Fault-tolerant Nonlinear MPC using Particle Filtering *

Laurentz E. Olivier[1] and Ian K. Craig[2]

[1] Department of Electrical, Electronic, and Computer Engineering, University of Pretoria, Pretoria, South Africa, (e-mail: laurentz.olivier@sasol.com)
[2]Department of Electrical, Electronic, and Computer Engineering, University of Pretoria, Pretoria, South Africa, (e-mail: ian.craig@up.ac.za)

Abstract: A fault-tolerant nonlinear model predictive controller (FT-NMPC) is presented in this paper. State estimates, required by the NMPC, are generated with the use of a particle filter. Faults are identified with the nonlinear generalized likelihood ratio method (NL-GLR), for which a bank of particle filters is used to generate the required fault innovations and covariance matrices. A simulated grinding mill circuit serves as the platform for illustrating the use of this fault detection and isolation (FDI) scheme along with the NMPC. The results indicate that faults can be correctly identified and compensated for in the NMPC framework to achieve optimal performance in the presence of faults.

Keywords: Fault-tolerant, generalized likelihood ratio, model predictive control, nonlinear, particle filter.

1. Introduction

Model predictive control (MPC) is a very successful control technology and has been widely applied in the processing industry (Bauer and Craig, 2008). A good overview of available industrial MPC applications is presented in Qin and Bagwell (2003). MPC makes use of an explicit plant model to predict the future behaviour of the plant with a proposed set of control actions. The control actions that produce the most optimal plant outputs over the prediction horizon are the optimal control values. There are therefore a number of reasons why the control performance of MPC could degrade over time (Morari and Lee, 1999). Morari and Lee (1999) stressed the need to detect abnormality and diagnose its root cause in order to sustain the benefits of model predictive control over a long period of time.

It is well known that control performance degrades in the event of malfunctions in actuators, sensors, or other system components (Zhang and Jiang, 2008), and easily develops into production stoppages (Blanke et al., 1997). The way in which MPC is formulated leads to a natural representation of faults within the system. Actuator failures are easily represented as control constraints, and faults that change the process dynamics can be accommodated by modifying the internal MPC model.

Fault-tolerant control (FTC) strategies can broadly be classified as either passive FTC or active FTC (Zhang and Jiang, 2008). With passive FTC the objective is to design the controller such that it is robust enough to handle a class of presumed faults. Active FTC has the objective of isolating faults and adapting the control strategy such that the stability and control performance of the entire system may be maintained.

Zhang and Jiang (2008) provides a good overview of active fault-tolerant control systems, and Prakash et al. (2005) shows how fault-tolerant control may be implemented in the MPC framework. Reviewing the integration of fault-tolerant control with MPC, it is evident that the design of the state observer is the key to being successful (Deshpande et al., 2009). This is because the observer generates outputs based on a proposed set of faults, which may be compared to the plant outputs for fault identification. One promising fault identification method for nonlinear systems, the nonlinear generalized likelihood ratio (NL-GLR) method used in this work (Deshpande et al., 2009), is based on this idea.

Particle filtering is an estimation technique based on representing probability densities with weighted samples (particles) (Arulampalam et al., 2002). Because this representation does not make any assumption about the form of the distribution, it can be used in general nonlinear, non-Gaussian systems. Particle filters have been used for fault detection and isolation before, see e.g. Wang and Syrmos (2008). To the knowledge of the authors they have however not previously been used in the GLR framework for integration with NMPC.

This paper presents the use of particle filters in the GLR framework for fault detection and isolation. The fault information is integrated with an NMPC controller. A grinding mill simulator is used as the platform to illustrate

* This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Number 90533).

the effective use of this method to provide optimal control performance in the presence of faults.

2. Fault Tolerant non-linear MPC

The discussion in the rest of this article pertains to the general discrete time state-space representation of a dynamic system

$$x_k = f\left(x_{k-1}, u_{k-1}, \theta_{k-1}, v_{k-1}\right) \tag{1}$$

$$y_k = g\left(x_k, \theta_k, e_k\right) \tag{2}$$

where $x \in \mathbb{R}^n$ is the state vector and $y \in \mathbb{R}^m$ is the output vector, $f(\cdot)$ and $g(\cdot)$ are possibly nonlinear functions describing the state transitions and the outputs respectively, u_k contains the exogenous inputs, θ_k represents the parameters, v_k is the state noise and e_k is the measurement noise.

2.1 Non-linear MPC

Considering the system presented in equations (1) and (2), the objective of a model predictive controller at each sampling instant is to minimise the scalar performance index

$$\min_u V(u, x_k) \tag{3}$$

$$s.t.\ x \in X, u \in U \tag{4}$$

$$\theta_c(x, u) \leq 0 \tag{5}$$

where $x : \mathbb{R} \to \mathbb{R}^{n_x}$ is the state trajectory, $u : \mathbb{R} \to \mathbb{R}^{n_u}$ is the control trajectory, x_k is the state at time step k and $\theta_c(x, u)$ is the constraint vector.

The performance index (or objective function) to be minimized penalizes output values different from the reference values as well as excessive control moves. This ensures that the output values will tend to the reference values without making undue control moves or violating constraints. The flexibility with which control objectives can be incorporated into the objective function is partly why MPC is such a popular technology. The objective function used in this work is similar to that shown in Qin and Bagwell (2003) as:

$$\begin{aligned} V(u, x_k) = & \sum_{i=1}^{N_p} \left[(y_{r,i} - y_i)^T Q_r (y_{r,i} - y_i) + Q_l y_i\right] \\ & + \sum_{i=0}^{N_c - 1} \Delta u_i^T R \Delta u_i \end{aligned} \tag{6}$$

where N_p and N_c are the prediction and control horizons respectively; Q_r, Q_l, and R are weighting matrices corresponding to the reference tracking, linear optimization objectives (LP weights), and control movements; y_r is the output reference and y is the output prediction. If any plant output variable does not have a specific reference value, but should rather be minimized, the corresponding entry in the Q_r matrix could be made zero while the corresponding entry in the Q_l matrix is given a positive weighting value (or negative if the value should be maximized).

The only difference in this formulation between the linear and nonlinear versions of the MPC is whether the output predictions are supplied by propagating the control vector through a linear or nonlinear model.

2.2 Fault detection and isolation

There are many ways in which model based fault detection and isolation (FDI) can be done, see e.g. Alcorta Garcia and Frank (1997) for an overview on methods for nonlinear systems.

Consider the innovation sequence calculated from the state observer:

$$\gamma(k) = y(k) - g(\hat{x}_k, \theta_k), \tag{7}$$

with $k \in [t, t+N]$. Without any faults the innovation sequence should be Gaussian white noise. If however there are any faults, we can use the test statistic over the innovation sequence window

$$\epsilon(t, N) = \sum_{k=t}^{t+N} \gamma(k)^T V(k)^{-1} \gamma(k) \tag{8}$$

to detect the fault. Note that $V(k)$ is the innovation covariance. If this test statistic exceeds a threshold the fault is confirmed. The threshold and window length are tuning parameters for the fault detection test.

Now suppose a set of observers is used, each operating with a different postulated fault, and each generates outputs along with the nominal observer. The problem of fault isolation is then finding the fault mode observer that best explains the measurement sequence $\{y(t) \ldots y(t+N)\}$ generated over the time window for which the fault was detected. The NL-GLR method can then be stated in mathematical form as:

$$\min_{b_{f_j}} \left(J_{f_j}\right) = \sum_{i=t}^{t+N} \gamma_{f_j}^T(i) V_{f_j}(i)^{-1} \gamma_{f_j}(i), \tag{9}$$

where $\gamma_{f_j}(i)$ and $V_{f_j}(i)$ are respectively the innovations and innovation covariance matrices generated by the fault mode observer corresponding to fault f_j. The isolated fault corresponds to the fault mode observer for which J_{f_j} is the smallest, with $\hat{b}_{f_j}$, the fault magnitude that produces this minimum value.

Because the innovation sequence of (7) and the innovation covariance matrix ($V(k)$) appear directly in the Kalman filtering framework, it is natural to make use of the Kalman filter as the state observer (or the extended Kalman filter (EKF) in the nonlinear case). In the linear case the use of the Kalman filter is wholly justified as it is the optimal estimator (Arulampalam et al., 2002) (assuming also that the noise is Gaussian). The EKF however, as is often used in the nonlinear case, suffers some known limitations (see Julier and Uhlmann (2004) for a more complete discussion). If is for this reason that particle filtering is rather used in this work as further discussed in Section 3.

2.3 Representing faults

There is a set of faults for which fault isolation is a trivial task. Examples include gross sensor faults and faults for which there are direct measurements available. Many modern sensors will inform the user when it has exceeded the calibration range, and if such a violation is impossible it is trivial to state that the sensor has failed at the corresponding condition.

Also consider a flow valve on a line where a flow measurement is made. If the valve is stuck while the controller is trying to manipulate the flow, the flow reading will be a direct indication of the magnitude of the fault. This class of faults is not considered in this work. Focus will be placed on actuator errors where no direct indication is available and that are much more difficult to isolate (Prakash et al., 2005).

If the j-th actuator is stuck abruptly at time t then the corresponding plant input can be represented as:

$$u_{u_j}(k) = m(k) + \left[b_{u_j} - e_{u_j}^T m(k)\right] e_{u_j}\sigma(k-t) \tag{10}$$

where b_{u_j} represents the constant value at which the j-th actuator is stuck, e_{u_j} is the fault vector with element j equal to one and all other elements equal to zero, and $\sigma(t)$ is the unit step function:

$$\sigma(t) = 0 \text{ if } t < 0;\ \sigma(t) = 1 \text{ if } t \geq 0. \tag{11}$$

Similarly, if a bias occurs in the j-th sensor a time t, then the measured output can be represented as:

$$y_{y_j}(k) = g(x_k, \theta_k) + b_{y_j} e_{y_j}\sigma(k-t) + v(k), \tag{12}$$

where b_{y_j} is the bias present in the j-th sensor. If the j-th sensor fails abruptly at time instant t, then the measurement can be represented as:

$$\begin{aligned} y_{y_j}(k) \ = \ & g(x_k, \theta_k) + \left[b_{y_j} - e_{y_j}^T g(x_k, \theta_k)\right] \cdot \\ & e_{y_j}\sigma(k-t) + v(k), \end{aligned} \tag{13}$$

where b_{y_j} is the constant output value of the j-th sensor. If there is a drift in the j-th sensor from time t, the measured output can be represented as:

$$y_{y_j}(k) = g(x_k, \theta_k) + b_{y_j} e_{y_j}\zeta(k-t) + v(k), \tag{14}$$

where b_{y_j} is the drift slope present in the j-th sensor and

$$\zeta(t) = 0 \text{ if } t < 0;\ \zeta(t) = t \text{ if } t \geq 0. \tag{15}$$

Any of these fault representations may be considered by the fault mode observers as used in the NL-GLR framework.

3. Particle Filtering

Let $Y_t = \{y_0, \ldots, y_t\}$ represent the sequence of all measurements up to the current time, then the general state estimation problem is formulated as the solution of the conditional distribution function $p(x_t|Y_t)$, which is the distribution of the state given all the observations up to time t. The idea behind the particle filter (Arulampalam et al., 2002) is to represent the required posterior density function (PDF) by a set of random samples and associated weights as

$$p(x_t|Y_t) \approx \sum_{i=1}^{N_s} w_t^i \delta(x_t - x_t^i) \tag{16}$$

where N_s is the number of particles and $\{x_t^i, w_t^i\}_{i=1}^{N_s}$ is the set of particle locations and weights. This representation is illustrated in Fig. 1 where the sizes of the dots represent the weights of the particles. If $N_s \to \infty$, this representation becomes equivalent to the functional description of the PDF.

The weights are defined to be (Ristic et al., 2004):

$$w_t^i \propto w_{t-1}^i \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{t-1}^i, y_t)}, \tag{17}$$

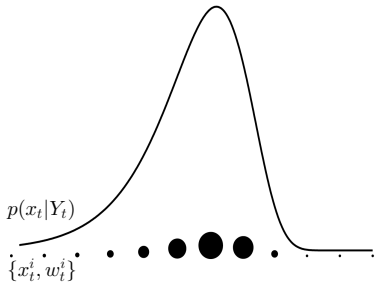

Fig. 1. Distribution representation with particles

where $q(x_t^i|x_{t-1}^i, y_t)$ is a proposal distribution called an importance density. Ideally the importance density should be the true posterior distribution $p(x_t|Y_t)$ but as this is not known in general, a proposal distribution is used. One popular suboptimal choice, that is used in this work, is the transitional prior

$$q(x_t^i|x_{t-1}^i, y_t) = p(x_t|x_{t-1}). \tag{18}$$

This is a conceptual description of how the particle filter works. There are many variations on the particle filter, but the variation used in this work is the sampling importance resampling (SIR) particle filter, for which the algorithm can be stated as:

- **Initialize:** For $t = 0$ draw N_s states (x_0) from the prior PDF $p(x_0)$.
- **Propagate particles:** For each $t > 0$ sample $\tilde{x}_t \sim p(x_t|x_{t-1})$ using the state transition function (1) and the state noise distribution.
- **Calculate weights:** The weight of each particle is given by: $w_t = p\left(y_t|g(\hat{x}_t, v_t)\right)$, which can be evaluated based on the output function (2) and the shape of the output distribution.
- **Normalize weights:** Normalize the weights as
$$\tilde{w}_t^i \ = \ \frac{w_t^i}{\sum_{j=1}^{Ns} w_t^j}.$$
- **Resample:** Multiply particles with high importance weights and suppress particles with low importance weights. This is to overcome the degeneracy problem present in the normal sequential importance sampling algorithm (Arulampalam et al., 2002). More information on the details of the resampling algorithm can be seen in Olivier et al. (2012).
- **PDF:** The required PDF is then given by
$$p(x_t|Y_t) \approx \frac{1}{N_s}\sum_{i=1}^{N_s} \delta(x_t - x_t^i).$$
- **State estimates:** Once the PDF $p(x_t|Y_t)$ is known, the state estimate is calculated as a point estimate from the distribution. In this work we use the mean, which with the particle representation after resampling simply becomes:
$$\hat{x}_t = \frac{1}{N_s}\sum_{i=1}^{N_s} x_t^i.$$

4. Application of the FT-NMPC

The efficacy of the FT-NMPC method is now illustrated through application on a milling circuit simulator. A full

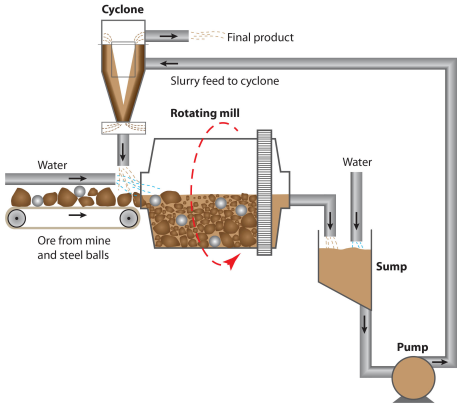

Fig. 2. Grinding mill circuit

nonlinear description of the plant is used in the NMPC, with faults being identified with the NL-GLR method using particle filters.

4.1 Process description

In this section we give a brief description and show the model equations for the grinding mill circuit. A much more complete description of this model can be found in Le Roux et al. (2013), in which all of the parameter values listed later in Table 1 were also derived.

The layout of the milling circuit is shown in Fig. 2. Ore from the mine is added to the grinding mill along with steel balls and water. Ore is ground down into fine particles inside the mill, and exits as a slurry through an end-discharge grate. Note that coarse ore and steel balls cannot pass through the grate. The slurry is sent to a sump where further water is added. The slurry is then pumped to a cyclone for classification. Sufficiently ground down material leaves the top of the cyclone as the product of the milling circuit. Material that should be ground down further leaves the bottom of the cyclone and re-enters the mill.

The inputs into the milling circuit are the mill water feed (MIW), mill solids feed (MFS), mill steel balls feed (MFB), the speed of turning of the mill (α_{speed}), the sump water feed (SFW), and the cyclone feed flow-rate (CFF). These are all the manipulated variables (MVs) used in the controller. The milling circuit outputs (also the controlled variables in the NMPC) are the mill load (LOAD), sump volume (SVOL), particle size estimate (PSE), circuit throughput (THP), and the mill power draw (P_{mill}).

The mill state transitions are given by:

$$\dot{X}_{mw} = MIW - V_{mwo} \tag{19}$$

$$\dot{X}_{ms} = (1-\alpha_r)\frac{MFS}{D_s} - V_{mso} + RC \tag{20}$$

$$\dot{X}_{mf} = \alpha_f \frac{MFS}{D_s} - V_{mfo} + FP \tag{21}$$

$$\dot{X}_{mr} = \alpha_r \frac{MFS}{D_s} - RC \tag{22}$$

$$\dot{X}_{mb} = \frac{MFB}{D_b} - BC. \tag{23}$$

with

$$V_{mwo} = V_V \cdot \varphi \cdot X_{mw}\left(\frac{X_{mw}}{X_{mw}+X_{ms}}\right) \tag{24}$$

$$V_{mso} = V_V \cdot \varphi \cdot X_{mw}\left(\frac{X_{ms}}{X_{mw}+X_{ms}}\right) \tag{25}$$

$$V_{mfo} = V_V \cdot \varphi \cdot X_{mw}\left(\frac{X_{mf}}{X_{mr}+X_{ms}}\right) \tag{26}$$

$$BC = \frac{1}{D_b\phi_b} \cdot P_{mill} \cdot \varphi \cdot \left(\frac{X_{mr}}{X_{mr}+X_{ms}}\right) \tag{27}$$

$$RC = \frac{1}{D_s\phi_r} \cdot P_{mill} \cdot \varphi \cdot \left(\frac{X_{mr}}{X_{mr}+X_{ms}}\right) \tag{28}$$

$$FP = \frac{P_{mill}}{D_s\phi_f\left[1+\alpha_{\phi_f}\left(\frac{LOAD}{v_{mill}} - v_{P_{max}}\right)\right]} \tag{29}$$

$$\varphi = \left(\frac{\max\left[0, \left(X_{mw} - \left(\frac{1}{\varepsilon_{ws}} - 1\right)X_{ms}\right)\right]}{X_{mw}}\right)^{0.5} \tag{30}$$

$$P_{mill} = P_{max} \cdot \{1 - \delta_{P_v} Z_x^2 - \delta_{P_s} Z_r^2\} \cdot (\alpha_{speed})^{\alpha_P} \tag{31}$$

$$Z_x = \frac{LOAD}{v_{P_{max}} \cdot v_{mill}} - 1 \tag{32}$$

$$Z_r = \frac{\varphi}{\varphi_{P_{max}}} - 1 \tag{33}$$

The sump state transition equations are:

$$\dot{X}_{sw} = V_{mwo} + SFW - V_{swo} \tag{34}$$

$$\dot{X}_{ss} = V_{mso} - V_{sso} \tag{35}$$

$$\dot{X}_{sf} = V_{mfo} - V_{sfo}. \tag{36}$$

$$V_{swo} = CFF \cdot \left(\frac{X_{sw}}{X_{ss}+X_{sw}}\right) \tag{37}$$

$$V_{sso} = CFF \cdot \left(\frac{X_{ss}}{X_{ss}+X_{sw}}\right) \tag{38}$$

$$V_{sfo} = CFF \cdot \left(\frac{X_{sf}}{X_{ss}+X_{sw}}\right). \tag{39}$$

The cyclone is described as:

$$V_{ccu} = (V_{sso} - V_{sfo}) \cdot \left(1 - C_1 \cdot e^{-\frac{CFF}{\varepsilon_c}}\right) \cdot \left(1 - \left[\frac{F_i}{C_2}\right]^{C_3}\right) \cdot \left(1 - P_i^{C_4}\right) \tag{40}$$

$$V_{cwu} = V_{swo} \cdot \frac{V_{ccu} - F_u \cdot V_{ccu}}{F_u \cdot V_{swo} + F_u \cdot V_{sfo} - V_{sfo}} \tag{41}$$

$$V_{cfu} = V_{sfo} \cdot \frac{V_{ccu} - F_u \cdot V_{ccu}}{F_u \cdot V_{swo} + F_u \cdot V_{sfo} - V_{sfo}} \tag{42}$$

$$F_u = 0.6 - (0.6 - F_i) \cdot e^{-\frac{V_{ccu}}{\alpha_{su}\varepsilon_c}}, \tag{43}$$

$$F_i = \frac{V_{sso}}{V_{swo} + V_{sso}} \tag{44}$$

$$P_i = \frac{V_{sfo}}{V_{sso}}. \tag{45}$$

with $V_{cfo} = V_{sfo} - V_{cfu}$ and $V_{cco} = (V_{sso} - V_{sfo}) - V_{ccu}$.

Table 1. Parameters and constants contained in the milling circuit equations.

Parameter	Value	Description
α_f	0.055	Fraction of fines in the ore
α_r	0.465	Fraction of rocks in the ore
ϕ_f	29.57	Power per ton of fines produced [kW·h/t]
ϕ_r	6.03	Rock abrasion factor [kW·h/t]
ϕ_b	90	Steel abrasion factor [kW·h/t]
ε_{ws}	0.6	Maximum water-to-solids volumetric flow at zero slurry flow
V_V	84	Volumetric flow per "flowing volume" driving force [h^{-1}]
P_{max}	1661	Maximum mill motor power [kW]
δ_{P_v}	0.5	Power change parameter for volume of mill filled
δ_{P_s}	0.5	Power change parameter for fraction solids in the mill
$v_{P_{max}}$	0.34	Fraction of mill volume filled for maximum power
$\varphi_{P_{max}}$	0.57	Rheology factor for maximum mill power
α_P	1.0	Fractional power reduction per fractional reduction from maximum mill speed
v_{mill}	59.1	Mill volume [m^3]
α_{ϕ_f}	0.01	Fractional change in kW/fines produced per change in fractional filling of mill
ε_c	128.85	Coarse split parameter
α_{su}	0.87	Parameter related to solids in cyclone underflow
C_1	0.6	Constant
C_2	0.7	Constant
C_3	4	Constant
C_4	4	Constant

The milling circuit outputs are:

$$\begin{aligned} LOAD &= X_{mw} + X_{ms} + X_{mr} + X_{mb} \\ SVOL &= X_{sw} + X_{ss} \\ PSE &= \frac{V_{fo}}{V_{co}+V_{fo}} \\ THP &= V_{co} + V_{fo} \end{aligned} \tag{46}$$

as well as P_{mill} given in (31). The parameter values contained in the process equations are listed in Table 1 (units are shown next to parameters that are not dimensionless).

4.2 Implementation details

The specific values used in each subsystem for the simulation are listed in Table 2. Note that only $LOAD$ and PSE have setpoints. The sump level ($SVOL$) does not need to be at any specific value, as long as the sump does not run dry or overflow. The mill power draw (P_{mill}) should be minimized and the circuit throughput (THP) should be maximized, hence they have LP weights.

4.3 Simulation results

The simulation is set-up with the parameters and tuning values as shown in Tables 1 and 2. The simulation is run for a total of 3 hours, and is propagated at a sampling period of 10 seconds. After 0.8 hours the valve supplying the mill feed water (MIW) gets stuck such that the feed water remains constant at a value of 10 for the rest of the simulation run. One hour into the simulation the PSE setpoint is increased by 15 % (such a large change will typically not be performed in practice and is used here for illustration purposes only). Two hours into the simulation the PSE setpoint is returned to its original value. The setpoint changes illustrate the control performance in the presence of faults.

Table 2. Values used for implementation.

Parameter	Symbol	Value
NMPC parameters		
MV high limits	$\overline{\mathrm{U}}$	[100 200 10 1 300 450]
MV low limits	$\underline{\mathrm{U}}$	[0 0 0 0.4 0 200]
Prediction horizon	$\mathrm{N_p}$	20
Control horizon	$\mathrm{N_c}$	3
Execution interval	t_S	1 minute
Setpoints	$\mathrm{y_r}$	[0.32 − 0.67 − −]
Reference weight	$\mathrm{Q_r}$	diag([1 0 500 0 0])
LP weight	$\mathrm{Q_l}$	diag([0 0 0 − 1 1])
MV move weight	R	diag([0.08 0.02 2.5 25 0 2.5])
FDI parameters		
Window length (samples)	N	60
Fault confirmation threshold	−	10
Particle filter parameters		
Number of particles	$\mathrm{N_s}$	1000
State noise matrix	$\mathrm{Q_y}$	diag([5×10^{-4}, 0.1, 5×10^{-4}, 0.05, 5, 0.01])
Output noise matrix	$\mathrm{R_u}$	$10^{-5} \times$ diag([1, 1, 0.1, 0.1, 1, 1, 0.1, 0.1])

In the first simulation run the fault information is artificially supplied to the NMPC after 1.5 hours. This is done to illustrate the control performance when the fault information is not available against when the fault information is available. It is visible from Fig. 3 that the controller struggles to achieve acceptable reference tracking performance without the fault information (the first setpoint change), but when this information is available the performance is vastly improved. Without the fault information the controller incorrectly believes that the changes it is making in the MIW are reflected in the plant. With the fault information available however the controller knows that it cannot make use of MIW to achieve the required PSE, and compensates by using the other manipulated variables.

In the second simulation run the full FT-NMPC is used with no artificial fault information supplied to the controller. The same simulation setup is used and the PSE reference tracking is shown in Fig. 4. The fault is correctly detected as an MIW actuator error, shortly after 1 hour into the simulation. The actual value at which the MIW was frozen is 10, and the detected fault magnitude is 9.23. The MIW values are shown in Fig. 5.

Similar FDI results are achieved for faults in the other actuators mentioned in Section 4.1. Abrupt sensor bias as shown in (12) is also correctly detected by the method. The calculations required by the particle filter bank are completed within the process sampling time of 10 seconds.

There is a trade-off between the speed of fault detection and the accuracy of the detection. If the detection window size is increased the detection accuracy of the fault magnitude will improve, but the time before detection will increase. If for example the window size is increased to 0.2 hours, the situation would be similar to that shown

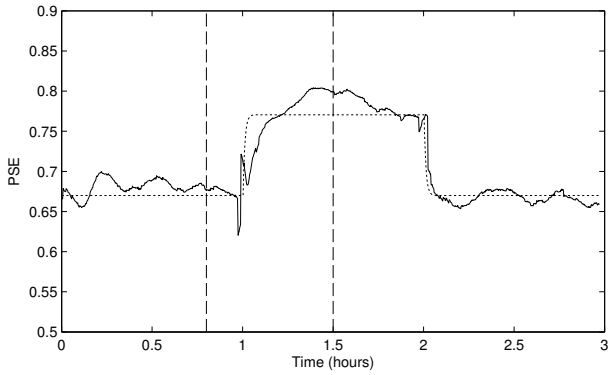

Fig. 3. PSE tracking performance with MIW fault (unknown to NMPC between 0.8 h and 1.5 h - indicated by horizontal dashed lines).

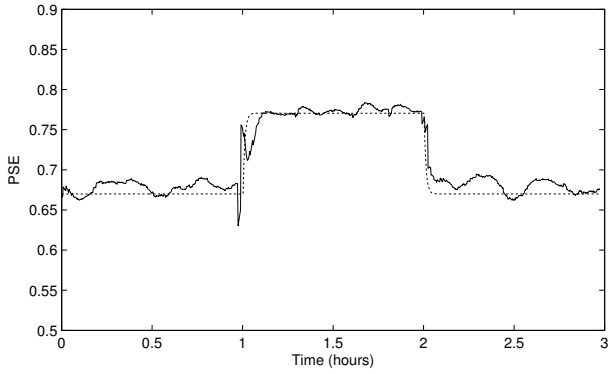

Fig. 4. PSE tracking performance with MIW fault (FT-NMPC).

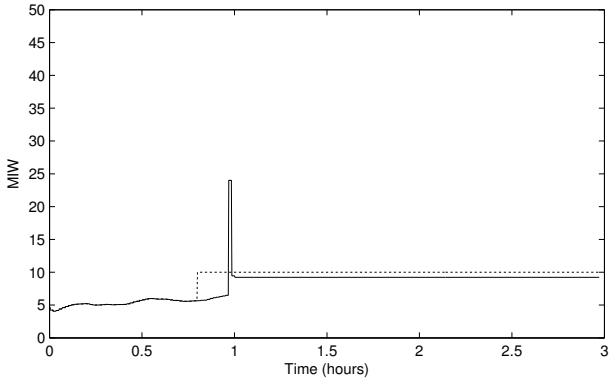

Fig. 5. MIW (FT-NMPC) where the solid line is the controller value and the dashed line is the actual value

in Fig. 3 where the plant is operating for quite a while without the proper fault information.

5. Conclusion

On-line monitoring and maintenance is one of the key aspects to ensure lasting benefit from MPC installations. Active fault tolerant control can help in this regard by detecting faults and adapting the MPC to retain optimal operation. These faults include actuator and sensor failures, but also include drifts and biases that are usually more difficult to detect but still lead to degraded MPC performance.

The NL-GLR method using particle filters was implemented for FDI along with NMPC. A grinding mill circuit simulator was used to illustrate the effectiveness of this method. The fault detection and correct fault identification shows how controller performance can be maintained in the presence of faults. This shows promise for implementation in real-world applications.

Acknowledgements

The authors would like to thank the South African National Research Foundation for their support of this research.

References

Alcorta Garcia, E. and Frank, P.M. (1997). Deterministic nonlinear observer-based approaches to fault diagnosis: a survey. *Control Eng. Practice*, 5, 663 – 670.

Arulampalam, M.S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE transactions on signal processing*, 50, 174 – 188.

Bauer, M. and Craig, I.K. (2008). Economic assessment of advanced process control - A survey and framework. *Journal of Process Control*, 18, 2 – 18.

Blanke, M., Izadi-Zamanabadi, R., Bøgh, S.A., and Lunau, C.P. (1997). Fault-tolerant control systems - a holistic view. *Control Eng. Practice*, 5, 693 – 702.

Deshpande, A.P., Patwardhan, S.C., and Narasimhan, S.S. (2009). Intelligent state estimation for fault tolerant nonlinear predictive control. *Journal of Process Control*, 19, 187 – 204.

Julier, S.J. and Uhlmann, J.K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92, 401 – 422.

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

Morari, M. and Lee, J.H. (1999). Model predictive control: Past, present and future. *Comput. Chem. Eng.*, 23, 667 – 682.

Olivier, L.E., Huang, B., and Craig, I.K. (2012). Dual particle filters for state and parameter estimation with application to a run-of-mine ore mill. *Journal of Process Control*, 22, 710 – 717.

Prakash, J., Narasimhan, S., and Patwardhan, S.C. (2005). Integrating model based fault diagnosis with model predictive control. *Ind. Eng. Chem. Res.*, 44, 4344 – 4360.

Qin, S.J. and Bagwell, T.A. (2003). A survey of industrial model predictive control technology. *Control Eng. Practice*, 11, 733 – 764.

Ristic, B., Arulampalam, S., and Gordon, N. (2004). *Beyond the Kalman filter: Particle filters for tracking applications*. Artech House, Boston.

Wang, X. and Syrmos, V.L. (2008). Interacting multiple particle filters for fault diagnosis of non-linear stochastic systems. In *American Control Conference*, 4274 – 4279. Seattle, Washington, USA.

Zhang, Y. and Jiang, J. (2008). Bibliographical review on reconfigurable fault-tolerant control systems. *Annual reviews in Control*, 32, 229 – 252.