint changes in order to ensure that the constraints on the inputs are not violated.

Another method to handle constraints explicitly is by using quadratic programming, the effect of which will be discussed later.

When drift rates and nonlinear factors were added to the simulator, control deteriorated further. This was due to the integral effect caused by drift of the disturbance variables. Three

different methods were looked at to accommodate for this error. The first was to assume that part of the error is caused by integral action, instead of assuming the error to be constant over the prediction horizon. The portion of the error that is assumed to be caused by integral action was handled as another tuning parameter. Using this method improved control and removed the error caused by the integral action.

The second method was to use feedforward control on the disturbance variables. The step response coefficients were used as the model for the feedforward controller. This improved performance substantially and also eliminated the error caused by the disturbances. The problem with this method is that it will be sensitive towards modelling errors. Another problem is that not all disturbances can be measured and can therefore not be used in a feedforward controller.

The third method to eliminate the integral errors was to use the disturbance variables (*Feed water* and *Mill speed*) also as manipulated variables. The controller now had five manipulated and three controlled variables. Using four controlled variables by adding the *Density to the cyclone* as another controlled variable was also investigated. The performance improved, but not as much as with the first two options. The reason for this is that extra interaction was introduced by using more manipulated variables. The advantage, however, is that this controller will be much more robust than the other two. Incorporating the disturbances as extra manipulated variables will not always work, since it is not possible to control all inputs or to model the effects of all the inputs.

Noise and disturbances were then added to the input and output variables of the simulator. Good control was still achieved, but the manipulated variables had to be moved severely in order to achieve the required response. This is because the controller attempted to control the noise. This problem was handled by incorporating a region of uncertainty. With this region of uncertainty method, a region around the setpoint (over the whole prediction horizon) of each controlled variable can be specified within which the variable is allowed to deviate. This reduced the movement of the manipulated variables without deteriorating the control.

The robustness of the controller was tested by incorporating a 20 % and a 50 % model error on all the input-output pairs. The conventional controller performed well with a 20 % model error, but became unstable with a 50 % model error. When the region of uncertainty was

incorporated again, the controller was stable and performed satisfactorily even with a 50 % modelling error. The uncertainty region therefore not only decreases the movement of the manipulated variables, but it also increases the robustness of the controller. It is therefore strongly recommended that this uncertainty region method be used if a robust controller is required on a system with an uncertain model with many disturbances (e.g. a milling circuit).

Some other controller configurations were also investigated. The first was to use the input module as part of the controller model. The controller model would then contain the dynamics of the local PI-controllers and the actuator. This slightly reduced the performance of the controller and it is therefore recommended that the model of the process be obtained without the dynamics of the input module included. In practice, however, this is not always possible.

Another option was to use the simulator to generate the model with all the dynamics and the noise. This would represent the case when the plant was used to generate the step response coefficients by using step tests without filtering the data. When this was done the controller became unstable. It is therefore recommended that a model be generated from the step tests and that this model be used to generate the step response coefficients or that the outputs be filtered before generating the coefficients.

When using the optimiser to find the controller parameters, quite large horizons were obtained, which resulted in a drastic increase in computational time. The sizes of the horizons were reduced by running the optimiser and decreasing the horizons until the error was increased by more than 1 % for a unit change in the horizon parameter. By doing this, the horizons were decreased substantially without an unacceptable decrease in the performance of the controller. This resulted in a controller that could execute much quicker due to the reduced computational effort.

Another method to handle the limits on the inputs and outputs is to use quadratic programming. This method uses an advanced optimisation technique that explicitly ensures that the inputs and outputs are kept within bounds. This controller does not have to be tuned with the limits in place since the variables will be kept within the limits as far as possible when the controller is running. Good control was obtained which was much better than when the least-squares method was used with the same parameters. The problem with quadratic

programming is that its computational effort is much more severe, making the controller slower. The controller was however still able to perform well within real time and with computers becoming more powerful and faster by the day, this should not be a problem. Another problem is the fact that the method became unstable for certain parameter values and a solution could not be obtained. The least-squares solution then had to be taken. It is therefore recommended that this method be improved and made more robust if quadratic programming is to be used. One method to do this is to use variable limits on the outputs (Zheng & Morari, 1995). The limit is increased if it is violated to ensure that the output limit is never violated. This variable limit is then also included in the objective function. The movement of the output is therefore penalised more as the variable limit is increased above the actual limit. Because the outputs will now never violate the variable limits, the quadratic programming method will never become unstable. The disadvantage of this method is that the real limits can now still be violated under extraordinary circumstances.

One of the other most important control objectives for a milling circuit is to maximise the throughput. From literature it was found that *Mill power* gives a good indication of the throughput of the milling circuit (Craig *et al.*, 1992). The more power the mill uses, the higher the throughput will be. The mill power is furthermore a parabolic function of the *Mill load*. The *Mill power* (or throughput) can therefore be optimised by changing the setpoint of the *Mill load*. This was done by writing an optimiser controller that changes the setpoint of the *Mill load* to keep the *Mill power* at its optimum. Average values over the time period of the optimiser controller were used to eliminate the effect of noise. The optimiser worked well with the model predictive controller and kept the mill power at its optimum, even when the maximum mill power value drifted.

When comparing the results obtained using model predictive control with those from the literature when the inverse Nyquist array technique was used (Hulbert *et. al.*, 1990), it was found that better control could be obtained with the MPC controller. It has also been shown that the MPC controller is robust and can handle noise and disturbances well. If a model of the process is available, the MPC controller can also be tuned quickly using the simulator and the optimiser. Implementation time will therefore be easy and short, with only the final fine-tuning to be done after implementation. If a model of the system is not available, step response coefficients can be generated directly from step tests. No model therefore has to be fitted to the process, saving time and effort. The disadvantage of not modelling the circuit is

that the simulator and optimiser will then not be used and the MPC controller will have to be tuned online. It is therefore recommended that a model is generated for the circuit and the simulator and optimiser be used to tune the controller to shorten the implementation time.