Chapter 5: Designing the MPC-Controller

The most important output variable that needs to be controlled in a milling circuit is the particle size of the material/medium that leaves the cyclone. The throughput is another important variable and needs to be maximised. This is done by maximising the power consumption of the mill. The other output variables 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 flow rates of the *Dilution water*, *Feed rocks* and *Flow to cyclone* were chosen as the manipulated variables. The *Feed water flow rate* and *Mill speed* were initially taken as disturbance variables, but can easily be included as manipulated variables (as will be shown later on).

There are a large number of tuning parameters for the three input and three output variables using MPC. 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). The problem of determining adequate tuning parameters was overcome by writing an optimising algorithm that automatically finds the best parameters for the controller. The working of this optimiser will be described next.

5.1 The Optimiser for Tuning the MPC Parameters

Any numerical optimisation function needs initial values. The optimiser starts with the current entered values for the parameters of the controller as the initial values. Usually these will be the default values computed using the rules of thumb described in Chapter 3. The goal of the optimiser is to minimise the error between the outputs and the setpoints of the outputs by changing the parameters of the controller. This is achieved in the following manner:

One simulation is performed with the controller switched on using the initial controller parameter values. The simulation is run for a predetermined duration (longer then at least for times the longest time constant to ensure that steady state is reached). While the simulation is running, step changes to the setpoints are applied. The magnitude and time of application of the step change is specified by the user. An error measure (performance index) for this controller is determined by calculating the error between the output of the simulator and the required setpoint at each time step. These errors are then accumulated over the simulation time. A choice of three different error measures can be made (Stephanopoulos, 1984):

- The integral of the squared error (ISE): $E = \int (c - c^{sp})^2 dt$
- The integral of the absolute value of the error (IAE): $E = \int |c - c^{sp}| dt$
- The integral of the time weighted absolute value of the error (ITAE): $E = \int t \cdot |c - c^{sp}| dt$

The first error measure (integral of squared error) will penalise large errors more and smaller errors less. Squaring the difference between the output and the setpoint will furthermore ensure that all errors are positive when added, preventing positive and negative errors from cancelling each other. The second error measure (integral of the absolute value of the error) will penalise large and small errors in an equal manner. The absolute value will make all errors positive to prevent positive and negative errors from cancelling each other. The third error measure (time weighted) will increasingly penalise errors as the simulation time increases. Initial errors will therefore be penalised much less than errors that still exist later on in the simulation. Using this method, the step changes should also be applied during the initial stages of the simulation (preferably at time zero), since errors due to step changes that occur after a long period of time would be penalised much more severely.

After the first simulation, the first parameter (the sampling interval) is changed using *Inverse parabolic interpolation* (Mathews, 1992). The first step in this method (*Inverse parabolic interpolation*) is to bracket the minimum. The minimum is bracketed if three values for the parameter (a, b, c) are found with $a < b < c$ such that the error when simulating using value b is smaller than both the errors of a and c (see Figure 5-1). The minimum error will then lie somewhere between a and c.

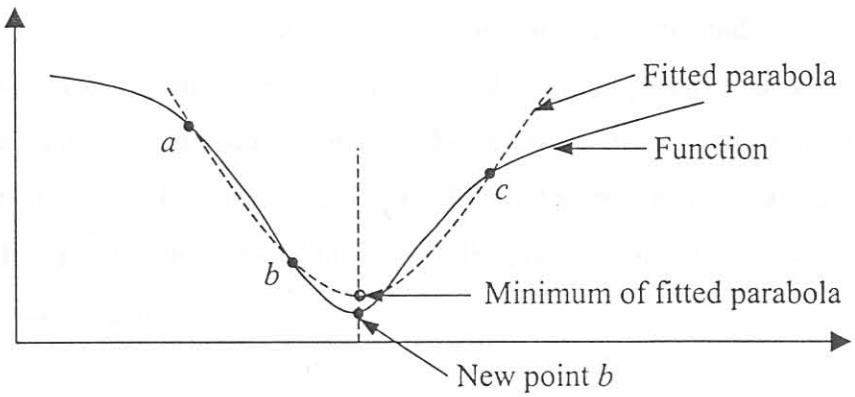

Figure 5-1: Inverse parabolic interpolation

The method used for initially bracketing the minimum is described next. The parameter is initially changed by a certain amount (usually increased). After the change has been made, the simulation is run again under exactly the same conditions as the previous simulation (same length of time, with the same step changes to the setpoints). The error is determined for this new simulation using the same error measure as before. This new error, together with the previous one, can now be used to determine the direction of change for the parameter that will cause the error to decrease. If the new error is smaller than the previous one, the parameter should be increased to decrease the error. If the error is larger than the previous one, the parameter should be decreased to decrease the error.

The parameter is then continually increased in the direction of decreasing error. The amount by which the parameter is increased is doubled with each iteration. This process is continued until the error starts to increase again (this last point will be point *c*). The minimum error will then be bracketed, with the minimum somewhere between the starting point (*a*) and the current point (*c*).

A parabolic function is then fitted to the three points (*a, b, c*) bracketing the minimum. Point *b* is the point with the smallest error between *a* and *c*. The parameter is set equal to the value at the minimum and the simulation is performed again. This new point, together with the two other points with the smallest errors will then form the next three points to fit a parabola to. This process is repeated until the minimum is reached.

It was found that this method did not work well with the integer (natural numbers) parameters such as the horizons. The reason for this is because this method sometimes started to oscillate. A slightly revised method was used for the integer parameters. When the minimum was bracketed, the minimum was assumed to be exactly halfway between point *a* and point *c*, rounded to the nearest whole number. This method of halving the interval was continued until *a* and *c* differed with only two. This meant that the minimum was at the integer value between point *a* and point *c*.

The model horizon is tuned first by the optimiser, followed by the control horizon and lastly the prediction horizon. The weighting matrices (starting with the weighting on the controlled variables) is then tuned using *Inverse parabolic interpolation*. If quadratic programming was selected, the constraint window and the model and prediction horizons for the associated variables are also tuned with the same method used for the other integer parameters. This whole sequence of optimising each parameter individually is repeated a number of times until the best set of parameters is obtained. Sometimes, especially with the horizons, a minimum will never be reached and the error will continue to decrease as the horizon is increased. Limits therefore have to be placed on the horizons, since horizons that are too long will require an unduly large computational effort. This will result in a controller that will not be able to control in real time. Another method is to stop the increase in the parameter if no further significant improvement in the performance index is achieved. The minimum amount of improvement for a unit change in the parameters can be specified to the optimiser. The optimiser will then find the smallest (or largest, in the case of the time step) set of parameters for which no significant improvement in performance is obtained by further increases in that parameter (see Figure 5-2).

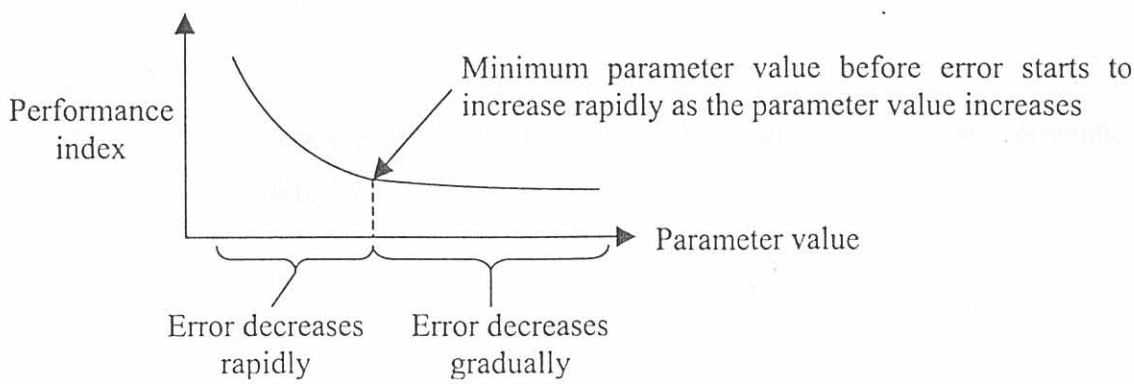

Figure 5-2: Minimising parameter values

The contribution of each output variable to the total error can be specified by using weighting factors. When the error is calculated for each output, it is multiplied with this weighting factor to determine its relative importance. The weighting of the variables that are less important to control, such as the *Sump level,* can therefore be made smaller. The error on this output variable will then be penalised less in the optimisation function.

5.2 Ideal Process

The first controller was designed for an ideal process. Only the dynamics for the process (the state parameters) were entered in the simulator. No noise, nonlinearities or drift rates were entered. The limits on the input and output variables were made large so that they had no influence, and the drift rates were set equal to zero. No local PI-controllers on the inputs were used, assuming perfect control for the inputs. The weighting factors on the *Particle size* and the *Sump level* were set to be of equal importance (both were assigned a value of one), while the weighting factor of the *Sump level* was set to 0,5 since strict control on the level is not that important. The simulation time that was used by the optimiser was set to 5 000 *s*. The time and sizes of the step changes that were applied while optimising are given in Table 5-1.

Table 5-1: Step changes applied to setpoints of controlled variables while optimising

Output	Initial setpoint	Final setpoint	Step time (*s*)
Sump level	60	70	2 000
Mill load	70	71	1 000
Particle Size	75	77	0

The controller parameters that were obtained after optimisation using the above-mentioned settings are given in Table 5-2.

Table 5-2: MPC parameters tuned with ideal plant

Input Parameters			Output parameters			
Input	U	W_2	**Output**	T	V	W_1
Dilution water	87	2,403	Sump level	100	198	5 995,4
Feed rocks	16	0,082	Mill load	21	200	275,4
Flow to cyclone	100	2,392	Particle Size	100	85	469,7
Sampling period (s)		37,1	**Performance index (no limits)**			1 385

Quite good control could be obtained using these parameters (see Figure 5-4). Appendix B shows the response of the manipulated and controlled variables of an ideal process to the setpoint changes given in Table 5-3.

Table 5-3: Step changes applied to setpoints of controlled variables

Output	Initial setpoint	Final setpoint	Step time (s)
Sump level	60	70	4 000
Mill load	70	71	2 000
Particle Size	75	77	0

5.3 Handling of Limits on Inputs and Outputs

As can be seen from Figure 5-3 and the input graphs in Appendix B, quite severe changes are made to the inputs to reach the required setpoints when no limits are set to the inputs. When the limits to the input and output variables were added, the control deteriorated (see Figure 5-4) since the inputs could not be changed as severely as specified by the controller. An indication of the deterioration of the performance of the controller is the performance index. The performance index is a measure of the error between the output and the setpoint. The performance index (calculated using the integral square error) for the controller with no limits on the inputs and outputs was 1 385. This error was calculated for the step changes given in Table 5-3 and with a simulation time of 10 000s. The performance index for the same

controller and step changes deteriorated to 2 207 when the limits on the inputs and outputs were added.

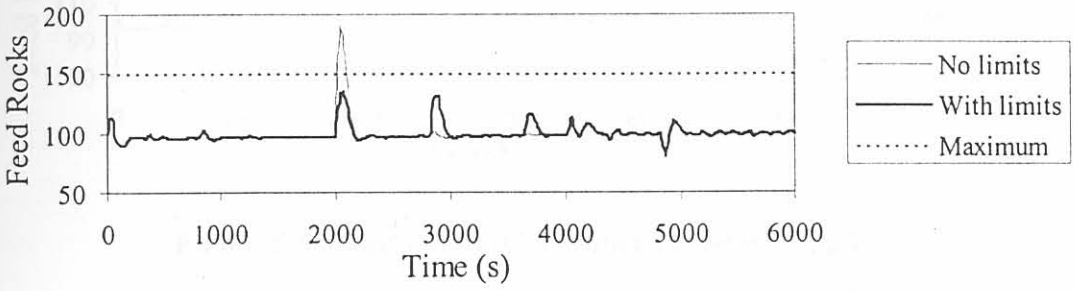

Figure 5-3: *Feed Rocks* (MV) with and without limits

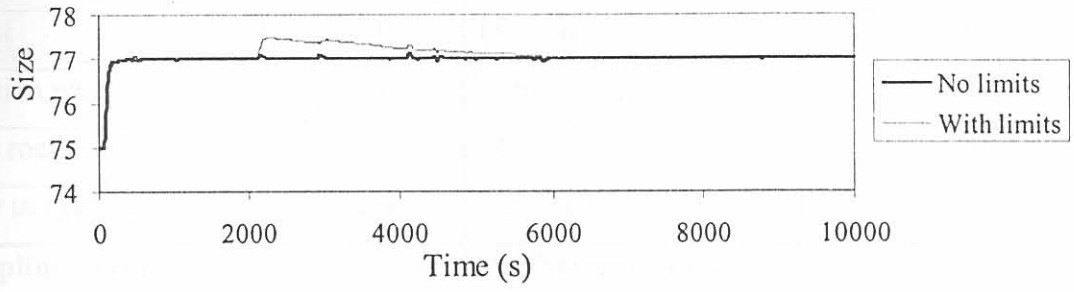

Figure 5-4: *Particle size* (CV) with and without limits

A better controller will therefore be one that does not change the input parameters as severely as the previous one. This can be achieved by increasing the weighting on the movement of the manipulated variables. The manipulated variable that violated its constraints was the *Feed rocks* input. The controller will therefore be improved by increasing the weighting for this variable. This is exactly what happened when the controller was optimised again, this time with the limits on the inputs and the outputs in place. The parameters for this controller are given in Table 5-4. By comparing these parameters with the ones in Table 5-2, it is clear that the weighting on the *Feed rocks* was increased substantially (from 0,082 to 3,698). The increased weighting lessened the movement of the *Feed Rocks* and ensured that it stayed within bounds (less than 150 *t/h* as shown in Figure 5-5). The performance of this controller was superior to that of the previous one, with a performance index of only 1 476 with the limits in place. This is much better than the previous controller with a performance index of 2 207.

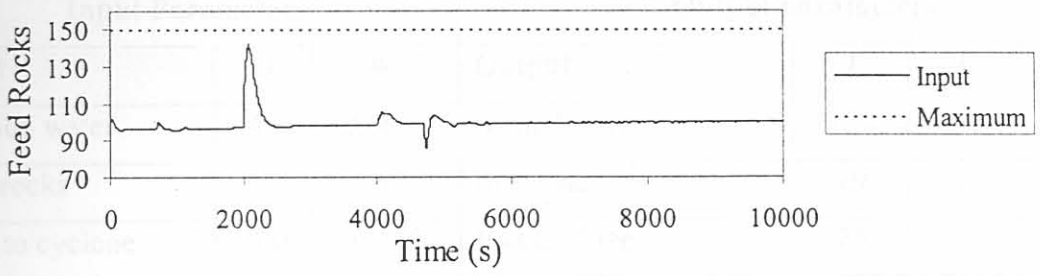

Figure 5-5: *Feed Rocks* (Controller tuned with limits)

Table 5-4: MPC parameters tuned with limits on the inputs and outputs

Input Parameters			**Output parameters**			
Input	***U***	***W*$_2$**	**Output**	***T***	***V***	***W*$_1$**
Dilution water	71	0,9713	Sump level	96	174	71 207
Feed rocks	15	3,698	Mill load	18	146	345,8
Flow to cyclone	98	2,914	Particle Size	49	61	676,8
Sampling period (*s*)		36,4	**Performance index**			1 476

5.4 Effect of Drift Rates and Nonlinear Factors

The next step in the development of the controller was to add drift rates and nonlinear factors to the inputs and the outputs. The dynamics of the actuators were added as well as local PI-controllers on the inputs. From the performance index, it is clear that the drift rates and nonlinear factors caused the performance of the controller to deteriorate substantially (from 1 476 to 4 257).

Table 5-5: MPC parameters tuned with drift rates and additional dynamics

Input Parameters			Output parameters			
Input	U	W_2	**Output**	T	V	W_1
Dilution water	43	0,0603	Sump level	100	59	71 207
Feed rocks	15	4,677	Mill load	19	41	486,2
Flow to cyclone	100	9,815	Particle Size	25	78	728,8
Sampling period (s)		35,8	**Performance index**			4 257

This deterioration in performance is due to the drift of the *Feed water* flowrate, which is not a manipulated variable. Drift of this variable can be seen as a ramp disturbance (a disturbance that is continuously increasing). The *Feed water* furthermore also has an integrating effect on the *Sump level*. The continuous change in the disturbance variable together with the integral action caused the *Sump level* to drift away (see Figure 5-6). It furthermore also caused a small offset in the *Mill load*. This effect can be accommodated for in three ways:

- Assuming that part of the error is caused by an integral error. The error will therefore be increasing for the future predictions instead of assuming a constant error.
- The disturbance variable can be incorporated in a feedforward control mechanism.
- The disturbance variable should be used as another manipulated variable.

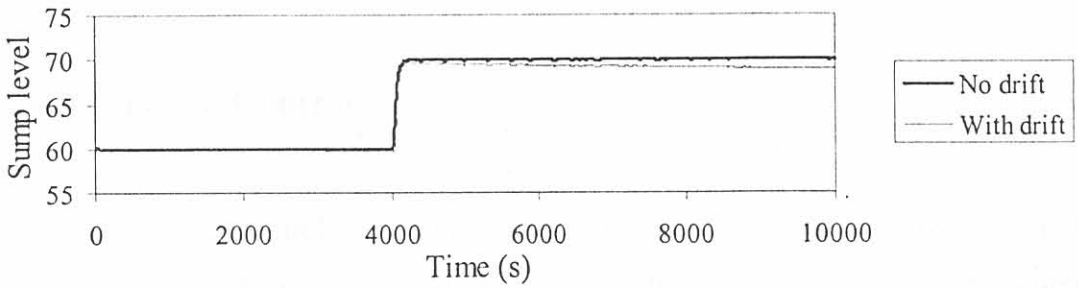

Figure 5-6: Effect of drift rates

5.5 Integral Error

When tuning the MPC controller, the portion of the error assumed to be caused by an integral error should be seen as another tuning factor. The *Sump level* drifted away from the setpoint due to integral action. Part of the error therefore has to be ascribed to integral action. It was found that the best results were obtained by assuming that the total error was caused by integral action. Using an integral error, the drift in the *Sump level* was removed (Figure 5-7 and Appendix B) and the controller performance index was improved from 4 257 to 2 465.

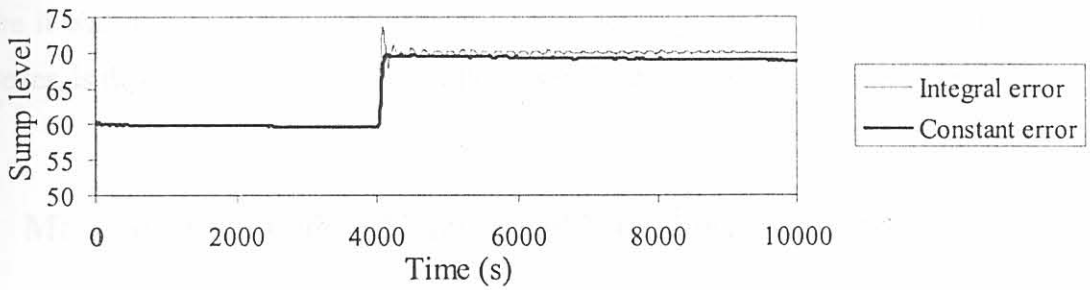

Figure 5-7: Integral error vs. constant error for *Sump level*

The controller can be tuned in the same manner for the other output variables by assuming that a fraction of each error is caused by integral action and handling that fraction as another tuning factor.

5.6 Feedforward Control

By using feedforward control on the *Feed water* and the *Mill speed*, control was improved dramatically. The performance index was decreased from 4 257 to 1 987. The improved performance is shown in Figure 5-8, where the drift in the *Sump level* has been completely removed.

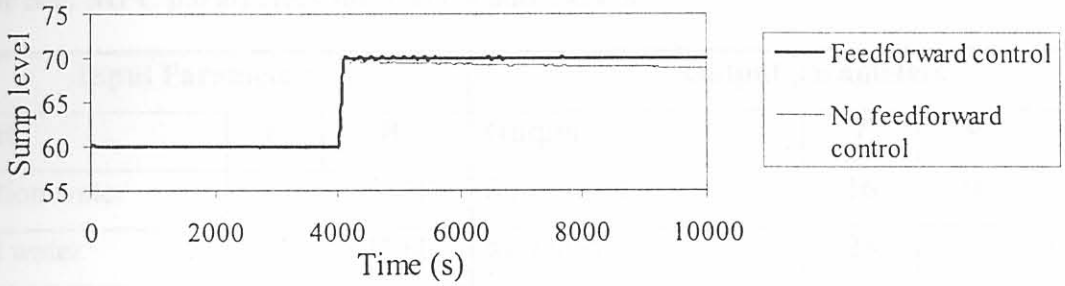

Figure 5-8: Control with and without feedforward control on *Feed water*

The feedforward controller even showed superior performance over the previous example where it was assumed that part of the error is caused by integral action. The disadvantage, however, is that any feedforward controller is sensitive towards modelling errors.

5.7 More Manipulated and Controlled Variables

The other option to handle drift of the inputs was to use all five inputs as manipulated variables. This also improved the performance of the controller and completely removed drift in the *sump level* (performance index decreased from 4 257 to 3 052), but the improvement was not as impressive as with feedforward control. This is due to the fact that additional interaction was introduced by the extra two manipulated variables. This controller, however, will be more robust and not as sensitive to model errors as the feedforward controller. The parameters for the controller with five controlled variables and three manipulated variables are shown in Table 5-6.

The controller was extended by adding the density to the cyclone as an extra controlled variable (parameters given in Table 5-7). In this case the performance index cannot be directly compared to the previous controllers, since the error of an extra variable is also added, which will increase the performance index and make the controller seem worse than it is. The results of the two controllers with three and four controlled variables are shown in Appendix B (both had five manipulated variables).

Table 5-6: MPC parameters for 5 MV's and 3 CV's

Input Parameters			Output parameters			
Input	U	W_2	**Output**	T	V	W_1
Dilution water	40	6,178	Sump level	16	56	71 244
Feed water	31	47,80	Mill load	23	58	325,4
Feed rocks	14	2,327	Particle Size	22	72	638,8
Flow to cyclone	53	13,09				
Mill speed	3	1 041,5				
Sampling period (*s*)		35,8	**Performance index**			3 052

Table 5-7: MPC parameters for 5 MV's and 4 CV's

Input Parameters			Output parameters			
Input	U	W_2	**Output**	T	V	W_1
Dilution water	35	6,027	Sump level	15	55	71 251
Feed water	35	37,79	Mill load	18	67	354,5
Feed rocks	15	4,261	Density	20	116	220,1
Flow to cyclone	48	11,96	Particle Size	22	74	764,5
Mill speed	23	876,6				
Sampling period (*s*)		37,9	**Performance index**			3 835

The extra interaction caused by the adding of the extra two manipulated variables is clearly shown in Figure 5-9.

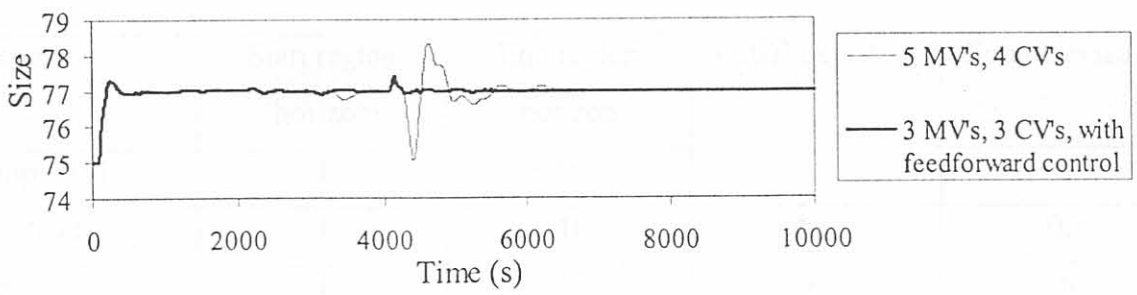

Figure 5-9: 5 MV's vs. 3 MV's with feedforward control

5.8 Noise

Noise can also have a severe effect on the performance of the controller. After addition of noise to the inputs as well as the outputs, the performance index (with feedforward control), deteriorated to about 8 000. The performance index will, however, change for each run, since the noise is randomly generated. A common problem with noise is that the controller will attempt to control the noise. This may cause an excessive change in the manipulated variables as is shown by the graphs of the inputs in Appendix B. This problem can be reduced by using a region of uncertainty as described in Chapter 3. A band around the setpoints was specified that would be large enough to contain the noise. The parameters for the uncertainty region on the controlled variables are shown in Table 5-8. This reduced the movement of the manipulated variable as can be seen from Figure 5-10 and Appendix B. The controlled variable is now allowed to drift within the bounds specified, especially that of the *Sump level* and *Mill load*, for which strict control is not crucial. The deviation of the *Size* from the setpoint, however, still remained more or less the same.

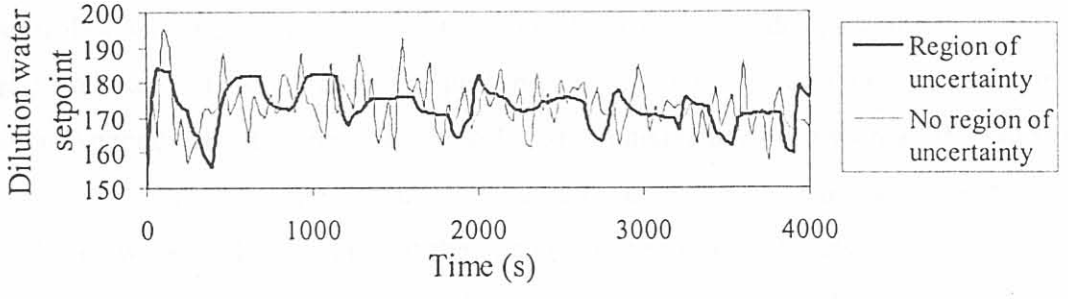

Figure 5-10: Changes in *Dilution water* setpoint with and without region of uncertainty

Table 5-8: Region of uncertainty parameters

Output	Start region horizon	End region horizon	Initial deviation	Final Deviation
Sump level	1	59	1	1
Mill load	1	41	0,3	0,3
Size	1	78	0,5	0

The uncertainty region parameters should also be seen as tuning parameters and changed until the best control is obtained. When deciding on the best controller, the amount of

movement of the manipulated variables should therefore be considered in addition to the deviation of the outputs from the setpoint.

5.9 Model Errors

The robustness of the controller was tested by making the parameters of the model used to create the model for the controller different from the model for the simulator. The controller was first tested for a 20 % modelling error by making the parameters of the model 120 % of that of the process (model parameters are 20 % larger than the process parameters). No noise was added to ensure that the performance indexes could be compared. The parameters of the controller were the same as those given in Table 5-5 and the region of uncertainty was also set to zero. Feedforward control was used. A 20 % modelling error decreased the performance index from 1 990 to 2 231. The control was still very satisfactory as can be seen from Appendix B.

The controller was also tested for a 50 % modelling error (model parameters = 150 % of process parameters). This decreased the performance index further to 3 957. By looking at the outputs in Appendix B, it seems that control is still satisfactory even with a 50 % modelling error, since the deviation of the *Size* is of the same order as the noise and will therefore not be noticeable. However, when looking at the changes in the inputs (*Dilution water* - see Figure 5-11 or Appendix B), it can be seen that the process is actually unstable, since the changes in the inputs are increasing.

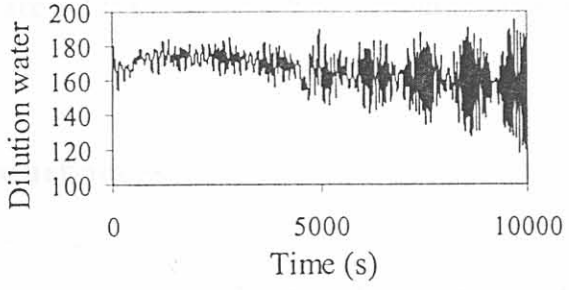

Figure 5-11: Changes in *Dilution water* with 50 % modelling error

When a region of uncertainty was used, the performance improved dramatically. The best performance was obtained by using an uncertainty region on only the *Size*-output. The parameters are shown in Table 5-9, which improved the performance index from 3 957 to 2 656. This time the process was stable when looking at the inputs (see Appendix B).

Table 5-9: Uncertainty region parameters for 50 % model error

Output	Start region horizon	End region Horizon	Initial deviation	Final Deviation
Size	1	78	0,1	0

The robustness of the controller is therefore increased by using a region of uncertainty, since it is less sensitive to changes in the model error. As can be seen from Appendix B, the control is still very satisfactory, even with a 50 % model error. Although the control of the *Particle Size* deteriorated visibly, the deviation from the setpoint is still less than the noise on that output (see Figure 5-12).

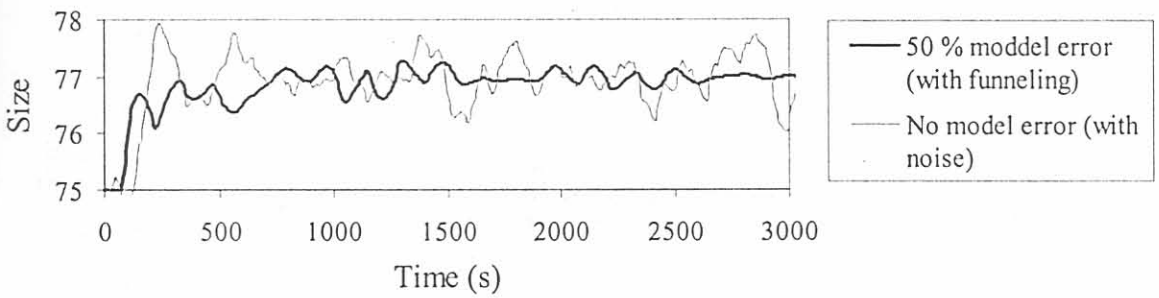

Figure 5-12: Control with 50 % modelling error

5.10 Handling of Disturbances

The robustness of the controller was further tested by applying disturbances to the input variables that are not used as manipulated variables (*Feed water* and the *Mill speed*). The time of application and size of the disturbances are shown in Table 5-10.

Table 5-10: Disturbances

Disturbance variable	Initial value	Final value	Time of disturbance
Feed water	7	9	2 000 s
Mill speed	80	82	6 000 s

The outputs with and without feedforward control are given in Appendix B. From the results it is clear that the controller with feedforward control again showed superior performance when compared with using no feedforward control or when using an integral error (see Figure 5-13 and Figure 5-14).

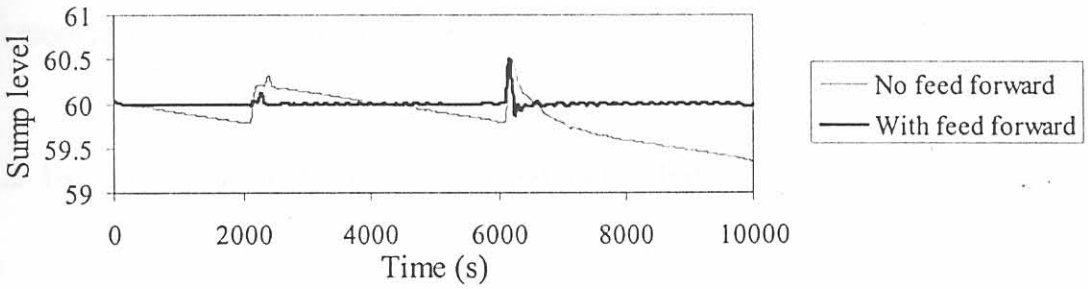

Figure 5-13: Disturbance rejection (with and without feedforward control)

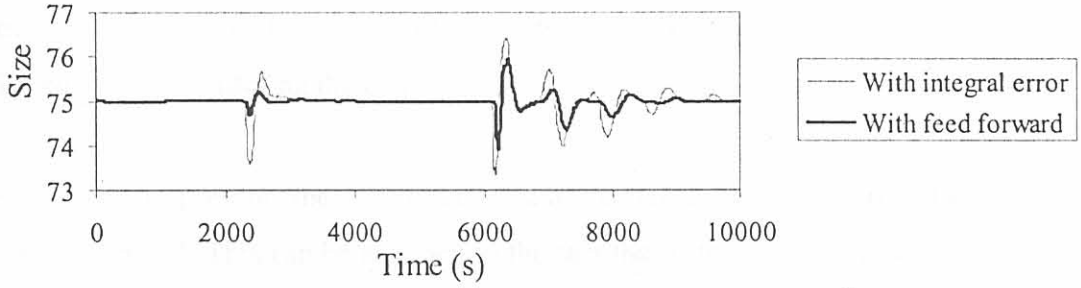

Figure 5-14: Disturbance rejection (feedforward control vs. using an integral error)

5.11 Other options

5.11.1 Using the Input model as part of the Controller Model

For all the previous tests, only the state model was used to generate the model for the MPC-controller. A more elaborate option is to also incorporate the input model as part of the model used by the controller. The dynamics of the valve and the local PI-controllers will therefore be incorporated in the model. When this was done, the controller performance deteriorated slightly. The deterioration in performance is shown by the increase in the performance index from 1 987 to 1 996. No gain in performance is therefore obtained by also incorporating the input module as part of the controller model.

5.11.2 Using the plant with noise as the Controller Model

The model for the controller was previously generated using the state model parameters. This model did not contain noise or any of the other nonlinear factors. A controller was also tested when the simulator, with all the noise, nonlinearities and dynamics of the inputs, was used to generate the model. The step response coefficients will therefore contain noise. This represents the case when the outputs of the plants would be used to generate the step response coefficients without filtering the outputs.

When noise was present, the controller became unstable when the simulator was used to generate the model. This can be ascribed to the fact that if there is a difference in the last two values of the step response coefficients, the controller will assume it is due to integral action and treat it as such. This difference, however, is caused by noise. It is therefore essential that the outputs be filtered before the model is generated.

When no noise was present, the controller worked, but the performance was much worse than previously (the performance deteriorated from 1 987 to 3 910). It is therefore better to generate the step response coefficients using a model of the plant instead of using the plant or simulator itself, or to filter the outputs before generating step response coefficients.

5.11.3 Retuning the Model of the Controller

Since a milling circuit will change over time due to, for example, wear, the model of the process will become less accurate. This phenomenon is represented by drift of the base level, and drift of the minimum and maximum values of the variables. To ensure that the model remains accurate, the controller can be retuned from time to time to update the model (Metzner, 1988). This will ensure that the model remains close to the real process even when the process slowly changes with time. This is a kind of adaptive control scheme, but it is not applied continuously. The controller will only be updated by manually choosing so. This should only be done while the plant is operating normally.

Continuous adaptive control is not used since milling circuits are difficult to model due to the presence of a large number of possible disturbances. The adaptive mechanism will cause the model to be changed too often in an attempt to model the disturbances, making the model less accurate. The advantage of the manual adaptive scheme is that it can be switched on while the plant is operating relatively smoothly. If large disturbances are present, the tuner can be switched off to prevent it from modelling these disturbances and making the model less accurate.

The effect of retuning the model was tested as follows: The simulation was run for 5 000*s* with no step changes being applied, after which a change in one of the parameters was made. The integral action of the *Dilution water* on the *Sump level* was changed from 0,0025 to 0,0035. This was done to show what effect a change in one of the plant parameters would have on the controller. In practice, this change would probably occur over a long period of time, but for the sake of studying the effect, the change was made after 5 000*s*. The simulation was then continued for another 5 000*s* without retuning the model of the controller. This caused some of the outputs like the *Sump level* and *Particle size* to deviate from their setpoints (see Figure 5-15 and Appendix B). From this it is clear that the controller will deteriorate in performance if the process changes over time.

The effect of retuning the model can now be demonstrated as follows. The simulation was again run for 5 000*s*, after which the same change to the process was made. When the simulation was run for the next 5 000*s*, the step response coefficients for the MPC-controller

were first regenerated for this different process. This improved the performance dramatically (see Figure 5-15 and Appendix B) and the performance index was decreased from 2 548 to only 32,8.

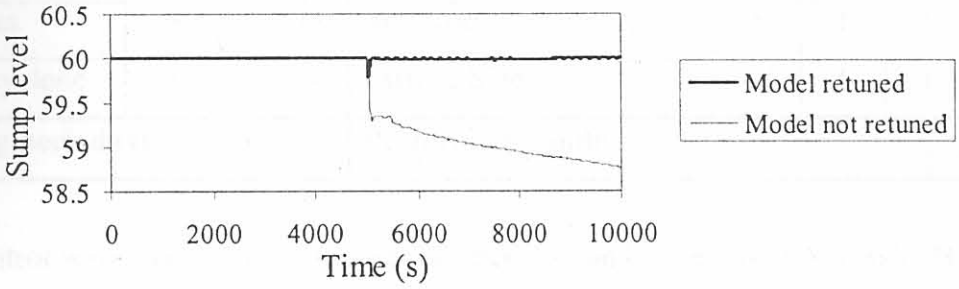

Figure 5-15: Retuning the controller model

5.12 Quadratic Programming

In quadratic model predictive control, a more advanced optimisation algorithm is used to calculate the changes in the manipulated variables. With quadratic programming, the optimisation method explicitly ensures that all the variables (inputs and outputs) remain within their limits. The controller can therefore be tuned without limiting the variables in the simulator, since the optimisation algorithm will ensure that the variables stay within their limits. The MPC parameter optimiser was again used to find the parameters for a quadratic MPC controller. For quadratic programming, two extra parameters are required for each output. These two parameters specify the constraint window of the future predicted values. The constraint window is the future predicted values that should be kept within the output limits. The first parameter (V_b) specifies from which future predicted value the constraint window should start. The second parameter (V_e) specifies up to which future predicted value the outputs should be constrained. The parameters obtained by using the MPC parameter optimiser are shown in Table 5-11.

Table 5-11: MPC parameters for Quadratic Programming

Input Parameters			Output parameters					
Input	U	W_2	**Output**	T	V	V_b	V_e	W_1
Dilution water	3	0,0245	Sump level	10	13	1	1	5 996
Feed rocks	5	0,0711	Mill load	8	5	1	1	1 816
Flow to cyclone	3	1,774	Particle Size	47	47	1	1	450,2
Sampling period (s)		35,1	**Performance index**					2 975

Good control was obtained using these parameters as can be seen from Appendix B (the drift rates were set to zero in this case). When the least squares method was used with these same parameters, the performance index deteriorated from 2 975 to 4 223. This is because the least squares method does not check for violation of the limits on the variables. Figure 5-16 shows the difference between quadratic programming and the least squares method when the same parameters were used.

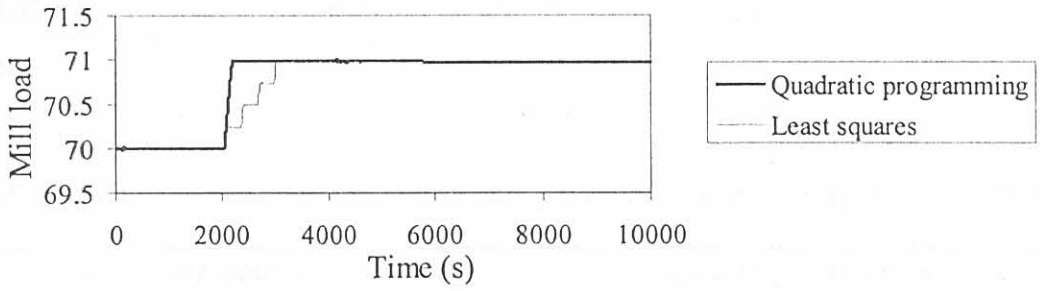

Figure 5-16: Quadratic programming vs. least squares

It was found that the technique used to solve the quadratic programming problem did become unstable for certain combinations of parameters. Restrictions on the parameter selection by the optimisation function were put in place to prevent such a combination of parameters from being chosen. The moment the optimiser finds a set of parameters that makes the quadratic programming solution unstable, the optimiser goes back to the previous set of parameters. This ensures that parameters leading to unstable quadratic programming solutions are never chosen. When running the controller, however, and the quadratic programming solution becomes unstable, the controller will take the least-squares solution for only that specific time step. This will ensure that a solution is always obtained.

5.13 Tuning with minimum of 1% improvement

When tuning the controller using the conventional MPC parameter optimiser, the horizons became very large. This is undesirable, since larger horizons will result in longer computational time. For large controllers with many inputs and outputs, this may become a limiting factor and prevent controllers from operating in real time.

In an effort reduce the size of the horizons, the controller was tuned for a minimum improvement in the performance index of 1 %. This means that the horizons were decreased until the performance index increased with more than 1 % for a change in the horizon of 1. By incorporating this feature in the optimiser, the sizes of the horizons were decreased substantially, while the performance index actually improved slightly from 1 987 to 1 948. This can be contributed to the fact that the previous optimiser probably got stuck at a local minimum, and this new mechanism enabled it to get away from this local minimum. The values of the new parameters are shown in Table 5-12, with the previous parameters shown in brackets.

Table 5-12: MPC parameters with minimum performance index improvement of 1 %

Input Parameters			**Output parameters**			
Input	U	W_2	**Output**	T	V	W_1
Dilution water	4 (43)	0,0506	Sump level	2 (100)	3 (59)	71 230
Feed rocks	3 (15)	9,528	Mill load	4 (19)	6 (41)	561,7
Flow to cyclone	13 (100)	1,905	Particle Size	16 (25)	23 (78)	469,4
Sampling period (*s*)	37,1 (35,8)		**Performance index**		1 948 (1 987)	

5.14 Power Control

Since the *Mill power* is a parabolic function of the *Mill load* (see Figure 2-3), the power can be controlled at its maximum value using an optimiser controller. It is controlled at its

maximum, since maximum power consumption will result in a maximum throughput. The power can be controlled by changing the setpoint of the *Mill load*. It was found that if the time interval for this controller (the time the optimiser controller waits before taking action) was set at 500 seconds, good control could be obtained. The setpoint of the *Mill load* was changed during each control action to optimise the *Mill power*. Average values for the *Mill power* over the time period were used to eliminate the effect of noise. These average values were used to determine if the *Mill power* was increasing or decreasing, which could in turn be used to determine how the *Mill load* should be changed. From Figure 5-17 it is clear that the power controller keeps the *Mill power* at its optimum by changing the setpoint of the *Mill load*. The *Mill power* continues to increase, since the maximum value of the *Mill power* has a positive drift rate. The *Mill power* therefore follows this positive drift of the maximum value. Without the power controller, the power falls away below the maximum value.

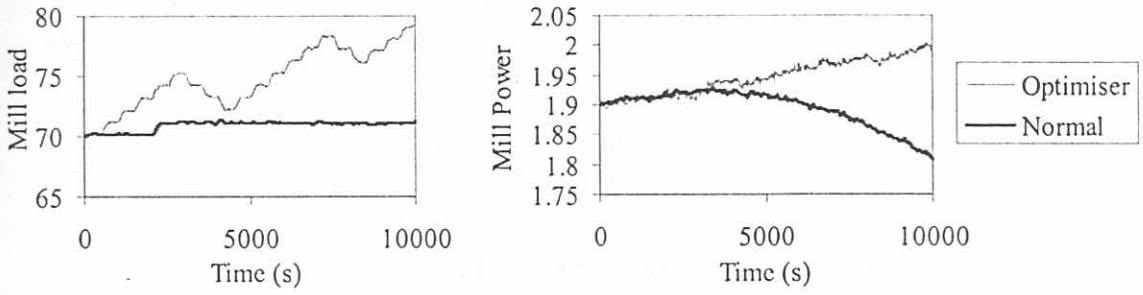

Figure 5-17: Power control

5.15 Comparison with PI-Control

A combination of single input single output proportional-integral (PI) controllers was tuned. The best results were obtained when the *Particle Size out of Cyclone* was controlled with the *Dilution water* fed to the sump, the *Mill load* was controlled with the *Feed Rocks* and the *Sump level* was controlled with the *Flow to Cyclone*. The resulting parameters for these controllers are given in Table 5-13.

Table 5-13: PI-controller parameters

Manipulated Variable	Controlled variable	Proportional action	Integral action
Dilution water	Particle size	11,7	85,2
Feed Rocks	Mill load	1 000	10
Sump level	Flow to cyclone	-1 000	10

The PI-controllers were used for comparison with the performance of the MPC-controller. When the same step changes to the setpoints were applied, it was clear that the MPC-controller showed much better performance with much less interaction (see Figure 5-18 and Appendix B).

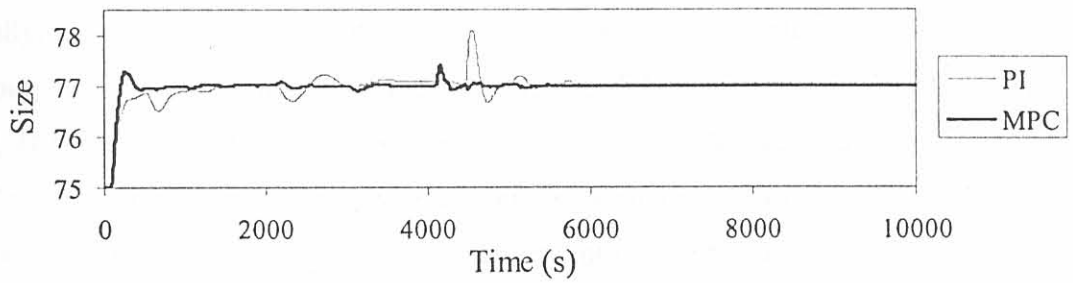

Figure 5-18: PI control versus MPC control

The interaction of the PI-controller can be lessened by using more advanced techniques like the Inverse Nyquist Array technique. This has been done (Hulbert *et. al.*, 1990), but at the expense of severely slowing the dynamics of the control of the sump level. When comparing the results of the MPC-controller with those in the literature with PI-control with the Inverse Nyquist Array technique, the MPC-controller was still superior.