REFERENCES

Abbaszadeh, M. and H. J. Marquez (2009). LMI optimization approach to robust H_∞ observer design and static output feedback stabilization for discrete-time nonlinear uncertain systems. *International Journal of Robust and Nonlinear Control* **19**(3), 313–340.

Abd El-Rahman, M. K., B. K. Mishra and R. K. Rajamani (2001). Industrial tumbling mill power prediction using the discrete element method. *Minerals Engineering* **14**(10), 1321–1328.

Alamo, T., J. M. Bravo and E. F. Camacho (2005). Guaranteed state estimation by zonotopes. *Automatica* **41**(6), 1035–1043.

Amestica, R., G. D. Gonzalez, J. Barria, L. Magne, J. Menacho and O. Castro (1993). A SAG mill circuit dynamic simulator based on a simplified mechanistic model. In: *XVIII International Mineral Processing Congress*. Sydney. pp. 117–129.

Amestica, R., G. D. Gonzalez, J. Menacho and J. Barria (1996). A mechanistic state equation model for semiautogenous mills. *International Journal of Mineral Processing* **44–45**, 349–360.

Apelt, T. A. (2002). Inferential measurement models for semi-autogenous grinding mills. PhD thesis. Department of Chemical Engineering, University of Sydney. Australia.

Apelt, T. A., S. P. Asprey and N. F. Thornhill (2001). Inferential measurement of SAG mill parameters. *Minerals Engineering* **14**(6), 575–591.

Apelt, T. A., S. P. Asprey and N. F. Thornhill (2002). Inferential measurement of SAG mill parameters II: State estimation. *Minerals Engineering* **15**(12), 1043–1053.

Atassi, A. N. and H. K. Khalil (1999). A separation principle for the stabilization of a class of nonlinear systems. *IEEE Transactions on Automatic Control* **44**(9), 1672–1687.

Atassi, A. N. and H. K. Khalil (2000). Separation results for the stabilization of nonlinear systems using different high-gain observer designs. *Systems & Control Letters* **39**(3), 183–191.

Austin, L. G. and P. T. Luckie (1972). Methods for determination of breakage distribution parameters. *Powder Technology* **5**(4), 215–222.

Austin, L. G. and V. K. Bhatia (1972). Experimental methods for grinding studies in laboratory mills. *Powder Technology* **5**(5), 261–266.

Austin, L. G., R. S. C. Rogers, K. A. Brame and J. Stubican (1988). A rapid computational procedure for unsteady-state ball mill circuit simulation. *Powder Technology* **56**(1), 1–11.

Austin, L. G., Z. Rogovin, R. S. C. Rogers and T. Trimarchi (1983). The axial mixing model applied to ball mills. *Powder Technology* **36**(1), 119–126.

Banini, G. A. (2000). An integrated description of rock breakage in comminution machines. PhD thesis. University of Queensland (JKMRC). Australia.

Bartolini, G., A. Pisano, E. Punta and E. Usai (2003). A survey of applications of second-order sliding mode control to mechanical systems. *International Journal of Control* **76**(9), 875–892.

Bazin, C. and C. B-Chapleau (2005). The difficulty associated with measuring slurry rheological properties and linking them to grinding mill performance. *International Journal of Mineral Processing* **76**(1–2), 93–99.

Bazin, C., M. St-Pierre and D. Hodouin (2005). Calibration of the perfect mixing model to a dry grinding mill. *Powder Technology* **149**(2–3), 93–105.

Ben-Tal, A. and A. Nemirovskii (2001). *Lectures on Modern Convex Optimization: Analysis, Algorithms and Engineering Applications*. MPS-SIAM Series on Optimization. MPS-SIAM. Philadelphia.

Ben-Tal, A. and A. Nemirovskii (2002). Robust optimization – Methodology and applications. *Mathematical Programming, Series B* **92**, 453–480.

Bhaumik, A., J. Sil, S. Banerjee and S. K. Dutta (1999). Supervised learning algorithm for open loop NN based control of tumbling mill. In: *Proceedings of the IEEE Region 10 TENCON 99 Conference.*. Vol. 1. pp. 395–398 vol.1.

Billings, S. A. (1980). Identification of nonlinear systems – A survey. *IEE Proceedings D on Control Theory and Applications* **127**(6), 272–285.

Bitmead, R. R., M. Gevers and V. Wertz (1990). *Adaptive optimal control – The thinking man's GPC*. Prentice-Hall. Englewood Cliffs, NJ.

Bock, H. G. and E. Kostina (2001). Robust experimental design. In: H.G. Bock, Project A4, Optimization methods for reactive flows of Sonderforshungsbereich 359 Reactive Flows, Diffusion and Transport, Report 1999-2001. Technical report. University of Heidelberg.

Bock, H. G. and K. J. Plitt (1984). A multiple shooting algorithm for direct solution of optimal control problems. In: *Proceedings 9th IFAC World Congress Budapest*. Pergamon Press. pp. 243–247.

Borell, M., P. Backstrom and L. Soderberg (1996). Supervisory control of autogenous grinding circuits. *International Journal of Mineral Processing* **44–45**, 337–348.

Bouche, C., C. Brandt, A. Broussaud and W. Drunick (2005). Advanced control of gold ore grinding plants in South Africa. *Minerals Engineering* **18**(8), 866–876.

Bravo, J. M., T. Alamo and E. F. Camacho (2006). Robust MPC of constrained discrete-time nonlinear systems based on approximated reachable sets. *Automatica* **42**(10), 1745–1751.

Camacho, E. F. and C. Bordons (2003). *Model Predictive Control*. Vol. Second Edition. Springer-Verlag. London, UK.

Casavola, A., D. Famularo and G. Franze (2003). Linear embedding vs. direct nonlinear MPC schemes: A case study. In: *Proceedings of the American Control Conference, 4-6 June*. Vol. 5. pp. 4305–4310.

Casavola, A., D. Famularob and G. Franzéa (2004). Robust constrained predictive control of uncertain norm-bounded linear systems. *Automatica* **40**, 1865–1876.

Chandramohan, R. and M. S. Powell (2005). Measurement of particle interaction properties for incorporation in the discrete element method simulation. *Minerals Engineering* **18**(12), 1142–1151.

Chandramohan, R. and M. S. Powell (2006). A structured approach to modelling SAG mill liner wear – Monitoring wear. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 133–148.

Chen, X., J. Zhai, Q. Li and S. Fei (2007*a*). Override and model predictive control of particle size and feed rate in grinding process. In: *Proceedings of the 26th Chinese Control Conference, July 26-31*. Zhangjiajie, Hunan, China. pp. 704–708.

Chen, X., J. Zhai, S. Li and Q. Li (2007*b*). Application of model predictive control in ball mill grinding circuit. *Minerals Engineering* **20**(11), 1099–1108.

Chen, X., Q. Li and S. Fei (2008). Constrained model predictive control in ball mill grinding process. *Powder Technology* **186**(1), 31–39.

Chen, X., S. Li, J. Zhai and Q. Li (2009). Expert system based adaptive dynamic matrix control for ball mill grinding circuit. *Expert Systems with Applications* **36**(1), 716–723.

Chu, D., T. Chen and H. J. Marquez (2007). Robust moving horizon state observer. *International Journal of Control* **80**(10), 1636–1650.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987*a*). Generalized predictive control: Part I: The basic algorithm. *Automatica* **23**(2), 137–148.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987*b*). Generalized predictive control: Part II: Extensions and interpretations. *Automatica* **23**(2), 149–160.

Cleary, P. (2001). Modelling comminution devices using DEM. *International Journal for Numerical and Analytical Methods in Geomechanics* **25**(1), 83–105.

Cleary, P. W., M. Sinnott and R. Morrison (2006). Prediction of slurry transport in SAG mills using SPH fluid flow in a dynamic DEM based porous media. *Minerals Engineering* **19**(15), 1517–1527.

Coetzee, L. C., I. K. Craig and E. C. Kerrigan (2008). Nonlinear model predictive control of a run-of-mine ore milling circuit. In: *proceedings of the 17th IFAC World Congress, July 6-11*. Seoul, Korea.

Coetzee, L. C., I. K. Craig and E. C. Kerrigan (2009). Robust nonlinear model predictive control of a run-of-mine ore milling circuit. *IEEE Transactions on Control Systems Technology, accepted for publication*.

Conradie, A. V. E. and C. Aldrich (2001). Neurocontrol of a ball mill grinding circuit using evolutionary reinforcement learning. *Minerals Engineering* **14**(10), 1277–1294.

Craig, I. K. and I. M. MacLeod (1995). Specification framework for robust control of a run-of-mine ore milling circuit. *Control Engineering Practice* **3**(5), 621–630.

Craig, I. K. and I. M. MacLeod (1996). Robust controller design and implementation for a run-of-mine ore milling circuit. *Control Engineering Practice* **4**(1), 1–12.

Craig, I. K., D. G. Hulbert, G. Metzner and S. P. Moult (1992*a*). Extended particle-size control of an industrial run-of-mine milling circuit. *Powder Technology* **73**(3), 203–210.

Craig, I. K., D. G. Hulbert, G. Metzner and S. P. Moult (1992*b*). Optimized multivariable control of an industrial run-of-mine milling circuit. *Journal of the South African Institute of Mining and Metallurgy* **92**(6), 169–176.

Cutler, C. R. and B. L. Ramaker (1980). Dynamic matrix control – A computer control algorithm. In: *Proceedings of the Joint Automatic Control Conference*. Vol. 1. San Francisco, CA.

Cuzzola, F. A., J. C. Geromel and M. Morari (2002). An improved approach for constrained robust model predictive control. *Automatica* **38**, 1183–1189.

Davila, J., L. Fridman and A. Levant (2005). Second-order sliding-mode observer for mechanical systems. *IEEE Transactions on Automatic Control* **50**(11), 1785–1789.

De Nicolao, G., L. Magni and R. Scattolini (1996). On the robustness of receding horizon control with terminal constraints. *IEEE Transactions on Automatic Control* **41**(3), 451–453.

Desbiens, A., A. Pomerleau and D. Hodouin (1996). Frequency based tuning of SISO controllers for two-by-two processes. *IEE Proceedings of Control Theory and Applications* **143**(1), 49–56.

Diehl, M., H. G. Bock and E. Kostina (2006). An approximation technique for robust nonlinear optimization. *Mathematical Programming: Series A and B* **107**(1), 213–230.

Diehl, M., H. G. Bock and J. P. Schloder (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization* **43**(5), 1714–1736.

Ding, B., Y. Xi and S. Li (2004). A synthesis approach of on-line constrained robust model predictive control. *Automatica* **40**, 163–167.

Djordjevic, N. (2005). Influence of charge size distribution on net-power draw of tumbling mill based on DEM modelling. *Minerals Engineering* **18**(3), 375–378.

Djordjevic, N., R. Morrison, B. Loveday and P. Cleary (2006). Modelling comminution patterns within a pilot scale AG/SAG mill. *Minerals Engineering* **19**(15), 1505–1516.

Dong, H. and M. H. Moys (2003). Load behavior and mill power. *International Journal of Mineral Processing* **69**(1-4), 11–28.

Duarte, M., A. Suarez and D. Bassi (1999*a*). Multivariable predictive neuronal control applied to grinding plants. In: *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials, 1999. IPMM 1999*. Vol. 2. pp. 975–982 vol.2.

Duarte, M., A. Suarez and D. Bassi (2001). Control of grinding plants using predictive multivariable neural control. *Powder Technology* **115**(2), 193–206.

Duarte, M., F. Sepulveda, A. Castillo, A. Contreras, V. Lazcano, P. Gimenez and L. Castelli (1999*b*). A comparative experimental study of five multivariable control strategies applied to a grinding plant. *Powder Technology* **104**(1), 1–28.

El-Farra, N. H. and P. D. Christofides (2003). Bounded robust control of constrained multivariable nonlinear processes. *Chemical Engineering Science* **58**(13), 3025–3047.

Freeman, R. (1995). Global internal stabilizability does not imply global external stabilizability for small sensor disturbances. *IEEE Transactions on Automatic Control* **40**(12), 2119–2122.

Galan, O., G. W. Barton and J. A. Romagnoli (2002). Robust control of a SAG mill. *Powder Technology* **124**(3), 264–271.

Gilbert, E. G. and K. T. Tan (1991). Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control* **36**(9), 1008–1020.

Grimm, G., M. J. Messina, S. E. Tuna and A. R. Teel (2004). Examples when nonlinear model predictive control is nonrobust. *Automatica* **40**(10), 1729–1738.

Hinde, A. (2007). Cumulative rates models for the comminution of ores in rotary mills. Private Communication. Internal Report MINTEK.

Hulbert, D. G. (1989). The state of the art in the control of milling circuits. In: *6th IFAC Symposium on Automation in Mining, Mineral and Metal Processing (Buenos Aires)*.

Hulbert, D. G. (2005). Models for mill circuit simulation. Private Communication. MINTEK.

Hull, D. G. (1997). Conversion of optimal control problems into parameter optimization problems. *Journal of Guidance, Control, and Dynamics* **20**(11), 57–60.

Jiang, Z. and Y. Wang (2001). Input-to-state stability for discrete-time nonlinear systems. *Automatica* **37**(6), 857–869.

Jiang, Z., I. M. Y. Mareels and Y. Wang (1996). A Lyapunov formulation of the nonlinear small-gain theorem for interconnected ISS systems. *Automatica* **32**(8), 1211–1215.

Kapakyulu, E. and M. H. Moys (2007). Modeling of energy loss to the environment from a grinding mill. Part I: Motivation, literature survey and pilot plant measurements. *Minerals Engineering* **20**(7), 646–652.

Karageorgos, J., P. Genovese and D. Baas (2006). Current trends in SAG and AG mill operability and control. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 191–206.

Kawajir, Y., C. Laird and A. Wachter (2006). *Introduction to Ipopt: A tutorial for downloading, installing, and using Ipopt, Revision: 799*. Carnegie Mellon University. Pittsburgh, PA, USA.

Kelly, E. G. and D. J. Spottiswood (1990). The breakage function; what is it really?. *Minerals Engineering* **3**(5), 405–414.

King, R. P. and F. Bourgeois (1993). Measurement of fracture energy during single-particle fracture. *Minerals Engineering* **6**(4), 353–367.

Kleinman, B. L. (1970). An easy way to stabilize a linear constant system. *IEEE Transactions on Automatic Control* **15**(12), 693.

Kothare, M. V., V. Balakrishnan and M. Morari (1996). Robust constrained model predictive control using linear matrix inequalities. *Automatica* **32**, 1361–1379.

Kouvaritakis, B., J. A. Rossiter and J. Schuurmans (2000). Efficient robust predictive control. *IEEE Transactions on Automatic Control* **45**(8), 145–159.

Langson, W., I. Chryssochoos, S. V. Rakovic and D. Q. Mayne (2004). Robust model predictive control using tubes. *Automatica* **40**, 125–133.

Latchireddi, S. and S. Morrell (2003*a*). Slurry flow in mills: Grate-only discharge mechanism (Part 1). *Minerals Engineering* **16**(7), 625–633.

Latchireddi, S. and S. Morrell (2003*b*). Slurry flow in mills: Grate-pulp lifter discharge systems (Part 2). *Minerals Engineering* **16**(7), 635–642.

Lazar, M., D. Munoz de la Pena, W. P. M. H. Heemels and T. Alamo (2008). On input-to-state stability of min-max nonlinear model predictive control. *Systems & Control Letters* **57**(1), 39–48.

Lee, E. B. and L. Markus (1967). *Foundations of optimal control theory*. Wiley. New York.

Lee, J. H. and Z. Yu (1997). Worst-case formulations of model predictive control for systems with bounded parameters. *Automatica* **33**(5), 763–781.

Lee, Y. I. and B. Kouvaritakis (2000). Robust receding horizon control for systems with uncertain dynamics and input saturation. *Automatica* **36**(10), 1497–1504.

Limon, D., J. M. Bravo, T. Alamo and E. F. Camacho (2005). Robust MPC of constrained nonlinear systems based on interval arithmetic. *IEE Proceedings of Control Theory and Applications* **152**(3), 325–332.

Limon, D., T. Alamo, F. Salas and E. F. Camacho (2006). Input to state stability of min-max MPC controllers for nonlinear systems with bounded uncertainties. *Automatica* **42**(5), 797–803.

Lougee-Heimer, R. (2003). The Common Optimization INterface for Operations Research. *IBM Journal of Research and Development* **47**(1), 57–66.

Loveday, B. K. and D. Naidoo (1997). Rock abrasion in autogenous milling. *Minerals Engineering* **10**(6), 603–612.

Loveday, B., R. Morrison, G. Henry and U. Naidoo (2006). An investigation of rock abrasion and breakage in a pilot-scale AG/SAG mill. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 379–388.

Lynch, A. J. (1979). *Mineral Crushing and Grinding Circuits: Their Simulation, Design, and Control (Developments in Mineral Processing Series, Vol 1)*. Elsevier Science Ltd. 335 Jan van Galenstraat, P.O. Box 211, Amsterdam, The Netherlands.

Ma, D. L. and R. D. Braatz (2001). Worst-case analysis of finite-time control policies. *IEEE Transactions on Control Systems Technology* **9**(5), 766–774.

Magni, L. and R. Sepulchre (1997). Stability margins of nonlinear receding-horizon control via inverse optimality. *Systems & Control Letters* **32**(4), 241–245.

Magni, L., D. M. Raimondo and R. Scattolini (2006). Regional input-to-state stability for nonlinear model predictive control. *IEEE Transactions on Automatic Control* **51**(9), 1548–1553.

Magni, L., G. De Nicolao, R. Scattolini and F. Allgöwer (2003). Robust model predictive control of nonlinear discrete-time systems. *International Journal of Robust and Nonlinear Control* **13**, 229–246.

Magni, L., H. Nijmeijer and A. J. van der Schaft (2001). A receding-horizon approach to the nonlinear H_∞ control problem. *Automatica* **37**(3), 429–435.

Marquez, H. J. and M. Riaz (2005). Robust state observer design with application to an industrial boiler system. *Control Engineering Practice* **13**(6), 713–728.

Mayne, D. Q., J. B. Rawlings, C. V. Rao and P. O. M. Scokaert (2000). Constrained model predictive control: Stability and optimality. *Automatica* **36**, 789–814.

McBride, A. and M. Powell (2006). A structured approach to modelling SAG mill liner wear – Numerical modelling of liner evolution. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 120–132.

Mhaskar, P. (2006). Robust model predictive control design for fault-tolerant control of process systems. *Industrial & Engineering Chemistry Research* **45**(25), 8565–8574.

Mhaskar, P. and A. B. Kennedy (2008). Robust model predictive control of nonlinear process systems: Handling rate constraints. *Chemical Engineering Science* **63**(2), 366–375.

Mhaskar, P., N. H. El-Farra and P. D. Christofides (2005). Robust hybrid predictive control of nonlinear systems. *Automatica* **41**(2), 209–217.

Mhaskar, P., N. H. El-Farra and P. D. Christofides (2006). Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control. *Systems & Control Letters* **55**(8), 650–659.

Michalska, H. and D. Q. Mayne (1993). Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control* **38**(11), 1623–1633.

Michalska, H. and D. Q. Mayne (1995). Moving horizon observers and observer-based control. *IEEE Transactions on Automatic Control* **40**(6), 995–1006.

Mishra, B. K. (2003*a*). A review of computer simulation of tumbling mills by the discrete element method: Part I – Contact mechanics. *International Journal of Mineral Processing* **71**(1–4), 73–93.

Mishra, B. K. (2003*b*). A review of computer simulation of tumbling mills by the discrete element method Part II – Practical applications. *International Journal of Mineral Processing* **71**(1–4), 95–112.

Morilla, F., F. Vázquez and J. Garrido (2008). Centralized PID control by decoupling for TITO processes. In: *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*. pp. 1318–1325.

Morrell, S. (1996). Power draw of wet tumbling mills and its relationship to charge dynamics – Part 1: A continuum approach to mathematical modelling of mill power draw. *Transactions of the Institution of Mining and Metallurgy Section C* **105**, C43—C53.

Morrell, S. (2004*a*). A new autogenous and semi-autogenous mill model for scale-up, design and optimisation. *Minerals Engineering* **17**(3), 437–445.

Morrell, S. (2004*b*). Predicting the specific energy of autogenous and semi-autogenous mills from small diameter drill core samples. *Minerals Engineering* **17**(3), 447–451.

Morrell, S. and I. Stephenson (1996). Slurry discharge capacity of autogenous and semi-autogenous mills and the effect of grate design. *International Journal of Mineral Processing* **46**(1–2), 53–72.

Morrell, S. and R. D. Morrison (1996). AG and SAG mill circuit selection and design by simulation. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 6-9 October*. Vol. 2. Vancouver, B. C., Canada. pp. 769–790.

Morrison, R., B. Loveday, M. Powell, N. Djordjevic and P. Cleary (2006). Applying Discrete Element Modelling to different modes of breakage in AG and SAG Mills. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 407–420.

Morrison, R. D. and P. W. Cleary (2004). Using DEM to model ore breakage within a pilot scale SAG mill. *Minerals Engineering* **17**(11–12), 1117–1124.

Morrison, R. D., F. Shi and R. Whyte (2007). Modelling of incremental rock breakage by impact – For use in DEM models. *Minerals Engineering* **20**(3), 303–309.

Nageswararao, K., D. M. Wiseman and T. J. Napier-Munn (2004). Two empirical hydrocyclone models revisited. *Minerals Engineering* **17**(5), 671–687.

Najim, K., D. Hodouin and A. Desbiens (1995). Adaptive control: State of the art and an application to a grinding process. *Powder Technology* **82**(1), 59–68.

Napier-Munn, T. J. and B. A. Wills (2006). *Wills' Mineral Processing Technology, Seventh Edition: An Introduction to the Practical Aspects of Ore Treatment and Mineral Recovery*. Butterworth-Heinemann. Linacre House, Jordan Hill, Oxford OX2 8DP, UK.

Napier-Munn, T. J., S. Morrell, R. D. Morrison and T. Kojovic (1996). Mineral comminution circuits their operation and optimisation. JKMRC Monograph Series.

Narayanan, S. S. (1987). Modelling the performance of industrial ball mills using single particle breakage data. *International Journal of Mineral Processing* **20**(3–4), 211–228.

Narayanan, S. S. and W. J. Whiten (1988). Determination of comminution characteristics from single particle breakage tests and its application to ball mill scale-up. *Transactions of the Institution of Mining and Metallurgy* **97**, C115–C124.

Neesse, T., V. Golyk, P. Kaniut and V. Reinsch (2004). Hydrocyclone control in grinding circuits. *Minerals Engineering* **17**(11–12), 1237–1240.

Pannocchia, G. (2004). Robust model predictive control with guaranteed setpoint tracking. *Journal of Process Control* **14**, 927–937.

Pannocchia, G. and E. C. Kerrigan (2003). Offset-free receding horizon control of constrained linear systems subject to time-varying setpoints and persistent unmeasured disturbances. Technical report. Department of Engineering, University of Cambridge. Cambridge, UK. CUED/F-INFENG/TR.468.

Pannocchia, G. and E. C. Kerrigan (2005). Offset-free receding horizon control of constrained linear systems. *AIChE Journal* **51**(12), 3134–3146.

Peng, H., Z. Yang, W. Gui, M. Wu, H. Shioya and K. Nakano (2007). Nonlinear system modeling and robust predictive control based on RBF-ARX model. *Engineering Applications of Artificial Intelligence* **20**(1), 1–9.

Peterka, V. (1984). Predictor-based self tuning control. *Automatica* **20**(1), 39–50.

Pomerleau, A., D. Hodouin, A. Desbiens and E. Gagnon (2000). A survey of grinding circuit control methods: From decentralized PID controllers to multivariable predictive controllers. *Powder Technology* **108**(2–3), 103–115.

Powell, M. S. and A. T. McBride (2004). A three-dimensional analysis of media motion and grinding regions in mills. *Minerals Engineering* **17**(11–12), 1099–1109.

Powell, M. S. and A. T. McBride (2006). What is required from DEM simulations to model breakage in mills?. *Minerals Engineering* **19**(10), 1013–1021.

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

Radhakrishnan, V. R. (1999). Model based supervisory control of a ball mill grinding circuit. *Journal of Process Control* **9**(3), 195–211.

Rajamani, R. K. and J. A. Herbst (1991*a*). Optimal control of a ball mill grinding circuit–II. Feedback and optimal control. *Chemical Engineering Science* **46**(3), 871–879.

Rajamani, R. K. and J. A. Herbst (1991*b*). Optimal control of a ball mill grinding circuit. I. Grinding circuit modeling and dynamic simulation. *Chemical Engineering Science* **46**(3), 861–870.

Ramasamy, M., S. S. Narayanan and C. D. P. Rao (2005). Control of ball mill grinding circuit using model predictive control scheme. *Journal of Process Control* **15**(3), 273–283.

Rao, C. V., J. B. Rawlings and D. Q. Mayne (2003). Constrained state estimation for nonlinear discrete-time systems: stability and moving horizon approximations. *IEEE Transactions on Automatic Control* **48**(2), 246–258.

Åström, K. J. (2002). *Control System Design*. Department of Mechanical and Environmental Engineering, University of California. Santa Barbara.

Åström, K. J. and P. Eykhoff (1971). System identification – A survey. *Automatica* **7**(2), 123–162.

Rawlings, J. B. and K. R. Muske (1993). Stability of constrained receding horizon control. *Transactions on Automatic Control* **38**(10), 1512–1516.

Richalet, J., A. Rault, J. L. Testud and J. Papon (1978). Model predictive heuristic control: Applications to industrial processes. *Automatica* **14**(5), 413–428.

Rossiter, J. A., B. Kouvaritakis and M. J. Rice (1998). A numerically robust state-space approach to stable-predictive control strategies. *Automatica* **34**(1), 65–73.

Sbarbaro, D., J. Barriga, H. Valenzuela and G. Cortes (2005). A multi-input-single-output Smith predictor for feeders control in SAG grinding plants. *IEEE Transactions on Control Systems Technology* **13**(6), 1069–1075.

Schuurmans, J. and J. A. Rossiter (2000). Robust model predictive control using tight sets of predicted states. In: *IEE Proceedings for Control Theory and Applications*. Vol. 147. pp. 13–18.

Scokaert, P. O. M. and D. Q. Mayne (1998). Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control* **43**(8), 1136–1142.

Scokaert, P. O. M., D. Q. Mayne and J. B. Rawlings (1999). Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control* **44**(3), 648–654.

Shi, F. and T. Kojovic (2007). Validation of a model for impact breakage incorporating particle size effect. *International Journal of Mineral Processing* **82**(3), 156–163.

Shi, F. N. and T. J. Napier-Munn (1996). A model for slurry rheology. *International Journal of Mineral Processing* **47**(1–2), 103–123.

Shi, F. N. and T. J. Napier-Munn (2002). Effects of slurry rheology on industrial grinding performance. *International Journal of Mineral Processing* **65**(3–4), 125–140.

Skogestad, S. (2003). Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control* **13**(4), 291–309.

Smith, V. C., D. G. Hulbert and A. Singh (2001). AG/SAG control and optimization with PlantStar 2000. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology held September 30 – October 3*. Vol. II. Vancouver, B. C., Canada. pp. 282–293.

Sontag, E. D. (1989). Smooth stabilization implies coprime factorization. *IEEE Transactions on Automatic Control* **34**(4), 435–443.

Sontag, E. D. (1990). Further facts about Input to State Stabilization. *IEEE Transactions on Automatic Control* **35**(4), 473–476.

Sontag, E. D.; Yuan Wang (1996). New characterizations of Input-to-State Stability. *IEEE Transactions on Automatic Control* **41**(9), 1283–1294.

Spurgeon, S. K. (2008). Sliding mode observers: a survey. *International Journal of Systems Science* **39**(8), 751–764.

Stanley, G. G. (1987). The extractive metallurgy of gold in South Africa. Technical Report Vol 1. South African Institute of Mining and Metallurgy. Johannesburg.

Tavares, L. M. and R. P. King (1998). Single-particle fracture under impact loading. *International Journal of Mineral Processing* **54**(1), 1–28.

Valenzuela, J., M. Bourassa, K. Najim and R. Del Villar (1994). Dynamic matrix control of an autogenous grinding circuit. *Minerals Engineering* **7**(1), 105–114.

Van Drunick, W. I. and B. P. N. Penny (2006). Expert mill control at Anglogold Ashanti. In: *Proceedings of an International Conference on Autogenous and Semiautogenous Grinding Technology, 23 – 27 September*. Vol. 3. Vancouver, B. C., Canada. pp. 266–281.

Vázquez, F. and F. Morilla (2002). Tuning decentralized PID controllers for MIMO systems with decouplers. In: *Proceedings of the 15th Triennial IFAC World Congress, Barcelona, Spain*. pp. 349–354.

Vogel, L. and W. Peukert (2003). Breakage behaviour of different materials – Construction of a mastercurve for the breakage probability. *Powder Technology* **129**(1–3), 101–110.

Vogel, L. and W. Peukert (2005). From single particle impact behaviour to modelling of impact mills. *Chemical Engineering Science* **60**(18), 5164–5176.

Wan, Z. and M. V. Kothare (2003). An efficient off-line formulation of robust model predictive control using linear matrix inequalities. *Automatica* **39**, 837–846.

Wang, Y. J. and J. B. Rawlings (2004*a*). A new robust model predictive control method I: Theory and computation. *Journal of Process Control* **14**, 231–247.

Wang, Y. J. and J. B. Rawlings (2004*b*). A new robust model predictive control method II: Examples. *Journal of Process Control* **14**(3), 249–262.

Wei, D. and I. K. Craig (2009). Grinding mill circuits – A survey of control and economic concerns. *International Journal of Mineral Processing* **90**(1-4), 56–66.

Whiten, W. J. (1974). A matrix theory of comminution machines. *Chemical Engineering Science* **29**(2), 589–599.

Wills, B. A. and T. J. Napier-Munn (2006). *Wills' Mineral Processing Technology: An introduction to the practical aspects of ore treatment and mineral recovery*. 7^{th} ed.. Elsevier, Butterworth-Heinemann. Linacre House, Jordan Hill, Oxford OX2 8DP, UK.

Xingyan, G., M. Zhizhong and W. Jian (1992). A predictive adaptive controller and its application to the control of the wet autogenous mill. In: *Proceedings of the IEEE International Symposium on Industrial Electronics, 1992.*. pp. 143–145 vol.1.

Xiong, Y. and M. Saif (2001). Sliding mode observer for nonlinear uncertain systems. *IEEE Transactions on Automatic Control* **46**(12), 2012–2017.

Yashima, S., Y. Kanda and S. Sano (1987). Relationships between particle size and fracture energy or impact velocity required to fracture as estimated from single particle crushing. *Powder Technology* **51**(3), 277–282.

Zheng, Z. Q. and M. Morari (1995). Stability of model predictive control with mixed constraints. *IEEE Transactions on Automatic Control* **40**(10), 1818–1823.

ADDENDUM A

SOFTWARE IMPLEMENTATION

A.1 INTRODUCTION

The robust nonlinear model predictive controller and nonlinear model predictive controller were implemented in C++ using open-source packages.

The first open-source package named IPOPT (Interior Point Optimiser, pronounced "I-P-Opt") (Kawajir *et al.*, 2006) does large-scale nonlinear parameter optimisation. It solves general nonlinear programming problems of the form

$$\min_{x\in\mathbb{R}^n} \quad f(x) \tag{A.1}$$

$$\text{s.t.} \quad g^L \leq g(x) \leq g^U \tag{A.2}$$

$$x^L \leq x \leq x^U \tag{A.3}$$

where $x \in \mathbb{R}^n$ are the optimisation variables (possibly with lower and upper bounds, $x^L \in (\mathbb{R}\cup\{-\infty\})^n$ and $x^U \in (\mathbb{R}\cup\{+\infty\})^n$), $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, and $g : \mathbb{R}^n \to \mathbb{R}^m$ are the general nonlinear constraints. The functions $f(x)$ and $g(x)$ can be linear or nonlinear and convex or non-convex, but should be twice continuously differentiable. The constraints have lower and upper bounds, $g^L \in (\mathbb{R}\cup\{-\infty\})^m$ and $g^U \in (\mathbb{R}\cup\{+\infty\})^m$. Equality constraints of the form $g_i(x) = \bar{g}_i$ can be specified by setting $g_i^L = g_i^U = \bar{g}_i$.

The optimisation problem (A.1)–(A.3) is rewritten in terms of the functions specified in Section (3.7.3). The nonlinear programming problem then becomes

$$\min_{X\in\mathbb{R}^{n_X}} \quad \varphi(X) \tag{A.4}$$

$$\text{s.t.} \quad g^L \leq \vartheta(x) \leq g^U \tag{A.5}$$

$$x^L \leq X \leq x^U \tag{A.6}$$

where $n_X \triangleq (n_x + n_u + n_D + n_{slack}) \cdot (N) + n_x + n_D$.

The second open-source package, named CPPAD (Lougee-Heimer, 2003), does automatic differentiation. Two classical ways of evaluating derivatives of functions on computers are to do symbolic differentiation or finite differences. Symbolic differentiation suffers from slow execution and the difficulty of converting a computer program to a single expression to be evaluated. Finite differences suffer from rounding errors in the discretisation process and cancellation. Automatic differentiation overcomes the mentioned problems by exploiting the fact that a computer program implementing the vector function $y = F(x)$ can generally be decomposed into a sequence of elementary assignments. Any one of the elementary assignments can be trivially differentiated by using a simple lookup table. The derivative can be constructed by applying the chain rule of differentiation to the elemental partial derivatives. This process yields exact (to numerical accuracy) derivatives.

A.2 IMPLEMENTATION

IPOPT requires some information about the nonlinear programming problem that it needs to solve. IPOPT provides a class *TNLP* that needs to be overridden to provide the nonlinear programming problem. The following information is required by IPOPT (Kawajir *et al.*, 2006) and member functions that need to be overridden and the return variables are provided in brackets:

1. Problem dimensions
    - Number of variables (get_nlp_info: Index &n)
    - Number of constraints (get_nlp_info: Index &m)
2. Problem bounds
    - Variable bounds (get_bounds_info: Number *x_l, Number *x_u)
    - Constraint bounds (get_bounds_info: Number *g_l, Number *g_u)
3. Initial starting point
    - Initial values for the primal X variables (get_starting_point: Number *x)
4. Problem structure
    - Number of non-zeros in the Jacobian of the constraints (get_nlp_info : Index &nnz_jac_g)
    - Sparsity structure of the Jacobian of the constraints (eval_jac_g : Index *iRow, Index *jCol)

5. Evaluation of problem functions

   - Objective function, $\varphi(X)$ (eval_f: Number &obj_value)
   - Gradient of the objective function, $\nabla\varphi(X)$ (eval_grad_f: Number *grad_f)
   - Constraint function values, $\vartheta(X)$ (eval_g: Number *g)
   - Jacobian of the constraints, $\nabla\vartheta(X)^T$ (eval_jac_g: Number *values)

The RNMPC is defined in Section 3.7. The basic definitions are:

- n_x — the number of states
- n_u — the number of manipulated variables
- n_c — the number of constraints
- n_p — the number of uncertain parameters
- T — the prediction horizon
- τ_s — the sampling time in seconds
- N — the number of steps over the prediction horizon, defined as $N = {}^{T}/_{\tau_s}$
- n_D — the number of entries in the $\mathbf{D}_i$ matrix, defined as $n_D \triangleq n_x \times n_p$
- n_{slack} — the number of slack variables, defined as $n_{slack} \triangleq (n_c + 1) \times n_p$

A.2.1 IPOPT TNLP class methods

A.2.1.1 Class constructor method `MyNLP::MyNLP`

This constructor method is called when a new instance of the class is created. This method does the following initialisations:

- Calculates the dimensions of X of equation (3.100), $\nabla\varphi(X)$ of equation (3.102), $\vartheta(X)$ of equation (3.103) and $\nabla\vartheta(X)$ of equation (3.107).
- Threading memory is allocated for the number of threads corresponding to the number of nodes. This memory is allocated whether threading is used or not. This allows all the nodes to be processed simultaneously on multi-processor and multi-core systems when threading is enabled.
- If threading is used, sets up the thread mutexes and signals.

- Allocates memory for the global arrays **`m_cost`**, **`m_costGrad`**, **`m_g`** and **`m_gJac`** and some arrays for storing the states, control moves, states setpoint, control moves setpoint, average control moves and optimisation variables needed for *warm startup*.
- Assigns the initial values for the states and control moves of X to **`m_xu`** and sets the entries corresponding to the **D**-matrix and slack variables (δ) to 0.0.
- Assigns some general values (such as the states setpoint, control moves setpoint, states and control moves weightings) to the thread memory of each node.
- If threading is not used, calls **`CppADInit`** for each node in order to initialise the CPPAD function objects and allocates memory for some miscellaneous variables required by that node in its associated thread memory.
- If threading is used, this method then creates the actual threads, with the same number of threads as the number of nodes.

A.2.1.2 Method `get_nlp_info`

The method **`get_nlp_info`** returns the following information on the nonlinear programming problem:

- **`n`**: (out), the number of variables in the problem (dimension of X) — For the RNMPC it is the dimension of X defined as $(n_x+n_u+n_D+n_{slack})\cdot N+n_x+n_D$.
- **`m`**: (out), the number of constraints in the problem (dimension of $\vartheta(X)$) — For the RNMPC it is defined as $(n_x+n_D+n_c+2\cdot n_{slack})\cdot N+n_x+n_D$.
- **`nnz_jac_g`**: (out), the number of nonzero entries in the Jacobian of the constraints (nonzero entries of $\nabla\vartheta(X)^T$) — For the RNMPC it is defined as $N(n_x+n_x(n_x+n_u))+N(n_D(2n_x+n_u+1))+N(n_c(n_x+2n_u+N))+2N(n_{slack}(2n_x+2n_u+1))+n_x+n_D-n_c\cdot n_u-2\cdot n_{slack}\cdot n_u$.
- **`nnz_h_lag`**: (out), the number of nonzero entries in the Hessian of the Lagrangian — This is not used by the RNMPC, because the quasi-Newton approximation of the Hessian is used.
- **`index_style`**: (out), the numbering style used in the row/column entries in the sparse matrix format (**`C_STYLE`**: indexes start at 0 is used in RNMPC, **`FORTRAN_STYLE`**: indexes start at 1).

All the values returned by this function are calculated in the class constructor. Only the calculated values are returned.

A.2.1.3 Method `get_bounds_info`

- **`n`**: (in), The number of variables in the problem (dimension of X).
- **`x_l`**: (out), The lower bounds x^L for X. For the RNMPC there is no lower bound on any of the variables and the lower bounds are therefore set to $-1.0e19$, which is similar to $-\infty$ in the program.
- **`x`**_u: (out), The upper bounds x^U for X. For the RNMPC there is no upper bound on any of the variables and the upper bounds are therefore set to $1.0e19$, which is similar to $+\infty$ in the program.
- **`m`**: (in), The number of constraints in the problem (dimension of $\vartheta(X)$).
- **`g_l`**: (out), The lower bounds g^L for $\vartheta(X)$. See details below for the RNMPC implementation.
- **`g`**_u: (out), The upper bounds g^U for $\vartheta(X)$. See details below for the RNMPC implementation.

The bounds on the constraints depend on whether it is an equality or an inequality constraint. Inequality constraints are all transformed to have only an upper bound of 0.0, except the second block of slack variables that only have a lower bound of 0.0. Equality constraints are all transformed to be equal to 0 by setting the lower and upper bound equal to 0. The constraints consist of N number of blocks with $n_x + n_D + n_c + 2 \cdot n_{slack}$ number of entries in each block. The last block only has $n_x + n_D$ number of entries. The nonlinear programming problem is described in (3.88)-(3.94). The upper and lower bounds for each block are as follows:

Constraint Function	Number of Entries	g^L	g^U
System Dynamics	n_x	0.0	0.0
Calculating **D**-matrix	n_D	0.0	0.0
User-defined state and input constraints on the system	n_c	$-1.0e19$	0.0
Slack variables for approximating the dual norms of the robust weighting terms	n_{slack}	$-1.0e19$	0.0
Slack variables for approximating the dual norms of the robust weighting terms	n_{slack}	0.0	$1.0e19$

A.2.1.4 Method `get_starting_point`

- **`n`**: (in), The number of variables in the problem (dimension of X).
- **`init_x`**: (in), if true, this method should provide an initial value for X.
- **`x`**: (out), The initial values for the primal variables. See below for details pertaining to the RNMPC.
- **`init_z`**: (in), if true, this method must provide an initial value for the bound multipliers z^L and z^U.
- **`z_L`**: (out), The initial values for the bound multipliers, z^L.
- **`z_U`**: (out), The initial values for the bound multipliers, z^U.
- **`m`**: (in), The number of constraints in the problem (dimension of $\vartheta(X)$).
- **`init_lambda`**: (in), if true, this method must provide an initial value for the constraint multipliers, λ.
- **`lambda`**: (out), The initial values for the constraint multipliers, λ.

The RNMPC initialises X with the initial state of the plant and the initial control moves and all the auxiliary variables to 0.0. The vector X consists of N blocks that each contains $n_x + n_u + n_D + n_{slack}$ variables. The last block contains only $n_x + n_D$ number of variables. The initial values for each block is given by:

Variables	Number of Entries	X
System state variables	n_x	Initial state
Control moves	n_u	Initial control moves
The **D**-matrix	n_D	0.0
Slack variables for approximating the dual norms of robust weighting terms	n_{slack}	0.0

These values should be returned when **`init_x`** is true. The initial values are calculated in the class constructor and the results returned when this method is called.

The values for **`z_l`**, **`z_u`** and **`lambda`** will only be requested if a *warm startup* is chosen. A *warm startup* can be used in RNMPC to initialise the system to the optimal solution of the previous time-step, which should be close to the optimal solution for the current time-step and therefore reduce the computational time needed to find the optimal solution for the current time-step. The values of **`z_l`**, **`z_u`** and **`lambda`** can be obtained when the optimisation of the previous time-step finishes and returns to the optimiser at the start of the current time-step using this method.

A.2.1.5 Method `eval_f`

This method returns the objective function value at the point X. For the RNMPC, this is the objective function of the controller.

- **`n`**: (in), The number of variables in the problem (dimension of X).
- **`x`**: (in), The values for the primal variables X, at which $\varphi(X)$ should be evaluated.
- **`new_x`**: (in), False if any of the other evaluation functions have been called with the same X values.
- **`obj_value`**: (out), The value of the objective function $\varphi(X)$.

This method calls **`sundials_run`** to calculate the objective function ($\varphi(X)$), the gradient of the objective function ($\nabla\varphi(X)$), the constraint function ($\vartheta(X)$) and the Jacobian of the constraints ($\nabla\vartheta(X)^T$). The method **`sundials_run`** only calculates a new solution for the functions if **`new_x`** is true, otherwise it only returns the previously calculated solution.

A.2.1.6 Method `eval_grad_f`

This method returns the gradient of the objective function at the point X, $\nabla\varphi(X)$.

- **`n`**: (in), The number of variables in the problem (dimension of X).
- **`x`**: (in), The values for the primal variables X, at which $\nabla\varphi(X)$ should be evaluated.
- **`new_x`**: (in), False if any of the other evaluation functions have been called with the same X values.
- **`grad_f`**: (out), The array containing the gradient of the objective function, $\nabla\varphi(X)$.

This method calls **`sundials_run`** to calculate the objective function ($\varphi(X)$), the gradient of the objective function ($\nabla\varphi(X)$), the constraint function ($\vartheta(X)$) and the Jacobian of the constraints ($\nabla\vartheta(X)^T$). The method **`sundials_run`** only calculates a new solution for the functions if **`new_x`** is true, otherwise it only returns the previously calculated solution.

A.2.1.7 Method `eval_g`

This method returns the value of the constraint function at the point X, $\vartheta(X)$.

- **`n`**: (in), The number of variables in the problem (dimension of X).
- **`x`**: (in), The values for the primal variables X, at which $\vartheta(X)$ should be evaluated.

- **new_x**: (in), False if any of the other evaluation functions have been called with the same X values.
- **m**: (in), The number of constraints in the problem, which is the dimension of $\vartheta(X)$.
- **g**: (out), The array containing the values of the constraint functions, $\vartheta(X)$.

This method calls **sundials_run** to calculate the objective function ($\varphi(X)$), the gradient of the objective function ($\nabla\varphi(X)$), the constraint function ($\vartheta(X)$) and the Jacobian of the constraints ($\nabla\vartheta(X)^T$). The method **sundials_run** only calculates a new solution for the functions if **new_x** is true, otherwise it only returns the previously calculated solution.

A.2.1.8 Method eval_jac_g

This method returns either the structure of the Jacobian of the constraints, or the values for the Jacobian of the constraints evaluated at point X.

- **n**: (in), The number of variables in the problem (dimension of X).
- **x**: (in), The values for the primal variables X, at which point $\nabla\vartheta(X)^T$ should be evaluated.
- **new_x**: (in), False if any of the other evaluation functions have been called with the same X values.
- **m**: (in), The number of constraints in the problem, which is the dimension of $\vartheta(X)$.
- **n_ele_jac**: (in), The number of nonzero elements in the Jacobian, which is the dimension of **iRow**, **jCol** and **values**.
- **iRow**: (out), The array containing the row indexes of the entries in the Jacobian of the constraints.
- **jCol**: (out), The array containing the column indexes of the entries in the Jacobian of the constraints.
- **values**: (out), The array containing the values of the entries in the Jacobian of the constraints.

If **values** is a NULL pointer, the structure of the Jacobian of the constraints is required. There are N blocks in the Jacobian of the constraints that look like Figure (3.1). The block in Figure (3.1) can be further subdivided into smaller blocks. The sub-blocks are populated using row major representation in the array, meaning that the column entries in the same row follow continuously in the array. The sparse structures of the sub-blocks are defined in the block structure in the order specified in Table A.1.

Table A.1: The definition of the sparse structure of the sub-blocks in the Jacobian of the constraints.

$\vartheta(X)$	X	Block Calculation	Block Size	Block Type	No Entries
States (s_i)	States (s_i)	$\frac{\partial s_i}{\partial s_i}$	$n_x \times n_x$	Diagonal	n_x
$\mathbf{D}_i$-matrix values	$\mathbf{D}_i$-matrix values	$\frac{\partial D_i}{\partial D_i}$	$n_D \times n_D$	Diagonal	n_D
States (s_{i+1})	States (s_i) and Inputs (q_i)	$\frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i}, \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial q_i}$	$n_x \times n_x \cdot n_u$	Rectangular	$n_x \times n_x \cdot n_u$
$\mathbf{D}_{i+1}$-matrix	States (s_i) and Inputs (q_i)	$\left(\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial s_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial s_i} \right), \left(\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial q_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial q_i} \right)$	$n_D \times n_x \cdot n_u$	Rectangular	$n_D \times n_x \cdot n_u$
$\mathbf{D}_{i+1}$-matrix	$\mathbf{D}_i$-matrix values	$\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial D_i}$	$n_D \times n_D$	Rectangular	$n_D \times n_x$
Inequality Constraints $(\theta_{1,i}, \ldots, \theta_{j,i}, \ldots \theta_{n_c,i})$	States (s_i) and Inputs (q_i)	$\frac{\partial \theta_{j,i}(s_i,q_i)}{\partial s_i}, \frac{\partial \theta_{j,i}(s_i,q_i)}{\partial q_i}$	$n_c \times n_x \cdot n_u$	Rectangular	$n_c \times n_x \cdot n_u$
Inequality Constraints $(\theta_{1,i}, \ldots, \theta_{j,i}, \ldots \theta_{n_c,i})$	Slack variables ($\delta_{j,i}$)	$\frac{\partial \theta_{j,i}(s_i,q_i) + e^T \delta_{j,i}}{\delta_{j,i}}$	$n_c \times n_{slack}$	Banded	$n_c \times n_p$
Slack variables $(\delta_{0,i}, \ldots, \delta_{j,i}, \ldots, \delta_{n_c,i})$ — Block 1 & 2	States (s_i) and Inputs (q_i)	$\frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,j,i}}{\partial s_i}(s_i,q_i) \right)^T}{\partial s_i}, \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{j,i}}{\partial s_i}(s_i,q_i) \right)^T}{\partial q_i}$	$n_{slack} \times n_x \cdot n_u$	Rectangular	$n_{slack} \times n_x \cdot n_u$
Slack variables $(\delta_{0,i}, \ldots, \delta_{j,i}, \ldots, \delta_{n_c,i})$ — Block 1 & 2	$\mathbf{D}_{i+1}$-matrix values	$\frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i) \right)^T}{\partial D_{i+1}}$	$n_{slack} \times n_x$	Rectangular	$n_{slack} \times n_x$
Slack variables $(\delta_{0,i}, \ldots, \delta_{j,i}, \ldots, \delta_{n_c,i})$ — Block 1 & 2	Slack variables ($\delta_{j,i}$)	$\frac{\partial \delta_{j,i}}{\partial \delta_{j,i}}$	$n_{slack} \times n_{slack}$	Diagonal	n_{slack}

If **values** is not a NULL pointer then the values of the Jacobian of the constraints are required and this method calls **sundials_run** to calculate the objective function ($\varphi(X)$), the gradient of the objective function ($\nabla\varphi(X)$), the constraint function ($\vartheta(X)$) and the Jacobian of the constraints ($\nabla\vartheta(X)^T$). The method **sundials_run** only calculates new a solution for the functions if **new_x** is true, otherwise only returns the previously calculated solution.

A.2.1.9 Method `finalize_solution`

This method returns the solution of the nonlinear programming problem.

- **status**: (in), Gives the final status of the solution such as SUCCESS or some failure.
- **n**: (in), The number of variables in the problem (dimension of X).
- **x**: (in), The final values for the primal variables X.
- **z_L**: (in), The final values for the bound multipliers, z^L.
- **z_U**: (in), The final values for the bound multipliers, z^U.
- **m**: (in), The number of constraints in the problem (dimension of $\vartheta(X)$).
- **lambda**: (in), The final values for the constraint multipliers, λ.
- **obj_value**: (in), The final value of the objective function $\varphi(X)$.

The values of **x** are saved for the next time-step. The values for **z_L**, **z_U** and **lambda** can be saved to be used in a *warm startup* of the next time-step.

A.2.2 RNMPC specific methods

These methods are not part of the base TNLP class, but are added to do the required calculations specifically geared towards RNMPC implementation.

A.2.2.1 Method `next_run`

This method sets up the optimisation problem for the next time-steps. Most of the initialisation is done in the class constructor and remains valid for all time-step. Only a few values need to change between iterations and this method makes it possible.

- **xSetpoint**: (in), Gives the setpoint that the controller should follow in this iteration. The setpoint can only be changed between iterations, not during an iteration.
- **xMeasured**: (in), The actual state of the process being controlled as measured or estimated online.

- **uPrevious**: (in), Provides the control moves of the previous time-step if $\Delta u(t)$ control is used rather than absolute control $u(t)$.
- **uAverage**: (out), The average control moves over a period of time.
- **objective**: (out), The objective value of the previous time-step.
- **Contraction**: (out), The amount the constraints are tightened to provide robustness.

A.2.2.2 Method get_u

This method returns the first control moves u_0^{opt}, which are part of the solution of the nonlinear programming problem.

- **u**: (out), The first control moves u_0^{opt} from the solution of the nonlinear programming problem.

A.2.2.3 Method SundialsRun

This method is called by **eval_f**, **eval_grad_f**, **eval_g** and **eval_jac_g**, to calculate the objective function ($\varphi(X)$), the gradient of the objective function ($\nabla\varphi(X)$), the constraint function ($\vartheta(X)$) and the Jacobian of the constraint function ($\nabla\vartheta(X)$). This method starts by calculating $x_0 - s_0$ and $-D_0$ of (3.104).

This method then sets up the threads to calculate $\varphi(X)$, $\nabla\varphi(X)$,$\vartheta(X)$ and $\nabla\vartheta(X)^T$ for each node by assigning the relevant pointers for the node and instructing the threads to execute. The threads can execute in parallel on multi-core or multi-processor systems, allowing the controller to perform faster on systems with more processors and/or processor cores. The method then waits for the threads to finish before continuing. An example of assigning pointers from the global arrays for the thread solving node 1 is shown in Figure (A.1) and Figure (A.2). The constraint function $\vartheta(X)$ can be subdivided into $N+1$ blocks (3.104)-

(3.106), with the calculations in the block (3.104) defined as:

$$G_{s_0} \triangleq x_k - s_0 \tag{A.7}$$

$$G_{D_0} \triangleq -I_{n_x} D_0 + 0 \tag{A.8}$$

$$G_{c_0} \triangleq \begin{cases} \theta_{1,0}(s_0, q_0) + e^T \delta_{1,0} \\ \vdots \\ \theta_{n_c,0}(s_0, q_0) + e^T \delta_{n_c,0} \end{cases} \tag{A.9}$$

$$G_{\delta 1_0} \triangleq \begin{cases} D_1^T \left(\frac{\partial L_0}{\partial s_0}(s_0, q_0) \right)^T - \delta_{0,0} \\ D_1^T \left(\frac{\partial \theta_{1,0}}{\partial s_0}(s_0, q_0) \right)^T - \delta_{1,0} \\ \vdots \\ D_1^T \left(\frac{\partial \theta_{n_c,0}}{\partial s_0}(s_0, q_0) \right)^T - \delta_{n_c,0} \end{cases} \tag{A.10}$$

$$G_{\delta 2_0} \triangleq \begin{cases} D_1^T \left(\frac{\partial L_0}{\partial s_0}(s_0, q_0) \right)^T + \delta_{0,0} \\ D_1^T \left(\frac{\partial \theta_{1,0}}{\partial s_0}(s_0, q_0) \right)^T + \delta_{1,0} \\ \vdots \\ D_1^T \left(\frac{\partial \theta_{n_c,0}}{\partial s_0}(s_0, q_0) \right)^T + \delta_{n_c,0} \end{cases} \tag{A.11}$$

the calculations in the blocks (3.105),$i = 1, \ldots, N-1$ defined as

$$G_{s_i} \triangleq f_{i-1}(s_{i-1}, q_{i-1}, \bar{p}) - s_i \tag{A.12}$$

$$G_{D_i} \triangleq \frac{\partial f_{i-1}(s_{i-1}, q_{i-1}, \bar{p})}{\partial s_{i-1}} \cdot D_{i-1} - I_{n_x} \cdot D_i + \frac{\partial f_{i-1}(s_{i-1}, q_{i-1}, \bar{p})}{\partial p} \tag{A.13}$$

$$G_{c_i} \triangleq \begin{cases} \theta_{1,i}(s_i, q_i) + e^T \delta_{1,i} \\ \vdots \\ \theta_{n_c,i}(s_i, q_i) + e^T \delta_{n_c,i} \end{cases} \tag{A.14}$$

$$G_{\delta 1_i} \triangleq \begin{cases} D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i, q_i) \right)^T - \delta_{0,i} \\ D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i, q_i) \right)^T - \delta_{1,i} \\ \vdots \\ D_{i+1}^T \left(\frac{\partial \theta_{n_c,i}}{\partial s_i}(s_i, q_i) \right)^T - \delta_{n_c,i} \end{cases} \tag{A.15}$$

$$G_{\delta 2_i} \triangleq \begin{cases} D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i, q_i) \right)^T + \delta_{0,i} \\ D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i, q_i) \right)^T + \delta_{1,i} \\ \vdots \\ D_{i+1}^T \left(\frac{\partial \theta_{n_c,i}}{\partial s_i}(s_i, q_i) \right)^T + \delta_{n_c,i} \end{cases} \tag{A.16}$$

and the calculations in the block (3.106) defined as

$$G_{s_N} \triangleq f_{N-1}(s_{N-1}, q_{N-1}, \bar{p}) - s_N \tag{A.17}$$

$$G_{D_N} \triangleq \frac{\partial f_{N-1}(s_{N-1}, q_{N-1}, \bar{p})}{\partial s_{N-1}} \cdot D_{N-1} - I_{n_x} \cdot D_N + \frac{\partial f_{N-1}(s_{N-1}, q_{N-1}, \bar{p})}{\partial p} \tag{A.18}$$

The sub-blocks in Figure (A.2) for $\nabla\vartheta(X)$ are stored sequentially in **`m_gJac`** in the following order

- **`m_eqJacEye`**,
- **`m_DJacEye`**,
- **`m_eqJacXU`**,
- **`m_DJacXU`**,
- **`m_DJacD`**,
- **`m_constraintsJacXU`**,
- **`m_constraintsJacSlack`**,
- **`m_slackJacXU`**,
- **`m_slackJacD`**,
- **`m_slackJacSlack`**,
- **`m_slackJacXU2`**,
- **`m_slackJacD2`**,
- **`m_slackJacSlack2`**.

After the threads have finished executing, the method assigns values to the final sub-blocks

$$\nabla f_N(x) \triangleq \frac{\partial \varphi(X)}{\partial s_N} \tag{A.19}$$

$$\nabla G_{s_N} \triangleq \frac{\partial G_{s_N}}{\partial s_N} \tag{A.20}$$

$$\nabla G_{D_N} \triangleq \frac{\partial G_{D_N}}{\partial D_N} \tag{A.21}$$

.

Figure A.1: Pointers to the global arrays representing X, $\varphi(X)$ and $\nabla\varphi(X)$ for the thread that processes node 1.

$\texttt{m\_g} = \vartheta(X)$																						
Node 0					Node 1					Node 2						Node $N-1$					Terminal Node N	
G_{s_0}	G_{D_0}	G_{c_0}	$G_{\delta 1_0}$	$G_{\delta 2_0}$	G_{s_1}	G_{D_1}	G_{c_1}	$G_{\delta 1_1}$	$G_{\delta 2_1}$	G_{s_2}	G_{D_2}	G_{c_2}	$G_{\delta 1_2}$	$G_{\delta 2_2}$	$\cdots$	$G_{s_{N-1}}$	$G_{D_{N-1}}$	$G_{c_{N-1}}$	$G_{\delta 1_{N-1}}$	$G_{\delta 2_{N-1}}$	G_{s_N}	G_{D_N}
							$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$											
							m_constraints	m_slack	m_slack2	m_eq	m_D											
Pointers to the global array **m_g** for the thread processing block 1																						

$\vartheta(X)$	$\texttt{m\_gJac} = \nabla\vartheta(X)$							
G_{s_1}	m_eqJacEye							
G_{D_1}			m_DJacEye					
G_{c_1}	m_constraintsJacXU			m_constraintsJacSlack				
$G_{\delta 1_1}$	m_slackJacXU		m_slackJacD	m_slackJacSlack				
$G_{\delta 2_1}$	m_slackJacXU2		m_slackJacD2	m_slackJacSlack2				
G_{s_2}	m_eqJacXU							
G_{D_2}	m_DJacXU		m_DJacD					
G_{c_2}								
$G_{\delta 1_2}$								
$G_{\delta 2_2}$								
X	s_1	q_1	D_1	δ_1	s_2	q_2	D_2	δ_2
Pointers to the global array **m_gJac** for the thread processing node 1								

Figure A.2: Pointers to the global arrays representing $\vartheta(X)$ and $\nabla\vartheta(X)$ for the thread that processes node 1.

A.2.2.4 Method `CppADThreadRun`

This method mainly encapsulates method **`CppADRun`** with the thread-handling code. When the thread is created in the class constructor, this method calls **`CppADInit`** to construct the CPPAD function objects that will be used by **`CppADRun`** to evaluate the objective function $\varphi(X)$, gradient of the objective function $\nabla\varphi(X)$, the constraint function $\vartheta(X)$ and the Jacobian of the constraint function $\nabla\vartheta(X)$.

This method locks onto **`g_thread_mutex_wait`** and waits for a signal from **`SundialsRun`** through **`g_thread_condition_wait`** to execute, after which this thread unlocks the **`g_thread_mutex_wait`** mutex, executes **`CppADRun`** and then attempts to lock the **`g_thread_mutex_wait`** again and then wait for a signal from **`SundialsRun`** to execute.

A.2.2.5 Method `CppADNoThreadRun`

If non-threaded execution is preferred, this method will just call **`CppADRun`**, without any thread-handling code. To use this method, **`CppADInit`** must be called from the class constructor before calling this method.

A.2.2.6 Method `CppADInit`

This method initialises the CPPAD object that will be used to evaluate $\varphi(X)$, $\nabla\varphi(X)$, $\vartheta(X)$ and $\nabla\vartheta(X)$.

It creates an instance **`Plant`** of the **`ProcessPlantODE`** class. The **`ProcessPlantODE`** class must have a member function called **`Ode`** that takes the following parameters:

- **`t`**: (in), The independent variable of the function to integrate.
- **`x`**: (in), The dependent variable of the function to integrate.
- **`f`**: (out), The change of the dependent variable with time ($\frac{\partial x}{\partial t}$).

The **`ProcessPlantODE`** class has some custom methods that allow additional values needed by the nonlinear model and cost function to be passed to the class:

- **`SetParams`**: Passes the parameter values needed by the nonlinear model.
- **`SetControls`**: Passes the value of the control moves to the nonlinear model.
- **`SetCost`**: Passes all the weighting and scaling matrices as well as the setpoint vectors for the states and the inputs needed by the cost function.

This method then calls

```
CppAD::Independent(xupuset);
```

that flags CPPAD to start "taping" the calculations. Once the taping is done, a CPPAD function object is created that will be used to evaluate the function and derivatives of the function such as the Jacobian and the Hessian. Only one variable can be declared the independent variable by CPPAD, and to capture all the independent variables of interest for the nonlinear model, the objective function and the constraint functions, a compound variable **xupuset** is defined that contains x, u, p, the weighting and scaling matrices and the setpoint vectors for the states and inputs for the objective function. The independent variable here is all the variables that one would like to differentiate against. Other variables can also be used during taping, but these variables will become constants in the CPPAD function object preventing them from changing in the future. The setpoints are therefore added to the independent variable, which allows the setpoints to be changed, without reconstructing the CPPAD function object through another "taping" with modified setpoint values.

The nonlinear model is then integrated to capture the calculations for the CPPAD function object by calling

```
xf = CppAD::Runge45(Plant, M, ti, tf, xi, e);
```

where **xf** is the final state of the nonlinear model and the objective function after the integration period (**tf** – **ti**), **M** is the number of steps over the integration period, **ti** is the starting time of the integration period, **tf** is the final time of the integration period, **xi** is the initial state of the nonlinear model and **e** is the absolute errors of the integration for each entry in **xf**.

The constraints pertaining to the limitations of the process variables are "taped" by evaluating the call to

```
Constraints(constraints, xf, u, deltaU);
```

where **constraints** is the solution of the constraint functions, **xf** is the final state of the nonlinear system for the current time-step, **u** is the constant control moves applied during the current time-step and **deltaU** is used to evaluate constraints on the change in control moves between the previous and current time-step.

The solutions of the nonlinear model, objective function and the constraint functions are combined into one variable **xf_constraints**, because CPPAD only allows one variable to be declared the dependent variable. The "taping" is stopped and the CPPAD function object is created by the call

```
thread->f.Dependent(xupuset, xf_constraints);
```

that assigns the CPPAD function object to the **f** object of the thread. Memory is preallocated

to allow the evaluation of third order derivatives by the call to

```
thread->f.capacity_taylor(4);
```

A similar CPPAD function object is created for the outputs of the nonlinear model (such as the PSE and SLEV), rather than the internal states by the following calls

```
CppAD::Independent(xup);
ProcessPlant(y, x_dot, x, u, params, paramsConst);
thread->fy.Dependent(xup, y);
thread->fy.capacity_taylor(3);
```

that assigns the CPPAD function object to the **fy** object of the thread and preallocates memory for second order derivatives of the CPPAD function object.

A.2.2.7 Method CppADRun

This method evaluates the objective function $\varphi(X)$, gradient of the objective function $\nabla\varphi(X)$, the constraint function $\vartheta(X)$ and the Jacobian of the constraint function $\nabla\vartheta(X)$ for one block. This method is usually called by $N+1$ threads simultaneously, each evaluating a different block, for parallel computing of the functions. Parallel execution of this function can lead to increased speed on multi-core and multi-processor systems.

This method starts by converting the percentage uncertainty of the variable parameters to upper and lower bounds on the parameter.

The method then evaluates $s_{i+1} = f_{\text{model}}(s_i, q_i, p)$, which is the integral of the nonlinear model over one time-step, the objective function and the constraint function by calling

```
thread->m_f = thread->f.Forward(0, thread->m_xupu);
```

where **m_xupu** is an array containing s_i, q_i and p. The 0 indicates that this is the normal function evaluation and not a derivative. The first n_x entries in **thread->m_f** is $s_{i+1} = f_{\text{model}}(s_i, q_i, p)$, the next entry is the objective function value and the next n_c entries is the solution of the constraint functions.

The method then evaluates $\nabla f_{\text{model}}(s_i, q_i, p)$, the gradient of the objective function and the Jacobian of the constraints by calling

```
thread->m_fJacTemp = thread->f.ForOne(thread->m_xupu, i);
for i = 1,...,n_x+n_u+n_p
```

The **ForOne** indicates that this is a first-order derivative of the integral of the nonlinear model over one time-step, the objective function and the constraint functions with regard to state (s_i), inputs (q_i) and parameters (p).

The method then evaluates $\nabla^2 f_{\text{model}}(x,u,p)$, the Hessian of the objective function and the Hessian of the constraint functions by calling

```
thread->m_fHessian = thread->f.ForTwo(thread->m_xupu, x1, x2);
```

The **ForTwo** indicates that this is a second-order derivative of the integral of the nonlinear model functions over one time-step, the objective function and the constraint functions with regard to state (s_i), inputs (q_i) and parameters (p).

The rest of the method extracts the results of the function and derivative evaluations above and assigns it to the correct variables. To understand how the values are extracted from the results, the packing of the results should be understood. Given a function $J : \mathbb{R}^n \rightarrow \mathbb{R}^p$ such that $y = J(x)$, where $y \in \mathbb{R}^p$ and $x \in \mathbb{R}^n$, and the Jacobian is $\frac{\partial J(x)}{\partial x}$. Given a function $H : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p$ such that $y = H(x,u)$, where $u \in \mathbb{R}^m$ then the Hessian is given by $\frac{\partial^2 H(x,u)}{\partial u \partial x}$. The packing of the Jacobian and the Hessian is show in Figure (A.3).

This method calls **RobustWeight** to form the robust weighting term from the current slack variable values for the objective function and the inequality constraints on the states and inputs that describe the limitations of the process variables.

This method calls the macro **JAC_MATRIX** to return a pointer to the starting point of the relevant Jacobian block and calls the macro **HESSIAN_MATRIX** to return a pointer to the relevant starting point of the required Hessian block. The **JAC_MATRIX** macro takes the variable pointer to the array, the required row and column entry and the number of column entries in a row. The **HESSIAN_MATRIX** macro takes the variable pointer to the array, the required output entry, the entry of the first derivative vector, the entry of the second derivative vector, the number of entries in the first derivative vector and the number of entries in the second derivative vector.

This method assigns the solutions in the following order:

- **m_eq**: Calls

  ```
  thread->m_eq[i] = f[i] - thread->curXU[VAR_BLOCK + i];
  ```

  that returns $f_{i-1}(s_{i-1}, q_{i-1}, \bar{p}) - s_i$.

- **m_eqJacEye**: The non-zeros entries of $-I_{n_x}$.

- **m_eqJacXU**: Copy the result of $(\frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i}, \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial q_i})$ by calling

  ```
  memcpy(&thread->m_eqJacXU[nIndex], &fJac[NXUPU*i], sizeof(double)*NXU);
  ```

- The solutions of the inequality constraint functions describing the limitations of the system process variables.

	Jacobian Packing in Memory											
	y_1			y_2			$\cdots$	y_p				
	x_1	$\cdots$	x_n	x_1	$\cdots$	x_n		x_1	$\cdots$	x_n		
Array Entries	$\frac{\partial J_1}{\partial x_1}$	$\cdots$	$\frac{\partial J_1}{\partial x_n}$	$\frac{\partial J_2}{\partial x_1}$	$\cdots$	$\frac{\partial J_2}{\partial x_n}$	$\cdots$	$\frac{\partial J_p}{\partial x_1}$	$\cdots$	$\frac{\partial J_p}{\partial x_n}$		

	Hessian Packing in Memory																							
	y_1							y_2							$\cdots$	y_p								
	u_1			u_m				u_1			u_m					u_1			u_m				
	x_1	$\cdots$	x_n	x_1	$\cdots$	x_n		x_1	$\cdots$	x_n	x_1	$\cdots$	x_n			x_1	$\cdots$	x_n	x_1	$\cdots$	x_n			
Array Entries	$\frac{\partial^2 H_1}{\partial u_1 \partial x_1}$	$\cdots$	$\frac{\partial^2 H_1}{\partial u_1 \partial x_n}$	$\frac{\partial^2 H_1}{\partial u_m \partial x_1}$	$\cdots$	$\frac{\partial^2 H_1}{\partial u_m \partial x_n}$	$\cdots$	$\frac{\partial^2 H_2}{\partial u_1 \partial x_1}$	$\cdots$	$\frac{\partial^2 H_2}{\partial u_1 \partial x_n}$	$\frac{\partial^2 H_2}{\partial u_m \partial x_1}$	$\cdots$	$\frac{\partial^2 H_2}{\partial u_m \partial x_n}$	$\cdots$	$\cdots$	$\frac{\partial^2 H_p}{\partial u_1 \partial x_1}$	$\cdots$	$\frac{\partial^2 H_p}{\partial u_1 \partial x_n}$	$\frac{\partial^2 H_p}{\partial u_m \partial x_1}$	$\cdots$	$\frac{\partial^2 H_p}{\partial u_m \partial x_n}$		

Figure A.3: Jacobian and Hessian packing in memory.

– **m_constraints**: Copies the result from of the evaluation of the constraint functions ($\theta_{j,i}(s_i,q_i)$) and then adds the robust weighting terms ($e^T\delta_{j,i}$) by calling **RobustWeight**. Calling

```
thread->m_constraints[i] = constraints[i] +
RobustWeight(thread->curSlack + NP*(i+1));
```

returns $\theta_{j,i}(s_i,q_i)+e^T\delta_{j,i},\ j=1,\ldots,n_c$.

– **m_constraintsJacXU**: Stores the Jacobian of the inequality constraint functions relative to the states (s_i) and the inputs (q_i) by calling

```
memcpy(&thread->m_constraintsJacXU[iRow*NXU],
&constraintsJac[NXUPU*iRow], sizeof(double)*NXU);
```

that returns $\frac{\partial\theta_{j,i}(s_i,q_i)}{\partial s_i}$ and $\frac{\partial\theta_{j,i}(s_i,q_i)}{\partial q_i}$, $j=1,\ldots,n_c$.

– **m_constraintsJacSlack**: The Jacobian of the inequality constraints relative to the slack variables (δ_{ji}) that basically returns $1_{n_p}^T \triangleq (1,\ldots,1)^T \in \mathbb{R}^{n_p}$.

- The solutions of the constraint functions pertaining to the $\mathbf{D}_i$-matrix.

– **m_D**: It first calculates ($\frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial p} - I_{n_x}\cdot D_i$) by calling

```
JAC_MATRIX(thread->m_D, iRow, jRow, NP) =
JAC_MATRIX(fJac, iRow, NXU+jRow, NXUPU) -
JAC_MATRIX(thread->nextD, iRow, jRow, NP);
```

and then adds $\frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial s_{i-1}}\cdot D_{i-1}$ to the answer by calling

```
JAC_MATRIX(thread->m_D, iRow, jRow, NP) +=
JAC_MATRIX(fJac, iRow, nX, NXUPU) *
JAC_MATRIX(thread->curD, nX, jRow, NP);
```

that returns $\frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial s_{i-1}}\cdot D_{i-1} - I_{n_x}\cdot D_i + \frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial p}$.

– **m_DJacXU**: This is the Jacobian of the constraints relating to the $\mathbf{D}_i$-matrix relative to the states (s_i) and the inputs (q_i). It first assigns $\frac{\partial\frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial s_i}$ by calling

```
JAC_MATRIX(thread->m_DJacXU, NP*iRow+jRow, iCol, NXU) =
HESSIAN_MATRIX(fHessian, iRow, NXU+jRow, iCol, NXUP, (NXU+NU));
```

and then adds $\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial s_i}$ by calling

```
JAC_MATRIX(thread->m_DJacXU, NP*iRow+jRow, iCol, NXU) +=
HESSIAN_MATRIX(fHessian, iRow, nX, iCol, NXUP, (NXU+NU)) *
JAC_MATRIX(thread->curD, nX, jRow, NP);
```

that returns $\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial s_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial s_i}$, which also includes $\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial q_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial q_i}$.

- **m_DJacD**: This is the Jacobian of the constraints relating to the $\mathbf{D}_{i+1}$-matrix relative to the the $\mathbf{D}_i$-matrix. It retrieves the result by calling

```
JAC_MATRIX(thread->m_DJacD, NP*iRow+jRow, nX, NX) =
JAC_MATRIX(fJac, iRow, nX, NXUPU);
```

that returns $\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial D_i}$.

- **m_DJacEye**: This is the non-zero entries in $-\mathbb{I}_{n_D}$.

- The solutions of the constraint functions pertaining to the slack variables.

  - **m_slack**, **m_slack2**: The solutions of the constraints functions pertaining to the slack variables. Calls

```
for (nX = 0; nX < NX; nX++)
 JAC_MATRIX(thread->m_slack, iRow, jRow, NP) +=
JAC_MATRIX(thread->nextD, nX, jRow, NP) *
JAC_MATRIX(costGrad, iRow, nX, NXUP) * box_weighting[jRow];

JAC_MATRIX(thread->m_slack2, iRow, jRow, NP) =
JAC_MATRIX(thread->m_slack, iRow, jRow, NP);
```

    and subtracts the current slack variable values from **m_slack** and adds the current slack variable values to **m_slack2** by calling

```
JAC_MATRIX(thread->m_slack, iRow, jRow, NP) -=
JAC_MATRIX(thread->curSlack, iRow, jRow, NP);

JAC_MATRIX(thread->m_slack2, iRow, jRow, NP) +=
JAC_MATRIX(thread->curSlack, iRow, jRow, NP);
```

    that returns $G_{\delta 1_i}$ (A.15) and $G_{\delta 2_i}$ (A.16).

  - **m_slackJacXU**, **m_slackJacXU2**: This is the Jacobian of the constraint functions pertaining to the slack variables relative to the states (s_i) and the inputs (q_i). It calls

```
JAC_MATRIX(thread->m_slackJacXU, NP*iRow+jRow, iCol, NXU) = 0.0;

for (nX = 0; nX < NX; nX++)
JAC_MATRIX(thread->m_slackJacXU, NP*iRow+jRow, iCol, NXU) +=
JAC_MATRIX(thread->nextD, nX, jRow, NP) *
HESSIAN_MATRIX(fHessian, NX+iRow, nX, iCol, NXUP, (NXU+NU)) *
box_weighting[jRow];
```

that return

$$\begin{matrix} \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_0} & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} \\ \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} \\ \vdots & \vdots \\ \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{n_C,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{n_C,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} \end{matrix}$$

to **m_slackJacXU** and because **m_slackJacXU2** = **m_slackJacXU** calls

```
JAC_MATRIX(thread->m_slackJacXU2, NP*iRow+jRow, iCol, NXU) =
JAC_MATRIX(thread->m_slackJacXU, NP*iRow+jRow, iCol, NXU);
```

– **m_slackJacD**, **m_slackJacD2**: This is the Jacobian of the constraint functions pertaining to the slack variables relative to the $\mathbf{D}_{i+1}$-matrix. It calls

```
JAC_MATRIX(thread->m_slackJacD, NP*iRow+jRow, nX, NX) =
JAC_MATRIX(costGrad, iRow, nX, NXUP)*box_weighting[jRow];
```

that returns

$$\begin{matrix} \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\ \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\ \vdots \\ \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{n_C,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \end{matrix}$$

to **m_slackJacD** and because **m_slackJacD2** = **m_slackJacD** calls

```
JAC_MATRIX(thread->m_slackJacD2, NP*iRow+jRow, nX, NX) =
JAC_MATRIX(thread->m_slackJacD, NP*iRow+jRow, nX, NX);
```

– **m_slackJacSlack**, **m_slackJacSlack2**: This is the Jacobian of the constraint functions pertaining to the slack variables (δ_{ji}) relative to the slack variables (δ_{ji}). The non-zero entries of $-\mathbb{I}_{n_{slack}}$ are returned to **m_slackJacSlack** and the non-zero entries of $\mathbb{I}_{n_{slack}}$ are returned to **m_slackJacSlack2**.

A.2.3 Main execution loop

The main execution loop is implemented in **Robust_NMPC.cpp**. Figure A.4 shows the important calls in the main execution loop and Figure A.5 shows the program flow during optimisation.

- The program starts by reading in the simulation scenario file (**FileInName**) and opening the simulation solution file (**FileOutName**) from the file names passed in as arguments to the program.
- Memory is allocated for the
  - initial state (**x0 (NX)**),
  - state setpoint (**x_setpoint (NX)**),
  - steady state values of the control moves (**u_setpoint (NU)**),
  - scaling of the states for the objective function (**x_scale (NX)**),
  - scaling of the control moves for the objective function (**u_scale (NU)**),
  - control move returned by the nonlinear optimisation problem for the current time-step (**u (NU)**),
  - output of the system such as the PSE and SLEV for the milling circuit (**y (NY)**),
  - final state of the current time-step after simulation (**xf (NX)**),
  - disturbances on the control moves (**u_dist**),
  - time-varying parameter vector (**p_dist**), and
  - objective function solution (**obj**).
- The program reads in the values of the above-mentioned variables from the simulation scenario file.
- An instance of the **MyNLP** class is created that contains the definition of the nonlinear programming problem to solve with a call to

  ```
  SmartPtr<TNLP> mynlp = new MyNLP(x_setpoint, u_setpoint, x_scale,
  u_scale, x0);
  ```

  that also initialises the class with the state setpoint, steady state values of the control moves, the scaling factors for the states, control moves used in the objective function and the initial values for the states.

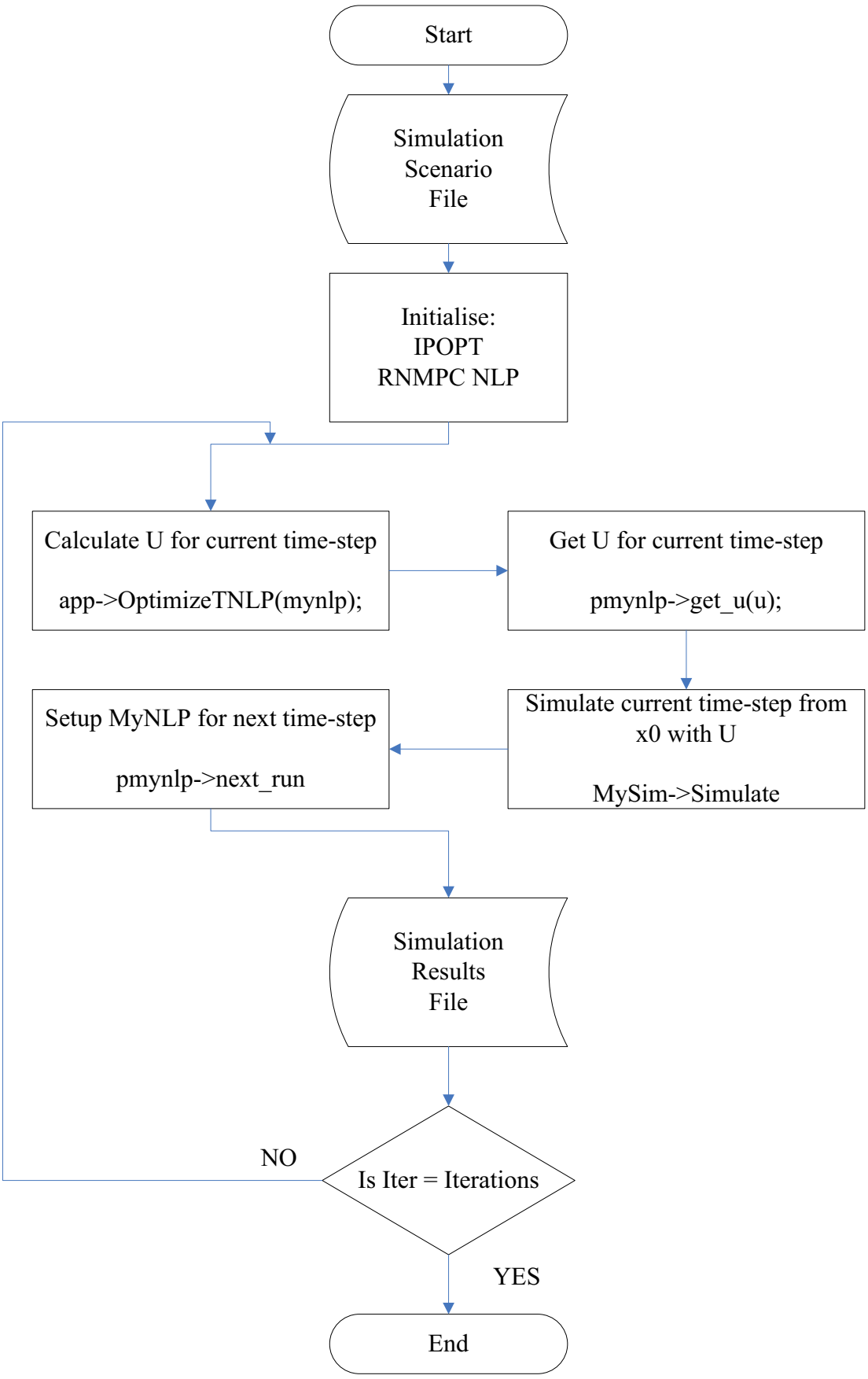


Figure A.4: Main execution loop.

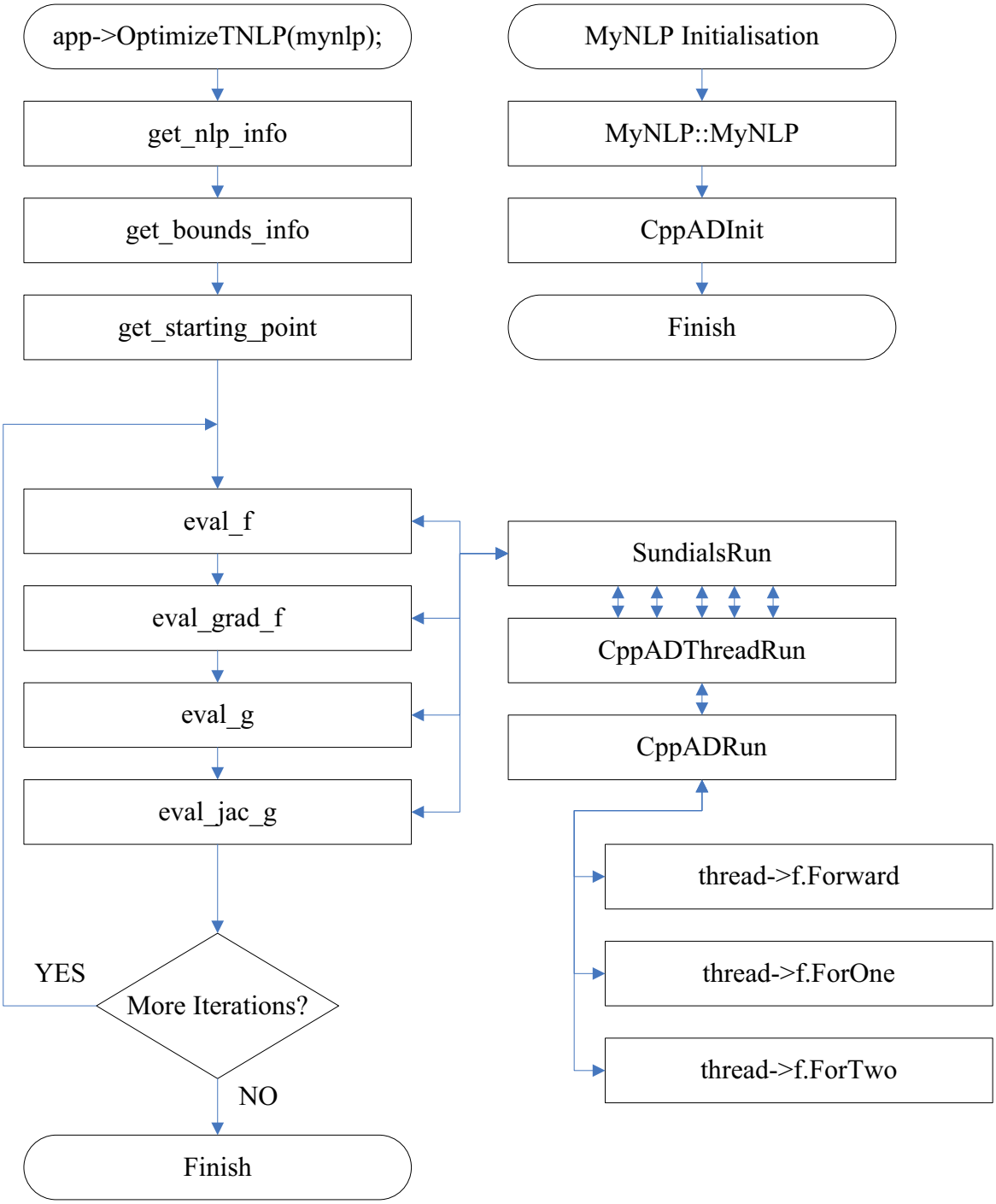


Figure A.5: Optimisation program flow.

- An instance of the IPOPT solver is created with a call to

```
SmartPtr<IpoptApplication> app = new IpoptApplication();
```

- IPOPT is instructed to use the quasi-Newton approximation of the Hessian and the IPOPT solver initiated with calls to

```
app->Options()->SetStringValue("hessian_approximation",
"limited-memory");
app->Initialize();
```

- The main simulation loop performs the following actions for the number of iterations specified in the simulation scenario file.
  - Reads the parameter vector, disturbances on the control moves and state setpoint for the current iteration from the simulation scenario file.
  - Calls

    ```
    status = app->OptimizeTNLP(mynlp);
    ```

    to perform a *cold-startup* optimisation or

    ```
    status = app->ReOptimizeTNLP(mynlp);
    ```

    to perform a *warm-startup* optimisation for the current time-step.
  - The control moves for the current time-step is extracted with a call to

    ```
    pmynlp->get_u(u);
    ```

The current time-step is simulated with a call to

```
mysim->simulate(x0, u, u_dist, p_dist, samples, xf, y);
```

with the initial state (`x0`), the calculated control moves (`u`), the disturbances on the control moves (`u_dist`), the disturbed parameter vector (`p_dist`), the number of sub-samples to calculate (`samples`) for more accurate simulation. The method returns the "real" final state (`xf`) of the system as well as the output of the system (`y`). The simulation basically just integrates the system dynamics with the provided parameters for one sampling interval.

  - The nonlinear programming problem is initialised for the next iteration with a call to

    ```
    pmynlp->next_run(x_setpoint, xf, u, u_setpoint, &obj, contraction);
    ```

with the state setpoint for the next time-step (`x_setpoint`), the initial state of the next time-step (`xf`) being the same as the final state of the current time-step, the previous control moves (`u`) for the next time-step and the steady-state values for the control moves (`u_setpoint`) for the next time-step. The method returns the objective function value (`obj`) of the current time-step as well as the margins applied to the constraints for robustness (`contraction`).

- The simulation results are saved at every iteration to the simulation results file. It is done to ensure that data are available, should the simulation not complete successfully. This will allow the simulation to be debugged, should it fail prematurely. The simulation data that is returned for each iteration is
  - the system state,
  - the control moves,
  - the system output,
  - the steady-state values of the control moves,
  - the setpoint of the system,
  - the objective function value,
  - the margins applied to the constraints for robustness,
  - the execution time of the iteration, and
  - the total execution time until the current iteration.
- The "real" final state of the system (`xf`) is assigned to the initial state (`x0`) of the system and the loop repeats until all the iterations are done.

The application returns statistics on each iteration to the terminal to help track the progress of the simulation as it executes. The current iteration, the total number of iterations, the execution time of the last iteration and the total execution time until the last iteration are displayed.

ADDENDUM B

AUXILIARY RESULTS

B.1 AUXILIARY SIMULATION RESULT

This section contains the graphs of mostly the NMPC controller that corresponds to the simulation scenarios of Section 5.3.

B.1.1 Constant setpoint following and disturbance rejection

The simulation scenario shown in Figure B.1a and Figure B.2a allows SLEV to vary freely with only upper and lower constraints enforced. Steering SLEV to setpoint compared to allowing SLEV to vary freely does not have a significant impact on the closed-loop performance under NMPC in terms of PSE setpoint tracking and the average circuit throughput (Table B.1). Allowing SLEV to vary within bounds allows the NMPC to change the density of the slurry inside the sump (assuming fully mixed conditions) and as a result allows the NMPC to control the feed density to the cyclone. Control of the feed density to the cyclone can increase the control envelope of the NMPC.

B.1.2 Reduced PSE setpoint to 75% and 70% < 75μm

Figure B.3b and Figure B.4b show that the NMPC tracks the reduced PSE setpoint of 75% < 75μm and LOAD setpoint of 45% volumetric filling well. The NMPC does not track PSE as closely as the RNMPC shown in Figure 5.16b and Figure 5.17b. The throughput shows large variations due primarily to the ore hardness and composition variations. Decreasing the setpoint for PSE to 75% increased the average throughput of the milling circuit to 74.3 from 72.6 tons per hour.

Figure B.3c and Figure B.4c show that the NMPC tracks the reduced PSE setpoint of 70% < 75μm and LOAD setpoint well. Decreasing the PSE setpoint to 70% resulted in an average throughput of 81.9 tons per hour, as seen in Table B.1.

The RNMPC of Section 5.3.3 and the NMPC results presented here show almost identical results, with the RNMPC tracking the PSE setpoint slightly better.

B.1.3 Regulate PSE, LOAD and Throughput

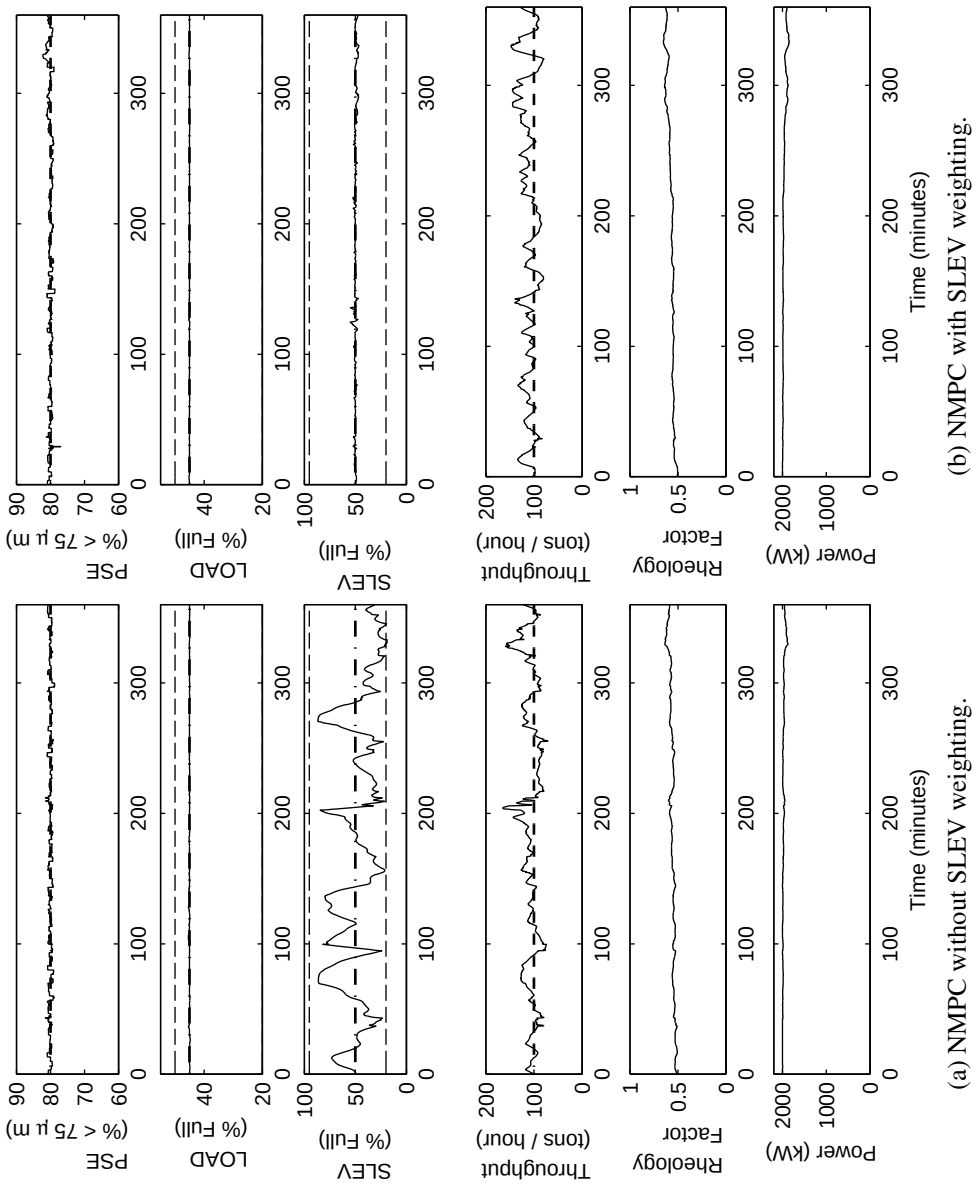
Figure B.5a and Figure B.6a show that adding the throughput to the objective function causes the NMPC to make a trade-off between following the PSE setpoint and throughput setpoint according to their respective weightings. Table B.1 shows that the PSE error increases significantly (from 0.30 to 3.31) while the average throughput decreases slightly (from 72.5 $^{\text{tons}}/_{\text{hour}}$ to 71.5 $^{\text{tons}}/_{\text{hour}}$), compared to the scenario where throughput is not included in the objective function (Figure 5.14b and Figure 5.15b). It was expected that adding THROUGHPUT to the objective function would increase the average throughput, but it resulted in a slight reduction in the average throughput, which was unexpected. Figure B.6 shows large variation in the manipulated variables that can be the cause of the poor overall performance.

Figure B.5b and Figure B.6b show that increasing the weighting on THROUGHPUT from 5 to 10 increases the PSE setpoint tracking error from 3.31 to 7.89 while decreasing the average throughput from 71.5 $^{\text{tons}}/_{\text{hour}}$ to 69.3 $^{\text{tons}}/_{\text{hour}}$. Figure B.6b shows that the large variation in the manipulated variables persists and this may explain the unexpected drop in average throughput.

Figure B.5c and Figure B.6c show that increasing the weighting on THROUGHPUT further from 10 to 20 increases the error in the PSE setpoint tracking from 7.89 to 13.01 (as expected) while increasing the average throughput from 69.3 $^{\text{tons}}/_{\text{hour}}$ to 70.8 $^{\text{tons}}/_{\text{hour}}$, as seen in Table B.1. Figure B.6c shows large variation in the manipulated variables, because the gain of the controller is too high. The gain of the controller is controlled by the weighting on the manipulated and controlled variables.

Figure B.7 and Figure B.8 show that the large variations in the MVs are reduced each time the weightings on the MVs are increased. The PSE setpoint tracking error is reduced while the average throughput is increased each time the weightings on the MVs are increased. The PSE setpoint tracking error is worse but the average throughput is higher compared to the case where THROUGHPUT is not included in the objective function (Figure 5.14b and Figure 5.15b), as seen in Table B.1.

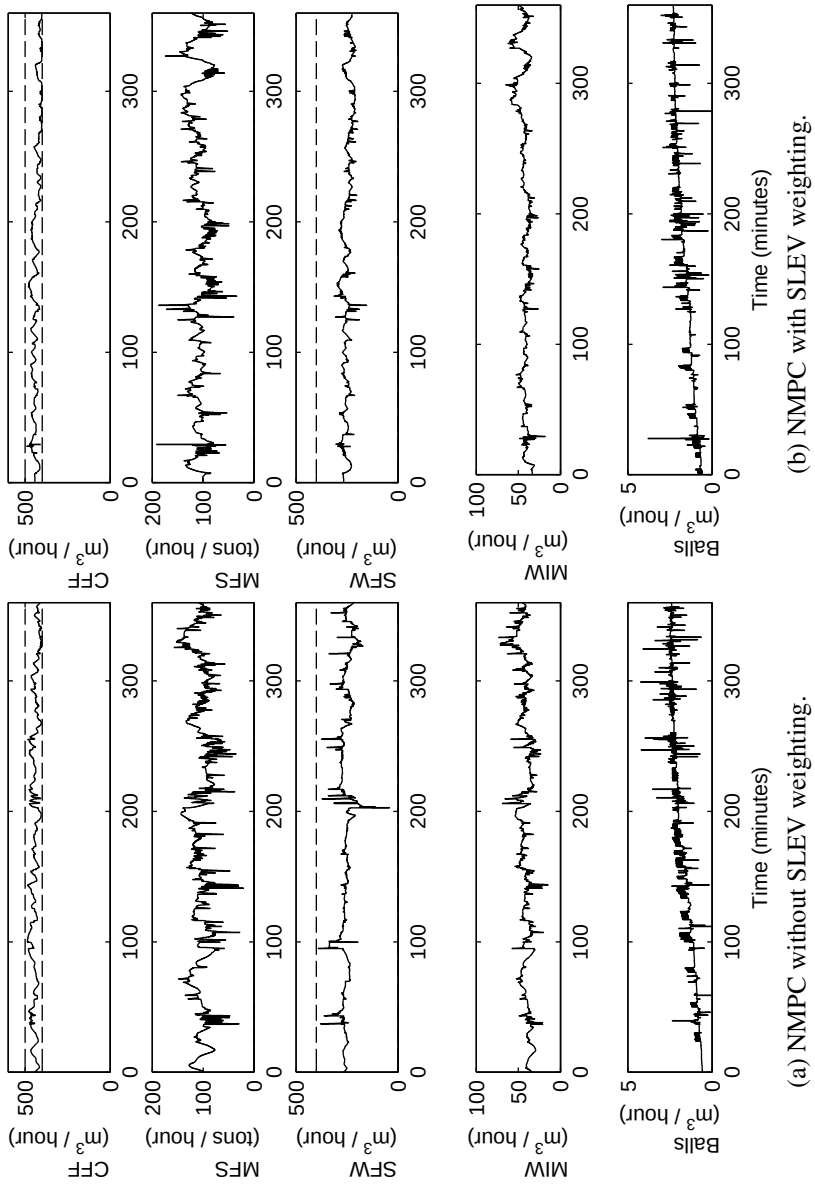
The results in this section show that the NMPC, like the RNMPC, cannot overcome the inherent trade-off between PSE and THROUGHPUT. The objective function allows the trade-off between PSE setpoint tracking and average throughput to be manipulated by changing the weighting of PSE and THROUGHPUT in the objective function.



(a) NMPC without SLEV weighting.

(b) NMPC with SLEV weighting.

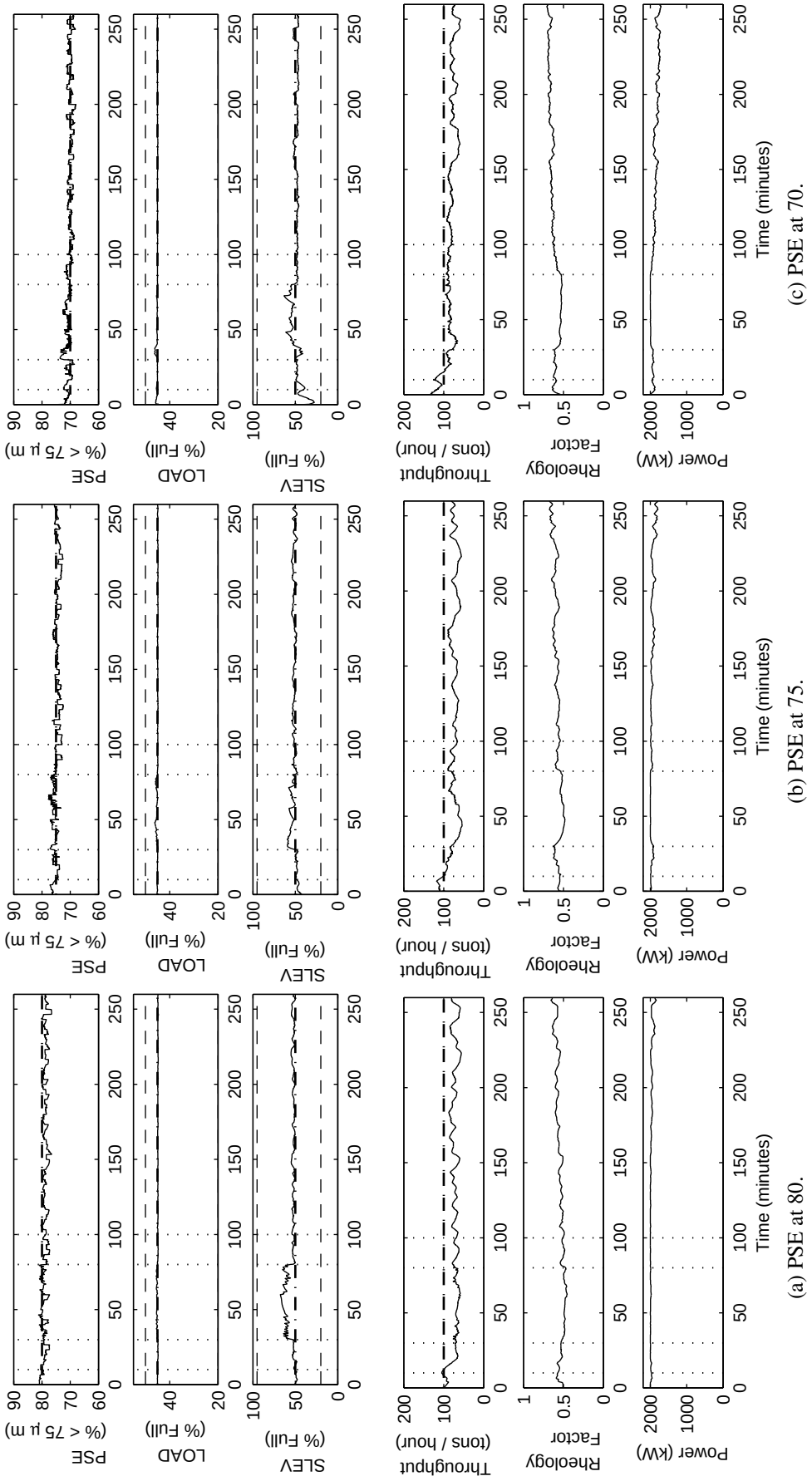
Figure B.1: CVs of NMPC
There is no step disturbance present in either the feed ore hardness or the fraction of rock in the feed ore. This represents the nominal scenario with only zero mean parameter variations present. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) NMPC without SLEV weighting.

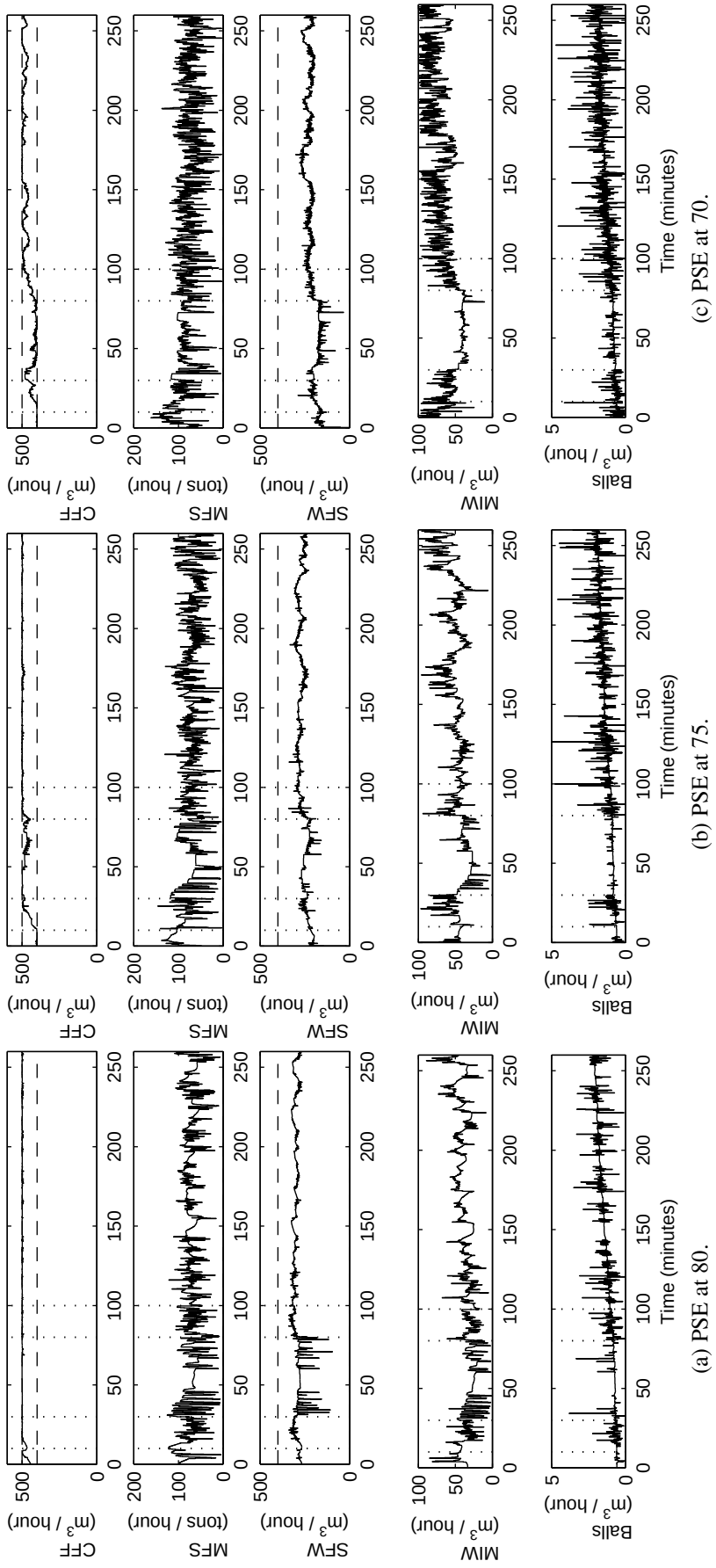
(b) NMPC with SLEV weighting.

Figure B.2: MVs of NMPC.
There is no step disturbance present in either the feed ore hardness or the fraction of rock in the feed ore. This represents the nominal scenario with only zero mean parameter variations present. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) PSE at 80. (b) PSE at 75. (c) PSE at 70.

Figure B.3: CVs of the NMPC.
The NMPC regulates PSE at a constant setpoint of 80% (a), 75% (b) and 70% (c) with LOAD at a constant setpoint of 45% and SLEV at a constant setpoint of 5 m^3. The PSE setpoint is reduced in order to increase the average throughput. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

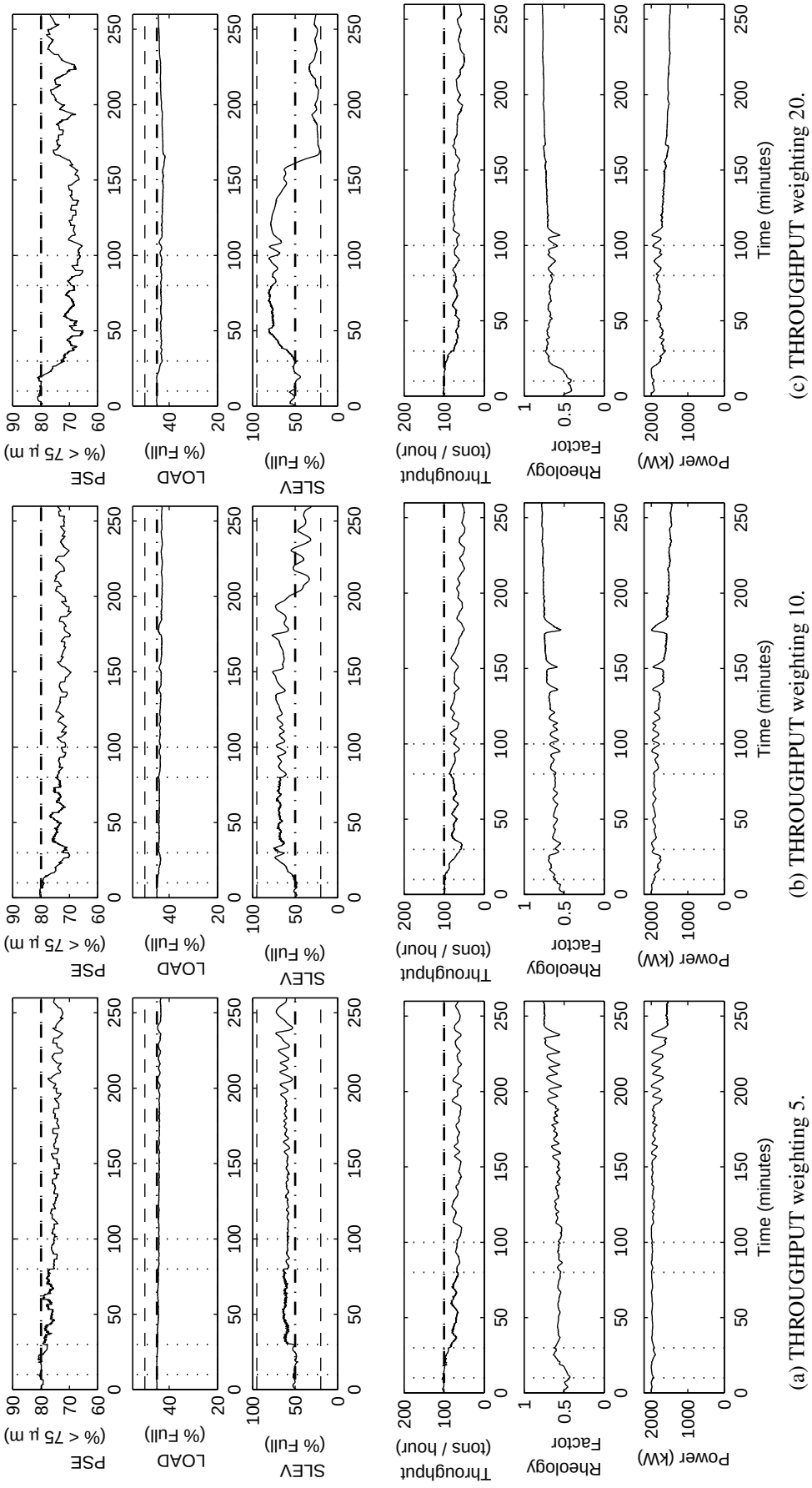


(a) PSE at 80.

(b) PSE at 75.

(c) PSE at 70.

Figure B.4: MVs of the NMPC.
The NMPC regulates PSE at a constant setpoint of 80% (a), 75% (b) and 70% (c) with LOAD at a constant setpoint of 45% and SLEV at a constant setpoint of 5 m^3. The PSE setpoint is reduced in order to increase the average throughput. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

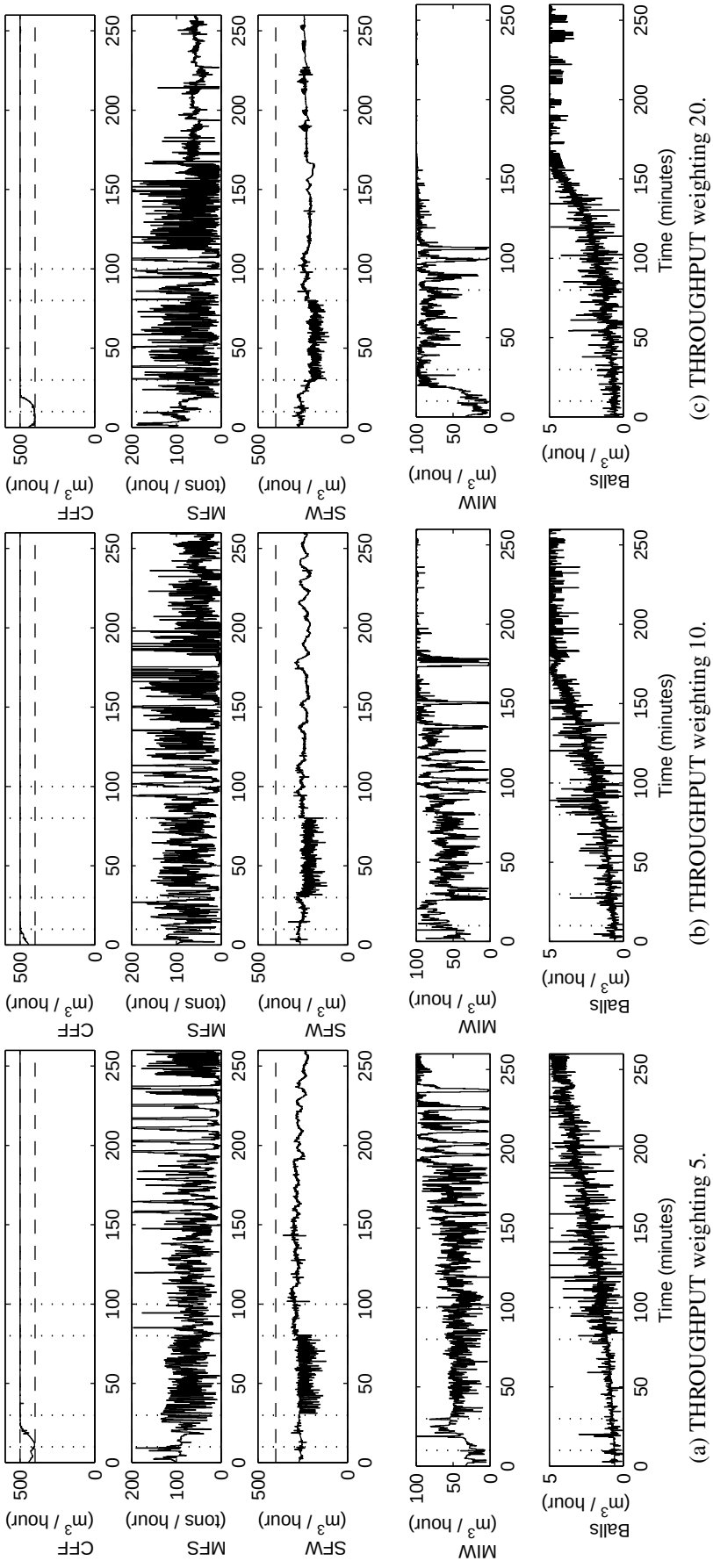


(a) THROUGHPUT weighting 5.

(b) THROUGHPUT weighting 10.

(c) THROUGHPUT weighting 20.

Figure B.5: CVs of the NMPC.
The NMPC regulates PSE and LOAD at constant setpoints and throughout at a constant setpoint of 100 tons/hour. The weighting of THROUGHPUT in the objective function is 5 (a), 10 (b) and 20 (c), while the weighting on PSE and LOAD is 100 respectively. The optimising property of the RNMPC is employed to try increasing throughput without affecting PSE. An increase in THROUGHPUT, however, causes a decrease in PSE. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

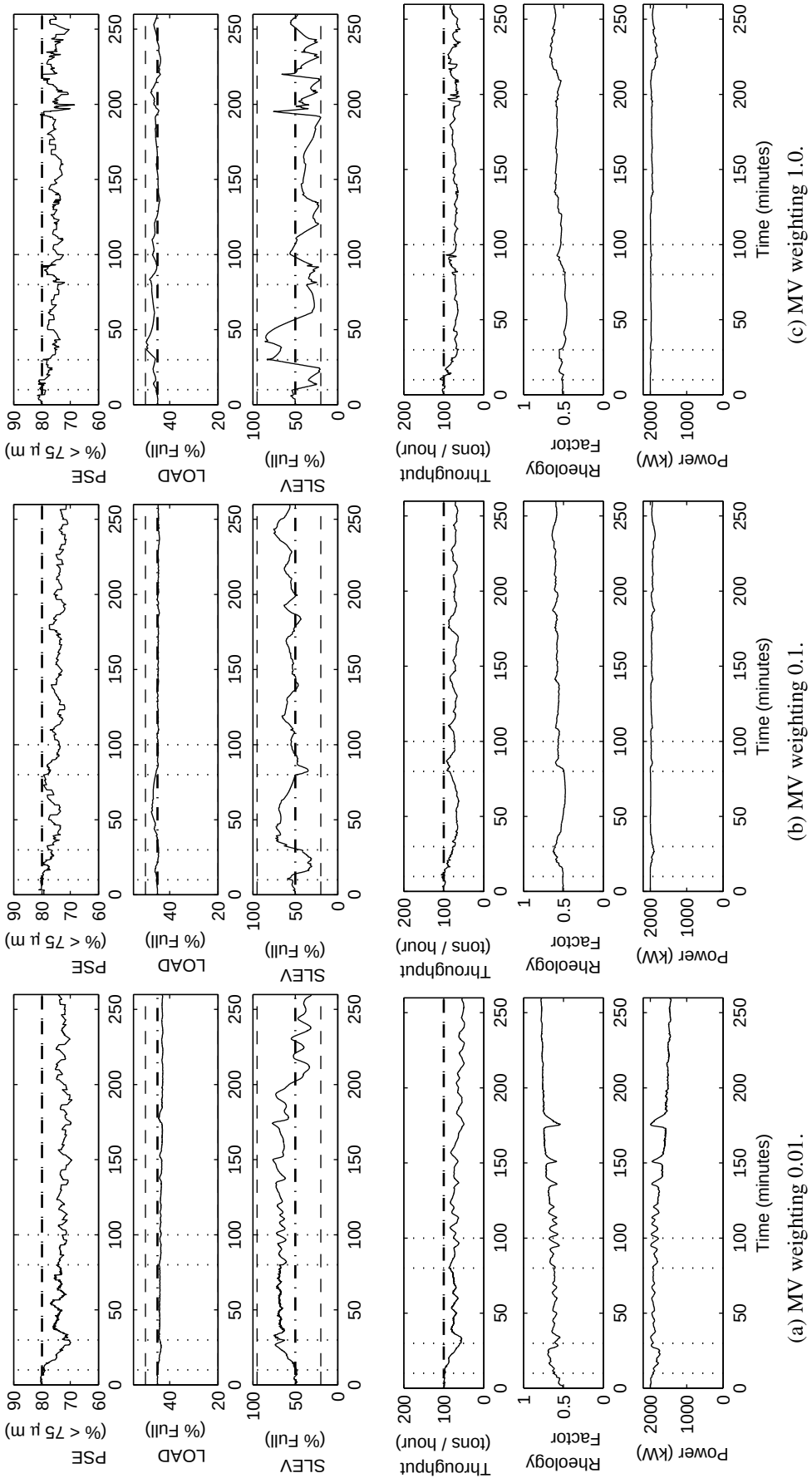


(a) THROUGHPUT weighting 5.

(b) THROUGHPUT weighting 10.

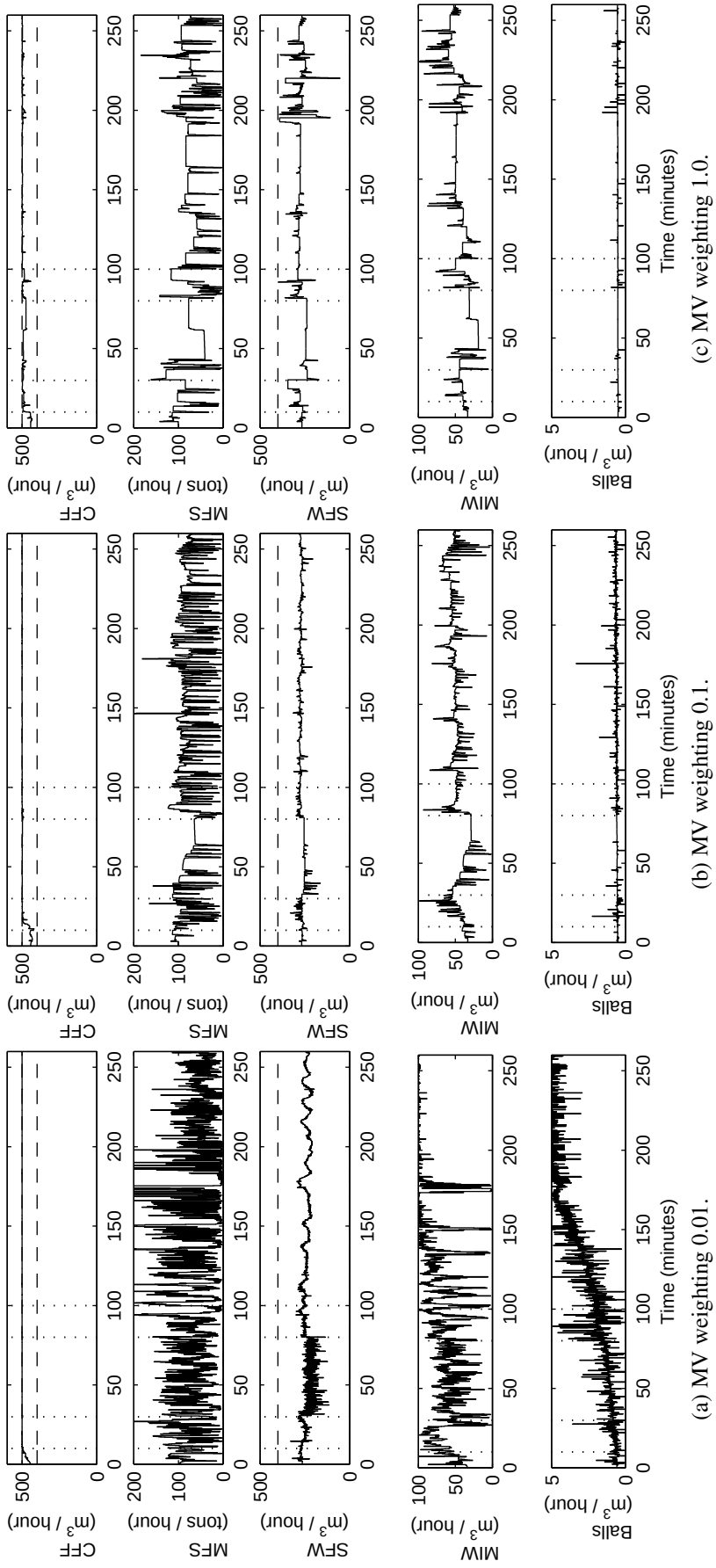
(c) THROUGHPUT weighting 20.

Figure B.6: MVs of the NMPC.
The NMPC regulates PSE and LOAD at constant setpoints and throughout at a constant setpoint of 100 tons/hour. The weighting of THROUGHPUT in the objective function is 5 (a), 10 (b) and 20 (c), while the weighting on PSE and LOAD is 100 respectively. Increasing the weighting on THROUGHPUT caused increasingly oscillatory behaviour in the MVs. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting 0.01. (b) MV weighting 0.1. (c) MV weighting 1.0.

Figure B.7: CVs of the NMPC.
The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to reduce the oscillation in the MVs. The increased weighting on the MVs results in better tracking of PSE and higher average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.8: MVs of the NMPC.
The weighting on the MVs is increased from 0.01 (b) to 0.1 (b) and 1.0 (c) in order to reduce the oscillation in the MVs. The increased weighting on the MVs results in better tracking of PSE and higher average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

B.2 ADDITIONAL SIMULATION SCENARIOS

B.2.1 Setpoints on PSE, LOAD, POWER and RHEOLOGY

In this scenario, setpoints for POWER and RHEOLOGY are added to the objective function of the RNMPC and NMPC. It is believed that operating the mill at maximum power causes most breakage to occur and therefore the highest throughput to be obtained. The breakage is also most effective at the optimum slurry rheology. This simulation scenario is more of academic interest, because the maximum power value is not known for varying process conditions and the optimal rheology is also not known online. For the simulation model, however, the optimum values are known and used in these simulations to determine if it yields better performance compared to the previous simulation scenarios.

The closed-loop response with setpoints for POWER and RHEOLOGY is shown for the RNMPC in Figure B.9 and Figure B.10 and for the NMPC in Figure B.11 and Figure B.12. The weightings on the MVs are increased to gauge the effect of a slower response on the closed-loop performance.

The RNMPC (Figure B.9a and Figure B.10a) and the NMPC (Figure B.11a and Figure B.12a) show very good closed-loop performance with PSE tracking being tighter and the average throughput being higher compared to the scenario where POWER and RHEOLOGY are not included in the objective function (RNMPC: Figure 5.14a & Figure 5.15a; NMPC: Figure 5.14b & Figure 5.15b).

Table B.1 shows that increasing the weighting on the MVs results in larger PSE and LOAD setpoint tracking errors for the RNMPC (Figure B.9b and Figure B.10c) and the NMPC (Figure B.11b and Figure B.12c), as would be expected, because the system responds more slowly to disturbances.

This scenario does improve on the scenarios where POWER and RHEOLOGY are not included in the objective function, because it reduces the variability of these two variables from their optimum values. Practically, the optimum values of POWER and RHEOLOGY would somehow need to be estimated, as well as the actual value for RHEOLOGY. Fixing these values also reduces the freedom of the controller to follow the more important variables, such as PSE.

B.2.2 Increase the weighting on the MVs for objective function with only PSE and LOAD setpoints

In this scenario, the effect of increasing the weighting on the MVs is investigated where only PSE and LOAD are included in the objective function. In Section 5.3.5 and Section B.1.3, increasing the weighting on the MVs improved the closed-loop performance. At first this seems counter-intuitive, but there were large variations and even oscillations occurring on

the MVs. Increasing the weighting reduced the variations and consequently improved the closed-loop performance of the process.

The closed-loop performance of the process under RNMPC (Figure B.13 and Figure B.14) and NMPC (Figure B.15 and Figure B.16) shows increased PSE and LOAD setpoint tracking errors as the weightings on the MVs are increased and the MVs show smaller variations. This is expected, because the controllers are slower to respond to disturbances and therefore increase the variation of the process variables around their setpoints.

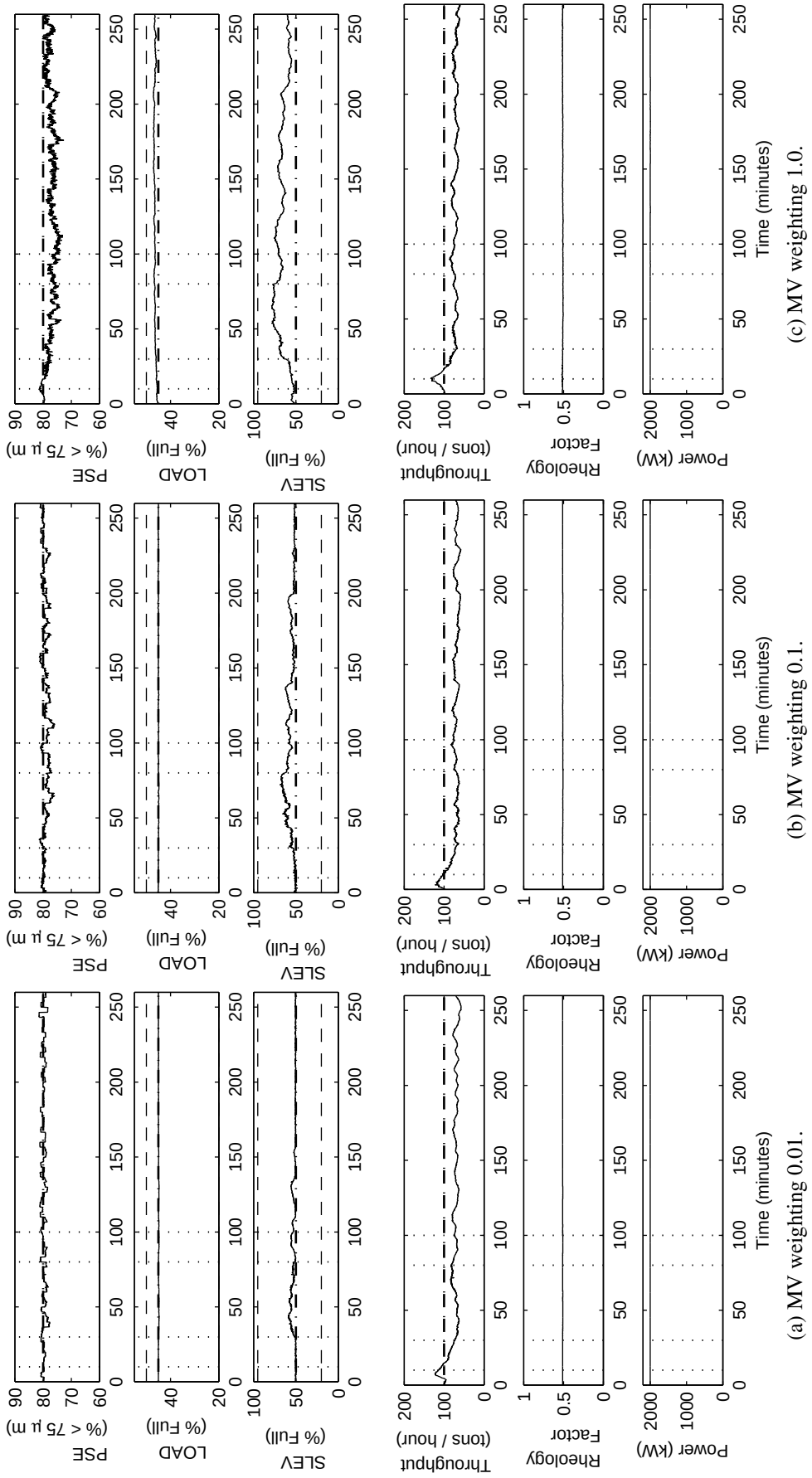
In this scenario where THROUGHPUT was not included in the objective function, increasing the weightings on the MVs reduced the closed-loop performance of the process with regard to PSE and LOAD setpoint tracking.

B.2.3 Increase weightings on the MVs for objective functions with PSE, LOAD and THROUGHPUT setpoints

This is similar to Section 5.3.5 and Section B.1.3 to see if the closed-loop performance for the RNMPC (Figure 5.24c and Figure 5.25c) and NMPC (Figure B.5c and Figure B.6c) with a weighting of 20 for THROUGHPUT in the objective function improves if the weightings on the MVs are increased.

The RNMPC (Figure B.17 and Figure B.18) and the NMPC (Figure B.19 and Figure B.20) show improved PSE setpoint tracking and an increase in the average throughput when the weightings on the MVs are increased from 0.01 to 1.0.

Including THROUGHPUT in the objective function changes the relative weighting between the CVs and the MVs, causing the overall gain of the controller to increase. The increased gain causes the controllers to make large control moves, which ultimately leads to sub-optimal closed-loop performance. Therefore, increasing the weightings on the MVs when THROUGHPUT is included in the objective function improved the overall closed-loop performance of the process.

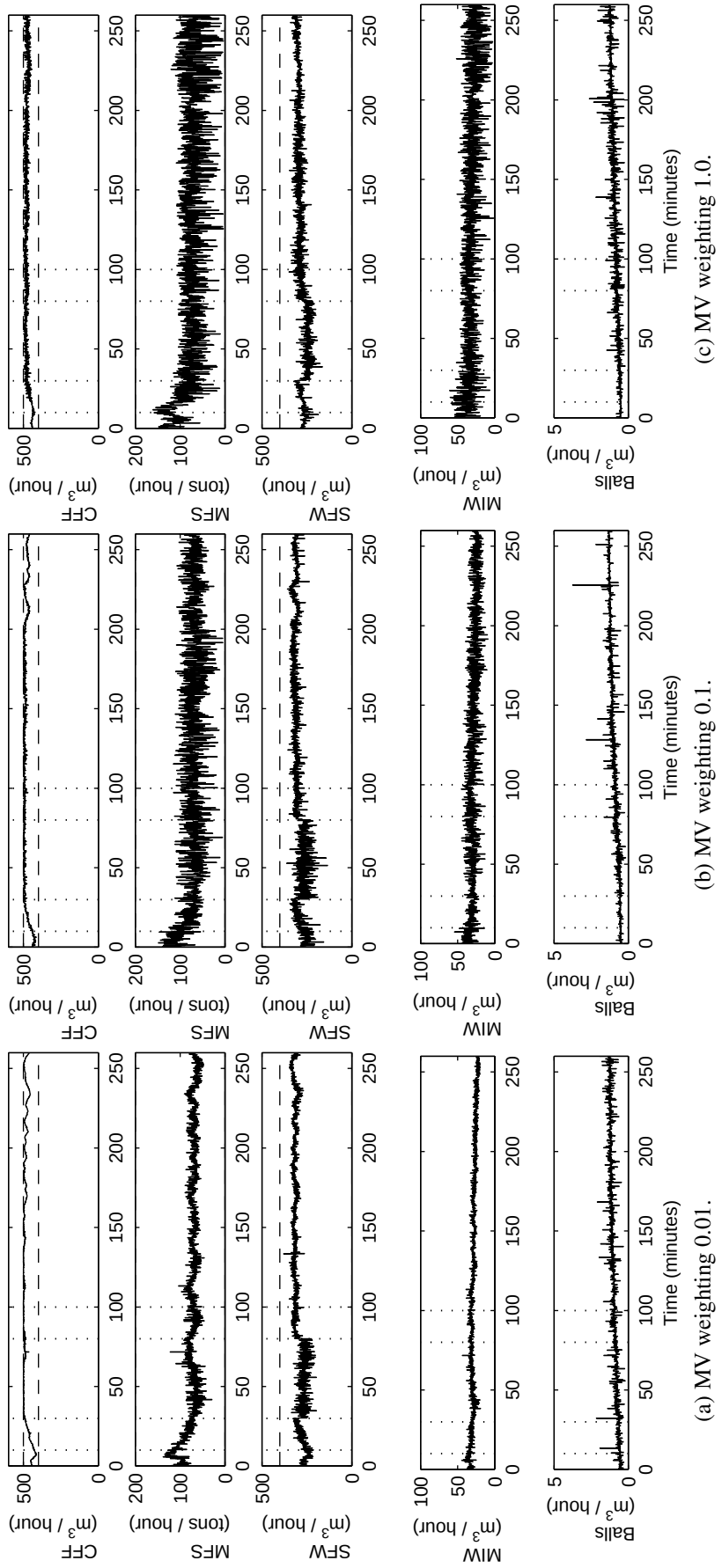


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.9: CVs of the RNMPC.
This scenario includes setpoints for the POWER and RHEOLOGY at their optimum values in the objective function. The added setpoints improve PSE setpoint tracking and the average THROUGHPUT. The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movements of the MVs on the setpoint tracking performance of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and LOAD setpoints while increasing the average THROUGHPUT slightly. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

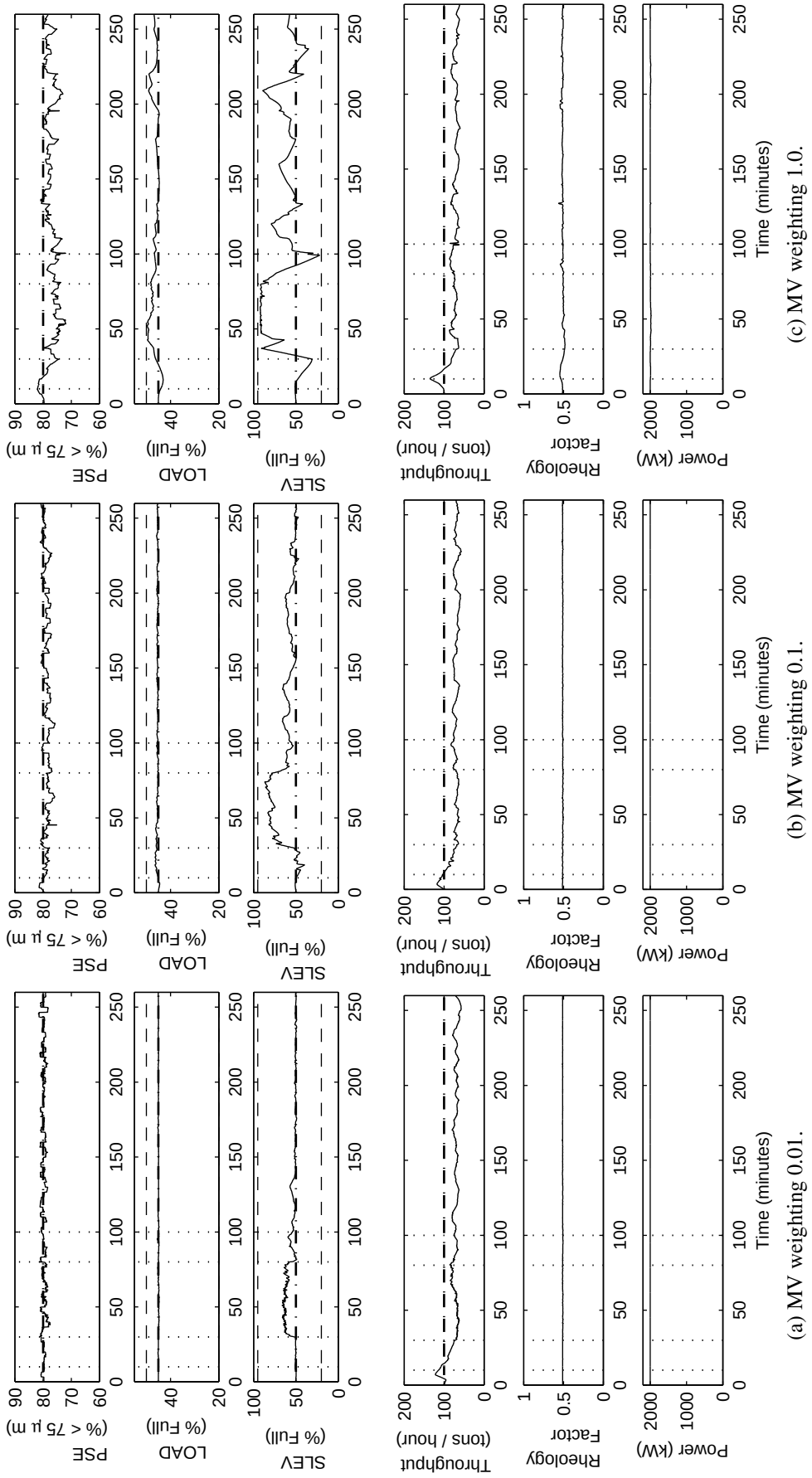


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.10: MVs of the RNMPC.
This scenario includes setpoints for the POWER and RHEOLOGY at their optimum values in the objective function. The added setpoints improve PSE setpoint tracking and the average THROUGHPUT. The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movements of the MVs on the setpoint tracking performance of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and LOAD setpoints while increasing the average THROUGHPUT slightly. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

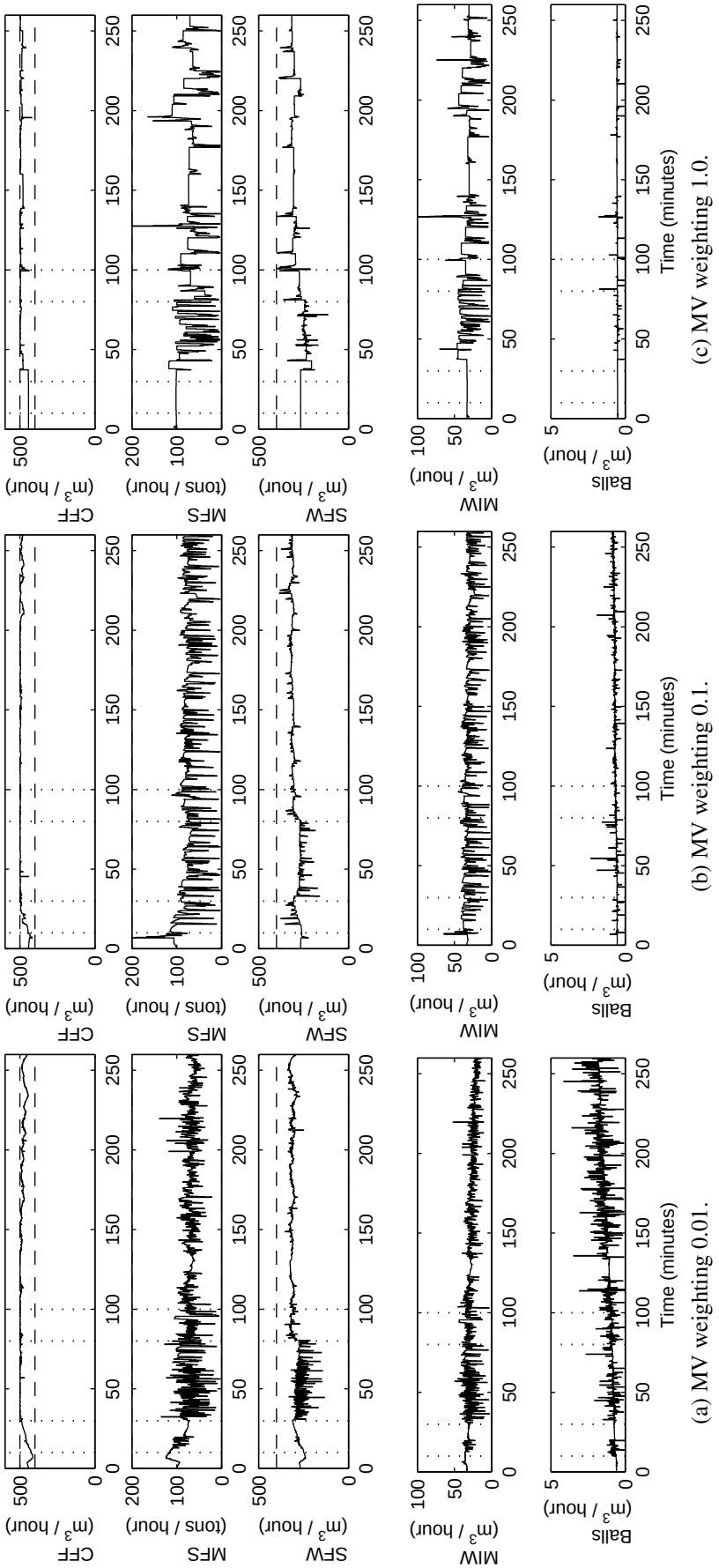


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.11: CVs of the NMPC.
This scenario includes setpoints for the POWER and RHEOLOGY at their optimum values in the objective function. The added setpoints improve PSE setpoint tracking and the average THROUGHPUT. The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movements of the MVs on the setpoint tracking performance of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and LOAD setpoints while increasing the average THROUGHPUT slightly. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

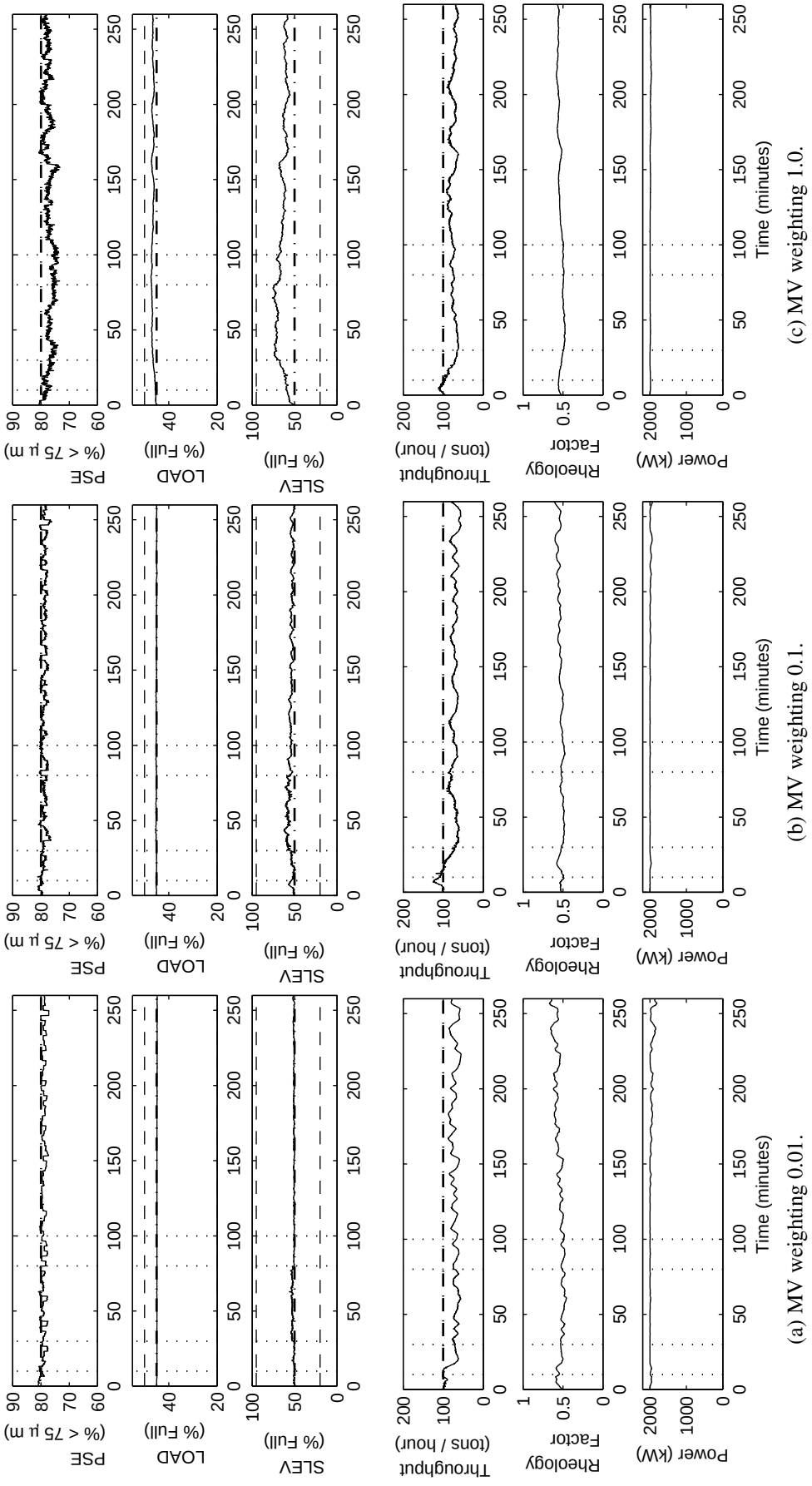


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.12: MVs of the NMPC.
This scenario includes setpoints for the POWER and RHEOLOGY at their optimum values in the objective function. The added setpoints improve PSE setpoint tracking and the average THROUGHPUT. The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movements of the MVs on the setpoint tracking performance of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and LOAD setpoints while increasing the average THROUGHPUT slightly. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

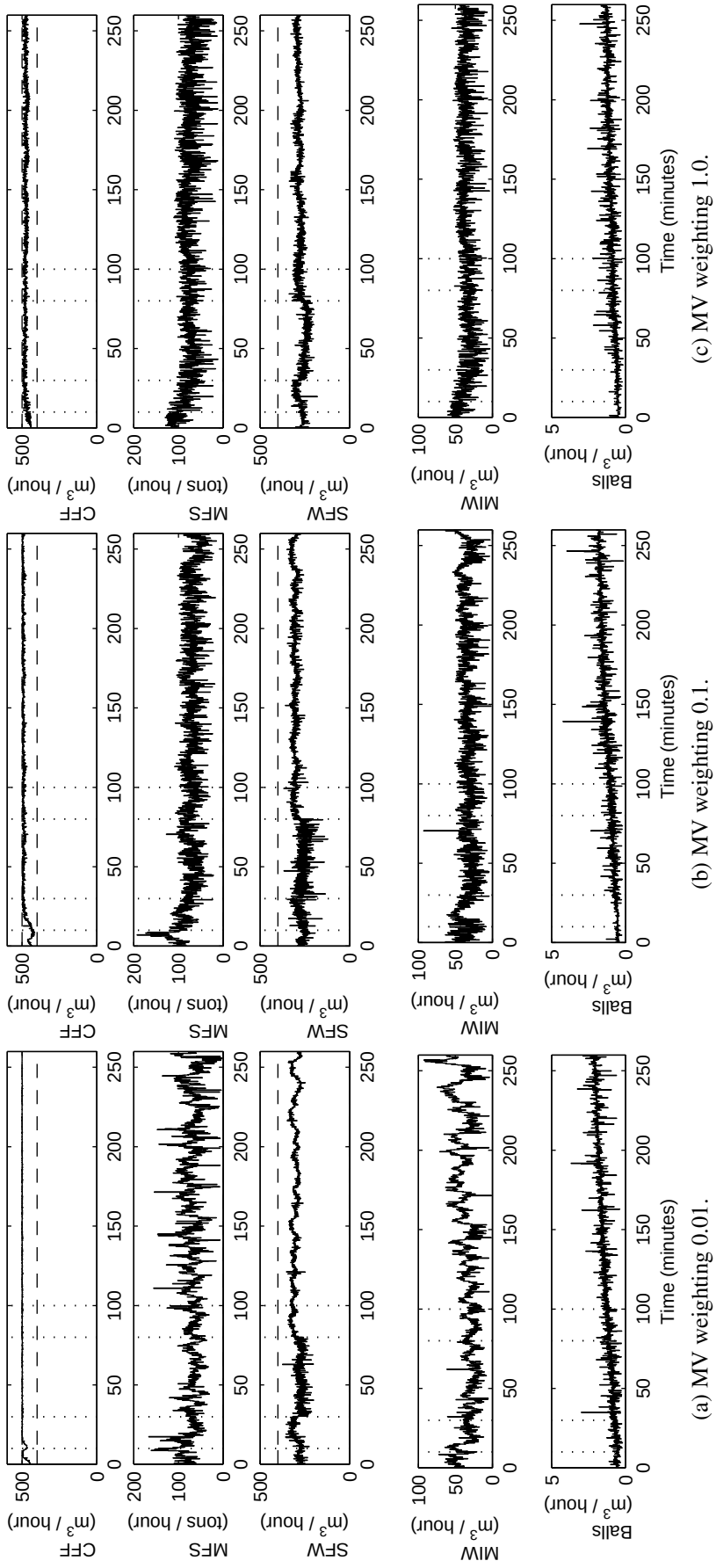


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.13: CVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movement of the MVs on the setpoint tracking of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and the LOAD setpoints while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

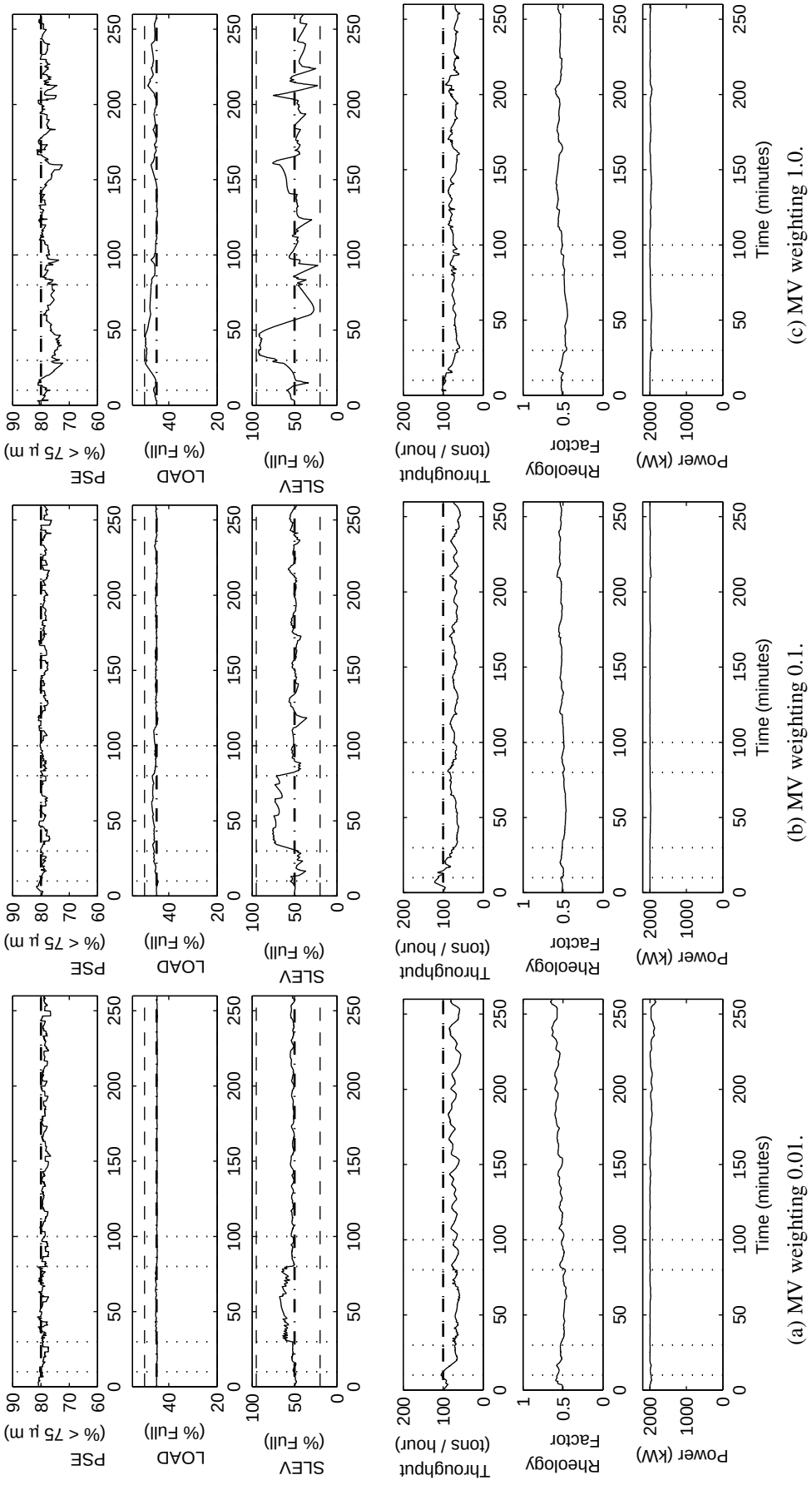


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.14: MVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movement of the MVs on the setpoint tracking of the CVs. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

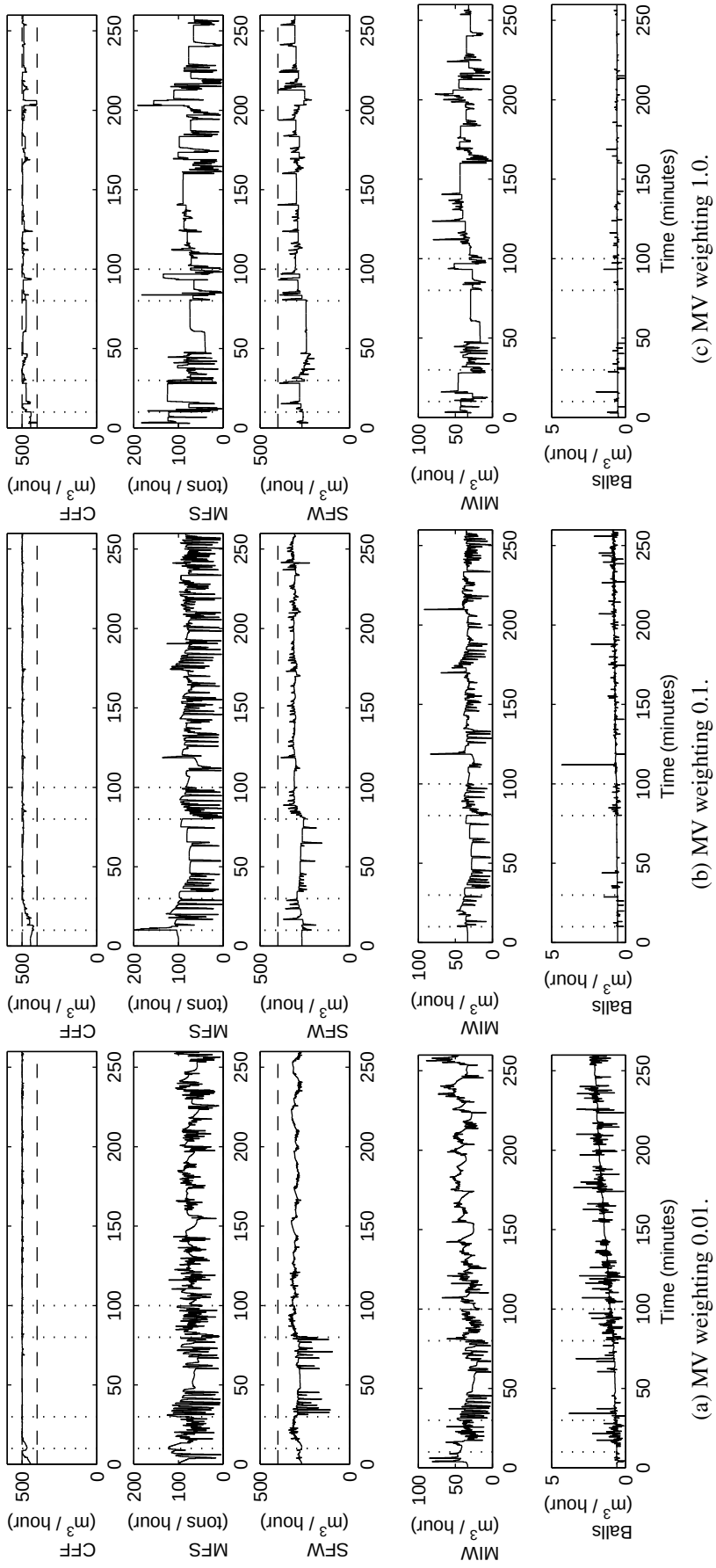


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

Figure B.15: CVs of the NMPC.
The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movement of the MVs on the setpoint tracking of the CVs. The increased weighting on the MVs results in poorer tracking of the PSE and the LOAD setpoints while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

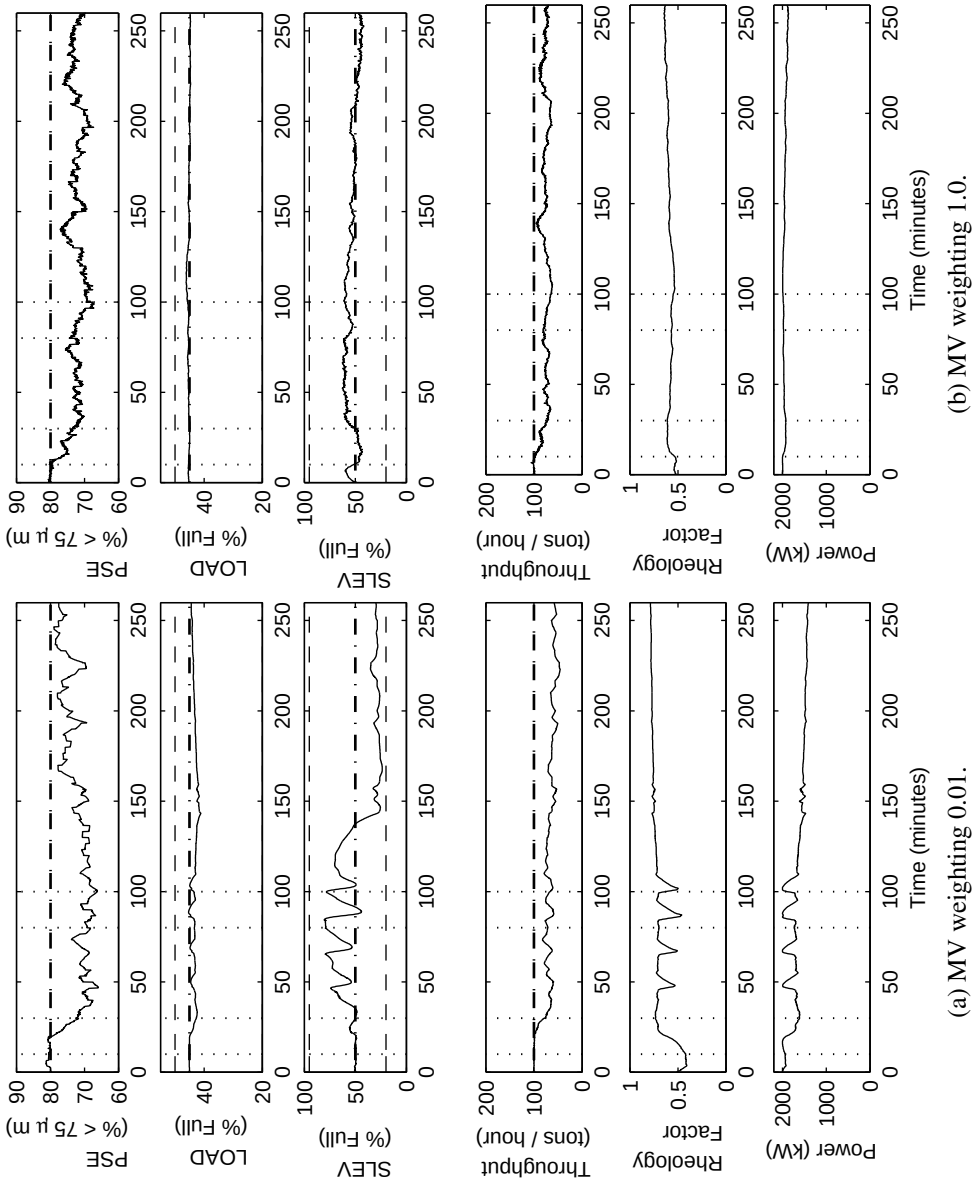


(a) MV weighting 0.01.

(b) MV weighting 0.1.

(c) MV weighting 1.0.

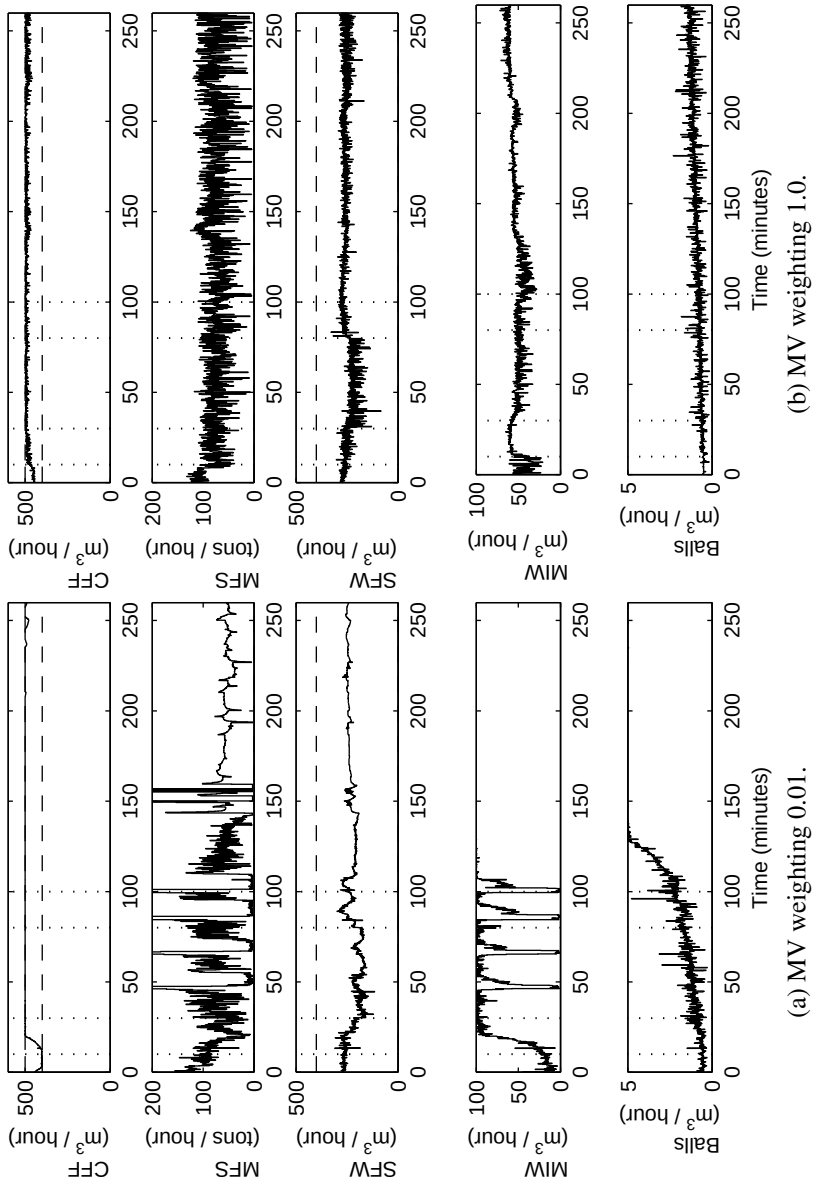
Figure B.16: MVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (a) to 0.1 (b) and 1.0 (c) in order to investigate the effect of smaller movement of the MVs on the setpoint tracking of the CVs. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting 0.01.

(b) MV weighting 1.0.

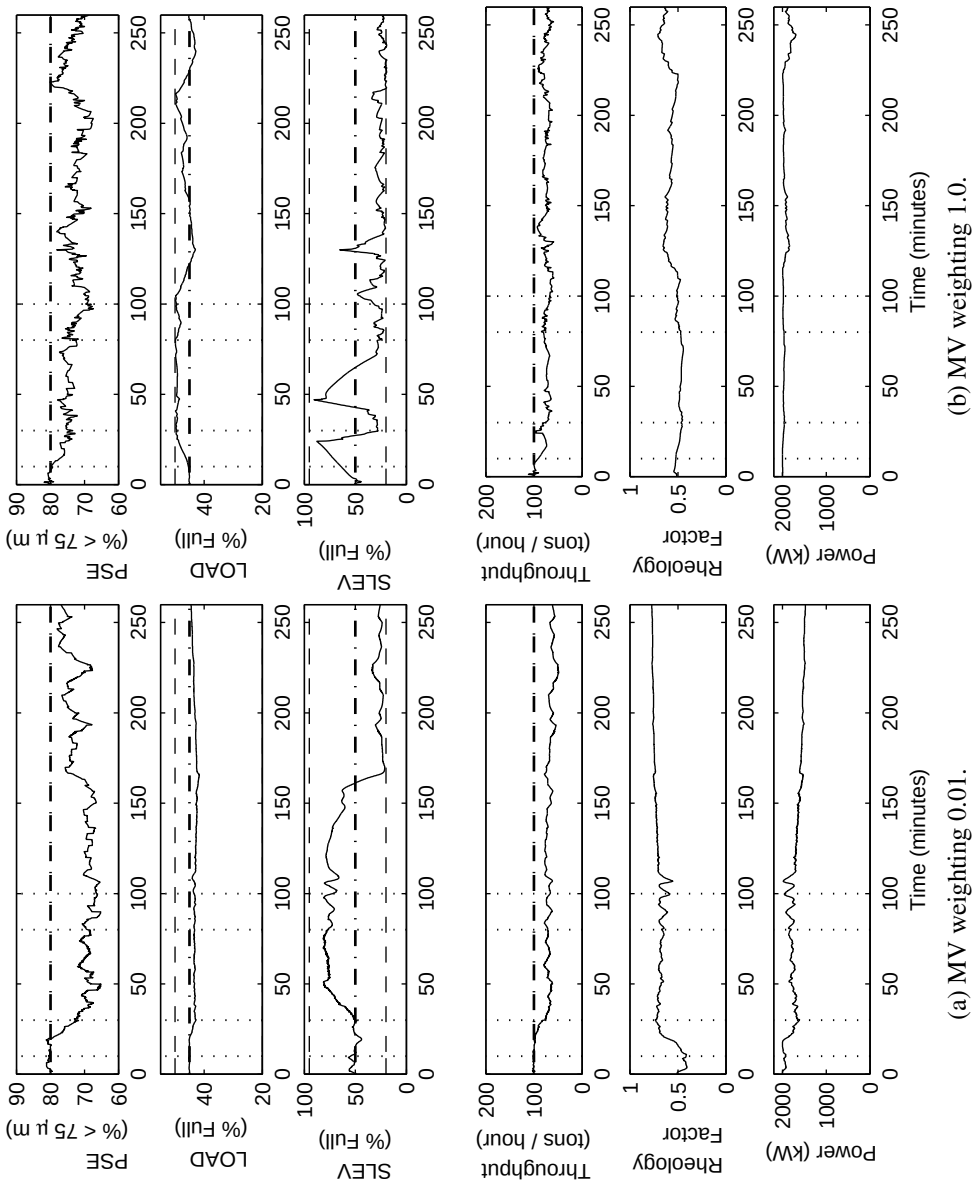
Figure B.17: CVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (a) and 1.0 (b) in order to reduce the oscillations of the MVs. The increased weighting on the MVs results in better tracking of the PSE setpoint while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting 0.01.

(b) MV weighting 1.0.

Figure B.18: MVs of the RNMPC.
The weighting on the MVs is increased from 0.01 (a) and 1.0 (b) in order to reduce the oscillations of the MVs. The increased weighting on the MVs results in better tracking of the PSE setpoint while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.



(a) MV weighting 0.01.

(b) MV weighting 1.0.

Figure B.19: CVs of the NMPC.
The weighting on the MVs is increased from 0.01 (a) and 1.0 (b) in order to reduce the oscillations of the MVs. The increased weighting on the MVs results in better tracking of the PSE setpoint while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

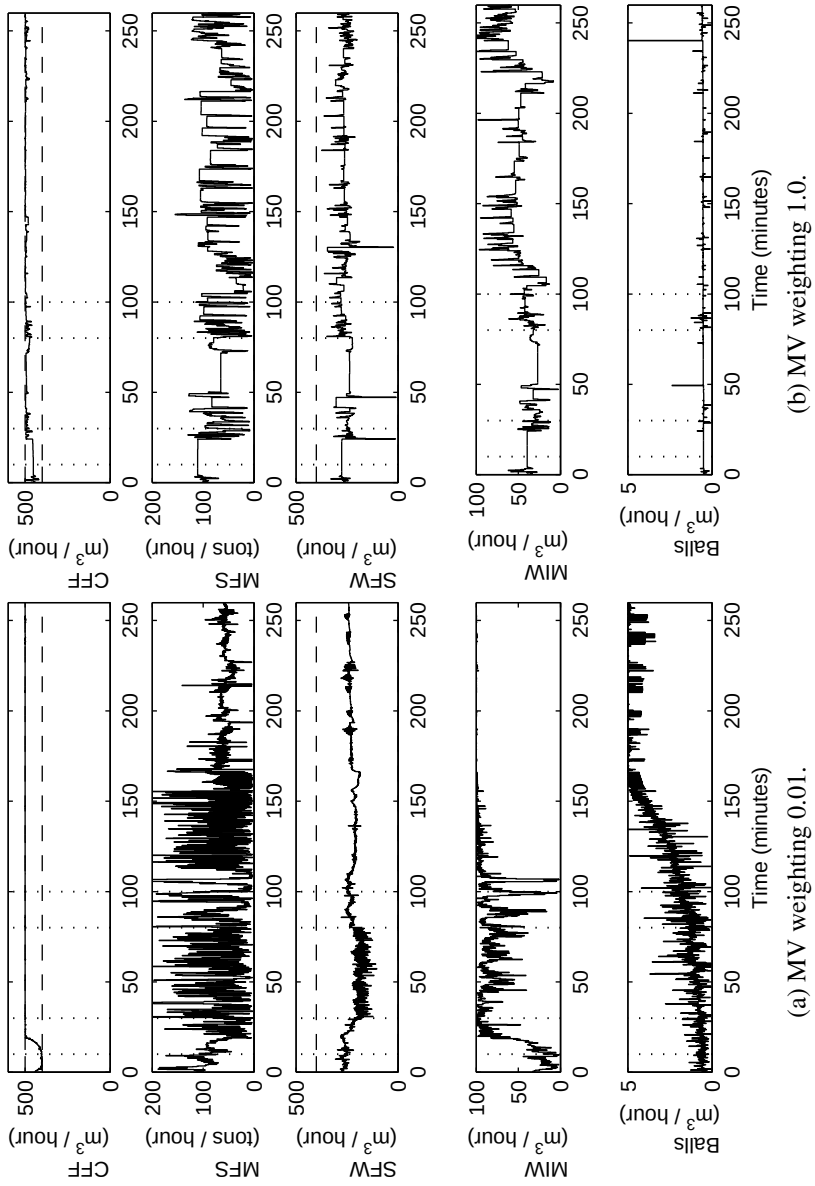


Figure B.20: MVs of the NMPC.
The weighting on the MVs is increased from 0.01 (a) and 1.0 (b) in order to reduce the oscillations of the MVs. The increased weighting on the MVs results in better tracking of the PSE setpoint while increasing the average THROUGHPUT. The dashed lines indicate the constraints on the variable and the vertical dotted lines indicate the start of the disturbance events. The dash-dot line indicates the setpoint.

B.3 SIMULATION SUMMARY

A summary of the simulation results are given in Table B.1 that details the tracking performance of the controller with regard to the particle size (PSE), mill load level (LOAD) and throughput. The simulation scenario is outlined in terms of the controlled variable weighting and setpoint values, as well as the step disturbances. The relevant changes in each simulation scenario are highlighted in bold. The dashes in the table represent values that are either zero or not applicable. The performance metrics are described in Section 5.2.

The headings of Table B.1 are defined as

PSE	Particle Size Estimate. [$\% < 75\mu$m]
LOAD	The volumetric filling of the mill. [%]
Throughput	The amount of solids discharged at the cyclone overflow. [$^{\text{tons}}/_{\text{hour}}$]
SLEV	Sump level. [m^3]
Power	The electrical power draw of the mill motor. [kW]
Rheology	An indication of the fluidity of the slurry inside the mill. [fraction]
U	The manipulated variables.
Disturbances	Describes the step disturbances in ore hardness, fraction of rock in the feed ore and SFW.
Time	Describes the average and maximum iteration time of the simulations [seconds].

The subheadings of Table B.1 are defined as

Δ	The sum of the squares of the error from the setpoint.
S	The setpoint of the variable.
W	The weight of the variable in the objective function.
A	The average value of the variable over the simulation duration.
RH	The increase in the hardness of the feed ore. [%]
AR	The increase in the fraction of rock in the feed ore. [%]
SFW	The increase in SFW. [$^{m^3}/_{\text{hour}}$]
T	The time when the disturbance is introduced.
M	The maximum value of the variable over the simulation duration.

The controllers are identified next to the figure numbers in Table B.1 by

R	Robust Nonlinear Model Predictive Controller.
N	Nonlinear Model Predictive Controller.
P	Proportional-Integral-Differential Controller.

The simulations show that RNMPC and NMPC are capable of tighter control of PSE, especially when constraints are active, because the multivariable controllers can leverage the multivariable nature of the milling circuit to increase the control envelope. The RNMPC and NMPC controllers are, however, not capable of improving throughput while maintaining PSE at the desired setpoint. The PI controllers performed very well, because they were tuned very aggressively. In certain milling circuits there are large time delays that result in less aggressive tuning of the PI controllers and degraded performance. MPC controllers were found in practice to perform well over longer periods compared to PI controllers (Chen *et al.*, 2007*b*, Ramasamy *et al.*, 2005).

Table B.1: Auxiliary simulation summary.

			PSE			LOAD			Throughput			SLEV		Power		Rheology		U	Disturbances					Time	
Simulation			Δ	S	W	Δ	S	W	A	S	W	S	W	S	W	S	W	W	RH	T	AR	T	S	A	M
Figure B.1	a	N	0.05	80	100	1.80E-05	45	100	106.31	99.9	0	5	**0**	2000	0	0.51	0	0.01	—	—	—	—	—	0.1	2.25
Figure B.2	b	N	0.07	80	100	4.80E-05	45	100	110.27	99.9	0	5	**1**	2000	0	0.51	0	0.01	—	—	—	—	—	0.1	1.6
Figure B.3	a	N	0.30	**80**	100	5.45E-03	45	100	72.54	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.2	1.77
Figure B.4	b	N	0.19	**75**	100	8.36E-03	45	100	74.33	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.23
	c	N	0.17	**70**	100	6.73E-03	45	100	81.87	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.08
Figure B.5	a	N	3.31	80	100	1.06E-01	45	100	71.45	99.9	**5**	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.04
Figure B.6	b	N	7.89	80	100	3.43E-01	45	100	69.25	99.9	**10**	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	1.77
	c	N	13.01	80	100	5.00E-01	45	100	70.83	99.9	**20**	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	1.38
Figure B.7	a	N	7.89	80	100	3.43E-01	45	100	69.25	99.9	**10**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	0.3	1.77
Figure B.8	b	N	4.71	80	100	5.86E-02	45	100	75.56	99.9	**10**	5	1	2000	0	0.51	0	**0.1**	50	10	50	100	50	30-80	0.2	2.3
	c	N	4.79	80	100	7.87E-01	45	100	75	99.9	**10**	5	1	2000	0	0.51	0	**1**	50	10	50	100	50	30-80	0.2	6.43
Figure B.9	a	R	0.1	80	100	1.62E-04	45	100	73.04	99.9	**0**	5	1	2000	**50**	0.51	**50**	**0.01**	50	10	50	100	50	30-80	36.2	665
Figure B.10	b	R	0.27	80	100	1.05E-03	45	100	71.43	99.9	**0**	5	1	2000	**50**	0.51	**50**	**0.1**	50	10	50	100	50	30-80	61.3	1168
	c	R	1.63	80	100	9.98E-03	45	100	74.81	99.9	**0**	5	1	2000	**50**	0.51	**50**	**1.0**	50	10	50	100	50	30-80	34.4	204
Figure B.11	a	N	0.1	80	100	2.49E-04	45	100	73.03	99.9	**0**	5	1	2000	**50**	0.51	**50**	**0.01**	50	10	50	100	50	30-80	0.2	2.07
Figure B.12	b	N	0.42	80	100	2.04E-03	45	100	71.93	99.9	**0**	5	1	2000	**50**	0.51	**50**	**0.1**	50	10	50	100	50	30-80	0.2	2.13
	c	N	1.81	80	100	9.88E-03	45	100	74.7	99.9	**0**	5	1	2000	**50**	0.51	**50**	**1.0**	50	10	50	100	50	30-80	0.2	2.99
Figure B.13	a	R	0.23	80	100	2.03E-03	45	100	72.22	99.9	**1**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	34.9	1014
Figure B.14	b	R	0.29	80	100	2.75E-03	45	100	73.82	99.9	**1**	5	1	2000	0	0.51	0	**0.1**	50	10	50	100	50	30-80	50.0	1295
	c	R	1.53	80	100	1.12E-02	45	100	76.07	99.9	**1**	5	1	2000	0	0.51	0	**1.0**	50	10	50	100	50	30-80	25.8	348
Figure B.15	a	N	0.30	80	100	5.45E-03	45	100	72.54	99.9	**1**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	0.2	1.77
Figure B.16	b	N	0.28	80	100	9.11E-02	45	100	73.79	99.9	**1**	5	1	2000	0	0.51	0	**0.1**	50	10	50	100	50	30-80	0.2	2.41
	c	N	1.31	80	100	6.07E-01	45	100	75.1	99.9	**1**	5	1	2000	0	0.51	0	**1**	50	10	50	100	50	30-80	0.2	3.75
Figure B.17	a	R	10.01	80	100	6.85E-02	45	100	68.09	99.9	**20**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	13.3	682
Figure B.18	b	R	9.19	80	100	4.38E-02	45	100	76.83	99.9	**20**	5	1	2000	0	0.51	0	**1.0**	50	10	50	100	50	30-80	44.7	1171
Figure B.19	a	N	13.01	80	100	5.00E-01	45	100	70.83	99.9	**20**	5	1	2000	0	0.51	0	**0.01**	50	10	50	100	50	30-80	0.3	1.38
Figure B.20	b	N	7.52	80	100	1.32E+00	45	100	75.73	99.9	**20**	5	1	2000	0	0.51	0	**1**	50	10	50	100	50	30-80	0.2	3.77

ADDENDUM C

PID TUNING WITH INTERACTIONS

In this addendum, the problem of tuning decentralised PID controllers while taking interaction of the MIMO system into account, is studied. There are two main approaches to handling interactions, the first is to design the decentralised controllers by taking interaction into account. Three main approaches exist to handle interaction in decentralised PID design: (1) by detuning the controllers to account for interaction, (2) using the critical gains of the system to tune the controllers and (3) using the whole transfer function to explicitly take interaction into account (Vázquez and Morilla, 2002). The second approach is to do a fully centralised controller design (Morilla *et al.*, 2008). A hybrid approach is to first design a decoupling network to minimise interaction between the loops and then apply decentralised methods to the decoupled plant (Vázquez and Morilla, 2002).

C.1 INTRODUCTION

Desbiens *et al.* (1996) presents a frequency-domain method to design decentralised PID controllers for a two-input-two-output (TITO) system while explicitly taking interaction into account. Pomerleau *et al.* (2000) applied the method by Desbiens *et al.* (1996) to a milling circuit to design the decentralised PID controllers and will be the method used in this addendum. The method by Desbiens *et al.* (1996) falls under the third approach of decentralised PID controller design methods, as described by Vázquez and Morilla (2002).

The method by Desbiens *et al.* (1996) aims to design decentralised controllers $G_{C1}(s)$ and $G_{C2}(s)$ for the 2×2 system

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \tag{C.1}$$

where the decentralised controllers have the following structure

$$G_C(s) = \begin{bmatrix} G_{C1}(s) & 0 \\ 0 & G_{C2}(s) \end{bmatrix} \tag{C.2}$$

and the closed-loop system that forms by combining the decentralised controllers $G_C(s)$ in a feedback loop with the 2×2 system $G(s)$ is given by

$$T(s) = \frac{G(s)G_C(s)}{I_2 + G(s)G_C(s)} \tag{C.3}$$

C.2 SUMP AND CYCLONE MODELS

The process transfer function $G_{11}(s)$ is given by (4.8) in Section 4.3.1 as

$$G_{11}(s) = -0.00035\frac{(1-0.63s)}{(1+0.54s)}e^{(-0.011s)} \tag{C.4}$$

and the process transfer function $G_{22}(s)$ is given by (4.12) in Section 4.3.3 as

$$G_{22}(s) = \frac{0.42}{s}. \tag{C.5}$$

The models that describe the interactions are given in the next subsections.

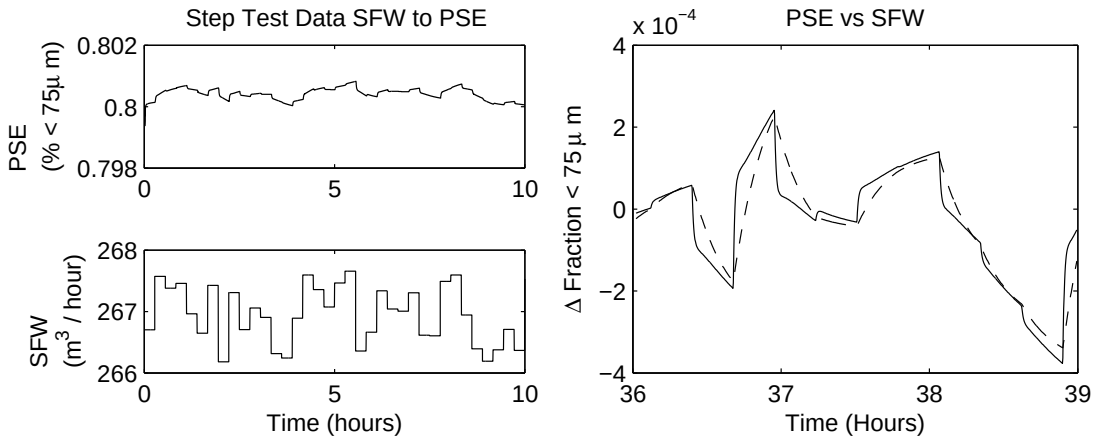
C.2.1 PSE – SFW model

The first interaction model describes the behaviour of PSE with a change in SFW. This model is represented by $G_{12}(s)$ in equation (C.1). PSE exhibits a first-order response to SFW and a first-order order transfer function model is fitted to the step test data of the nonlinear model with the following form:

$$G_{\text{PSE-SFW}}(s) = \frac{K_{\text{PS}}}{(1+P_{\text{PS}}s)}e^{(-\theta_{\text{PS}}s)} \tag{C.6}$$

$$= \frac{0.00055}{(1+0.24s)}e^{(-0.011s)} \tag{C.7}$$

The step response data for the model fitting as well as the comparison between the linear and nonlinear models are shown in Figure C.1. The linear model for PSE-CFF shows good agreement with the nonlinear model response.



(a) PSE vs SFW (b) Nonlinear vs Linear model

Figure C.1: The change in PSE with a step change in SFW. The nonlinear (solid line) model is compared to the linear model (dashed line).

C.2.2 SLEV – CFF model

The second interaction model describes the behaviour of SLEV with a change in CFF. This model is represented by $G_{21}(s)$ in equation (C.1). SLEV exhibits an integrating response to CFF and an integrator transfer function model is fitted to the step test data of the nonlinear model with the following form:

$$G_{\text{SLEV-CFF}}(s) = \frac{K_{\text{PS}}}{s} \tag{C.8}$$

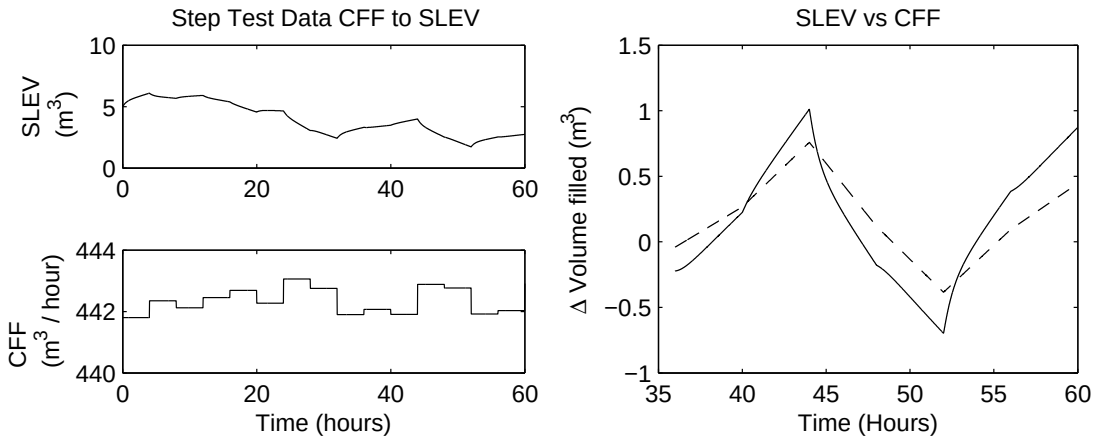
$$= \frac{-0.29}{s} \tag{C.9}$$

The step response data for the model fitting as well as the comparison between the linear and nonlinear models are shown in Figure C.2. The linear model for PSE-CFF shows good agreement with the nonlinear model response.

C.2.3 Interacting sump and cyclone model

The final model with interaction for equation (C.1) is given by

$$G(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)}e^{(-0.011s)} & \frac{0.00055}{(1+0.24s)}e^{(-0.011s)} \\ \frac{-0.29}{s} & \frac{0.42}{s} \end{bmatrix} \tag{C.10}$$



(a) SLEV vs CFF (b) Nonlinear vs Linear model

Figure C.2: The change in SLEV with a step change in CFF. The nonlinear (solid line) model is compared to the linear model (dashed line).

C.3 FREQUENCY BASED SPECIFICATIONS (FBS) TUNING METHOD

This section will give a quick overview of the FBS tuning method of Desbiens *et al.* (1996) before applying it to the interacting sump and cyclone model in the next section.

The FBS tuning method starts by describing the transfer functions that are seen by the two controllers for a 2×2 system $G(s)$ by

$$G_1(s) = G_{11}(s) - \frac{G_{12}(s)G_{21}(s)G_{C2}(s)}{1 + G_{C2}(s)G_{22}(s)} \tag{C.11}$$

$$G_2(s) = G_{22}(s) - \frac{G_{12}(s)G_{21}(s)G_{C1}(s)}{1 + G_{C1}(s)G_{11}(s)}. \tag{C.12}$$

The transfer functions (equation (C.11) and (C.12)) include the interacting controller that leads to the tuning of the one controller affecting the tuning of the other controller. There are two requirements for the closed-loop response:

1. No steady-state tracking errors, and
2. a closed-loop second-order tracking response of the setpoints.

The specifications for the tracking closed-loop responses can be specified using the following transfer function

$$\frac{Y_1(s)}{R_1(s)} = \frac{1 - \tau_{01}s}{(1 + \tau_{11}s)(1 + \tau_{21}s)} \tag{C.13}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1 - \tau_{02}s}{(1 + \tau_{12}s)(1 + \tau_{22}s)} \tag{C.14}$$

that makes provision for non-minimum phase zeros in order to produce realisable controller designs for certain plants (Desbiens *et al.*, 1996). The closed-loop tracking specifications of equation (C.13) and (C.14) are translated into open-loop characteristics given by

$$\begin{aligned} G_{OL1}(s) &= G_{C1}(s)G_1(s) \\ &= \frac{1-\tau_{01}s}{s(\tau_{11}+\tau_{21}+\tau_{01}+\tau_{11}\tau_{21}s)} \end{aligned} \tag{C.15}$$

$$\begin{aligned} G_{OL2}(s) &= G_{C2}(s)G_2(s) \\ &= \frac{1-\tau_{02}s}{s(\tau_{12}+\tau_{22}+\tau_{02}+\tau_{12}\tau_{22}s)} \end{aligned} \tag{C.16}$$

The open-loop characteristics (equation (C.15) and (C.16)) together with the tracking closed-loop specifications (equation (C.13) and (C.14)) are needed to provide enough information to solve the unknowns for $G_{C1}(s)$ and $G_{C2}(s)$, because the tuning of the one controller is dependent on the tuning of the other controller. The two controllers $G_{C1}(s)$ and $G_{C2}(s)$ can be obtained from equation (C.15) and (C.16) by rewriting the equations as

$$A_1(s)G_{C1}^2(s)+A_2(s)G_{C1}(s)+A_3(s) = 0 \tag{C.17}$$

$$B_1(s)G_{C2}^2(s)+B_2(s)G_{C2}(s)+B_3(s) = 0 \tag{C.18}$$

where

$$A_1(s) = G_{11}(s)\left(G_{11}(s)G_{22}(s)-G_{12}(s)G_{21}(s)\right)\left(1+G_{OL2}(s)\right) \tag{C.19}$$

$$\begin{aligned} A_2(s) &= G_{11}(s)G_{22}(s)\left(1-G_{OL1}(s)G_{OL2}(s)\right) \\ &\quad +\left(G_{OL2}(s)-G_{OL1}(s)\right)\left(G_{11}(s)G_{22}(s)-G_{12}(s)G_{21}(s)\right) \end{aligned} \tag{C.20}$$

$$A_3(s) = -G_{OL1}(s)G_{22}(s)\left(1+G_{OL2}(s)\right) \tag{C.21}$$

$$B_1(s) = G_{22}(s)\left(G_{11}(s)G_{22}(s)-G_{12}(s)G_{21}(s)\right)\left(1+G_{OL1}(s)\right) \tag{C.22}$$

$$\begin{aligned} B_2(s) &= G_{11}(s)G_{22}(s)\left(1-G_{OL1}(s)G_{OL2}(s)\right) \\ &\quad +\left(G_{OL1}(s)-G_{OL2}(s)\right)\left(G_{11}(s)G_{22}(s)-G_{12}(s)G_{21}(s)\right) \end{aligned} \tag{C.23}$$

$$B_3(s) = -G_{OL2}(s)G_{11}(s)\left(1+G_{OL1}(s)\right) \tag{C.24}$$

The frequency response of $G_{C1}(s)$ and $G_{C2}(s)$ can be obtained from equation (C.17) and (C.18) by calculating the frequency responses of $A_1(s), A_2(s), A_3(s), B_1(s), B_2(s)$ and $B_3(s)$ on a frequency-by-frequency basis over a spread of frequencies: $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$. The spread of frequencies should be large enough to

- capture the integrating response at low frequencies,
- capture important behaviour at high frequencies, and
- include the closed-loop cross-over frequency.

Both controllers will have two possible frequency responses, because of the quadratic nature

of equation (C.17) and (C.18). The appropriate frequency response for each controller is chosen based on the facts that

- the controller should have an integrating response, and
- the required sign of the controller is known, based on $G_{11}(s)$ and $G_{22}(s)$.

The final step is to approximate the chosen frequency response for each controller ($G_{C1}(j\omega)$ and $G_{C2}(j\omega)$) by $G_{Cp1}(s)$ and $G_{Cp2}(s)$. The form of $G_{Cp1}(s)$ and $G_{Cp2}(s)$ proposed by Desbiens *et al.* (1996) is given by

$$G_{Cp1}(s) = \frac{K_{C1}\left(1+T_1 s\right)\left(1+T_{d11}s+T_{d21}s^2\right)}{T_1 s\left(1+T_{f11}s+T_{f21}s^2\right)} \tag{C.25}$$

$$G_{Cp2}(s) = \frac{K_{C2}\left(1+T_2 s\right)\left(1+T_{d12}s+T_{d22}s^2\right)}{T_2 s\left(1+T_{f12}s+T_{f22}s^2\right)} \tag{C.26}$$

with a second-order differential term and a second-order filtering term that is needed to accurately approximate the complex frequency responses $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ produced by solving equation (C.17) and (C.18). The designs of $G_{Cp1}(s)$ and $G_{Cp2}(s)$ are first performed by hand and then refined using an optimisation technique, which is described in Desbiens *et al.* (1996).

C.4 PID CONTROLLER DESIGN

This section details the design scenarios investigated for controlling the interacting sump and cyclone model with decentralised PID for a number of closed-loop tracking specifications. The tracking specifications for the PSE-CFF loop ($Y_1(s)/R_1(s)$) must have a settling time of 5 minutes or $5/60 = 0.083$ hours. In order to reach a settling time of 0.083 hours, a dominant time constant of $0.083/4 \approx 0.021$ hours is required. The SLEV-SFW loop ($Y_2(s)/R_2(s)$) can be slower to act as a buffer to disturbances. The sign of the controller for the PSE-CFF loop should be negative or +90 degrees at low frequencies from $G_{11}(s)$ in equation (C.4). The sign of the controller for the SLEV-SFW loop should be positive or -90 degrees at low frequencies from $G_{22}(s)$ in equation (C.5). A few of the design scenarios that were tested are given in the following subsections.

C.4.1 Design scenario 1

This design is for first-order tracking specifications on both controllers, while using the full interacting model with time-delay. The time-delay is approximated using a first-order Padé

approximation. The model used for this design scenario is given by

$$G_{D1}(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)} \cdot \frac{(-s+181.8)}{(s+181.8)} & \frac{0.00055}{(1+0.24s)} \cdot \frac{(-s+181.8)}{(s+181.8)} \\ \frac{-0.29}{s} & \frac{0.42}{s} \end{bmatrix} \tag{C.27}$$

The first-order tracking specifications are given by

$$\frac{Y_1(s)}{R_1(s)} = \frac{1}{(1+0.021s)} \tag{C.28}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1}{(1+0.05s)} \tag{C.29}$$

The frequency responses for the two controllers $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ are given in Figure C.3, which shows that design 1 for $G_{C1}(j\omega)$ has an integrating response, but it does not have the correct sign and neither of the two designs of $G_{C2}(j\omega)$ has an integrating response. The method, therefore, does not provide a feasible design for the model with the design specifications given in equation (C.31) and (C.32).

C.4.2 Design scenario 2

This design is for first-order tracking specifications on both controllers, while using the interacting model without time-delay. The time-delay can be brought back through a Smith predictor structure. The model used for this design scenario is given by

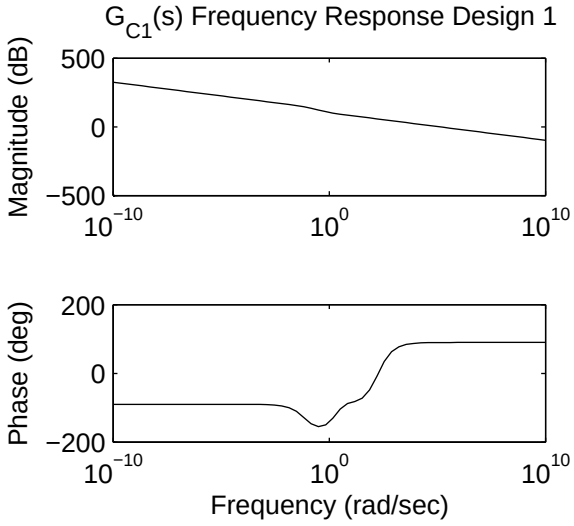
$$G_{D2}(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)} & \frac{0.00055}{(1+0.24s)} \\ \frac{-0.29}{s} & \frac{0.42}{s} \end{bmatrix} \tag{C.30}$$

The first-order tracking specifications are given by

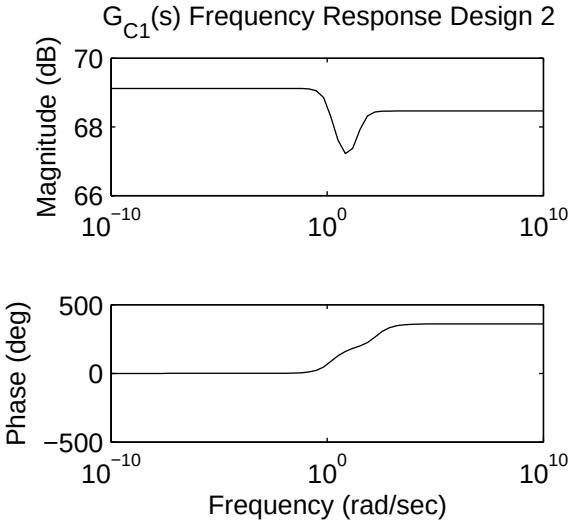
$$\frac{Y_1(s)}{R_1(s)} = \frac{1}{(1+0.021s)} \tag{C.31}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1}{(1+0.05s)} \tag{C.32}$$

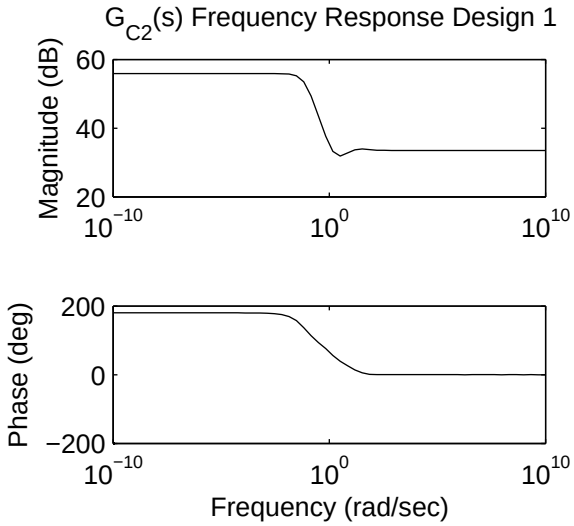
The frequency responses for the two controllers $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ are given in Figure C.4, which shows that design 1 for $G_{C1}(j\omega)$ has an integrating response, but it does not have the correct sign and neither of the two designs of $G_{C2}(j\omega)$ has an integrating response. The method, therefore, does not provide a feasible design for the model of equation (C.30) with the design specifications given in equation (C.31) and (C.32).



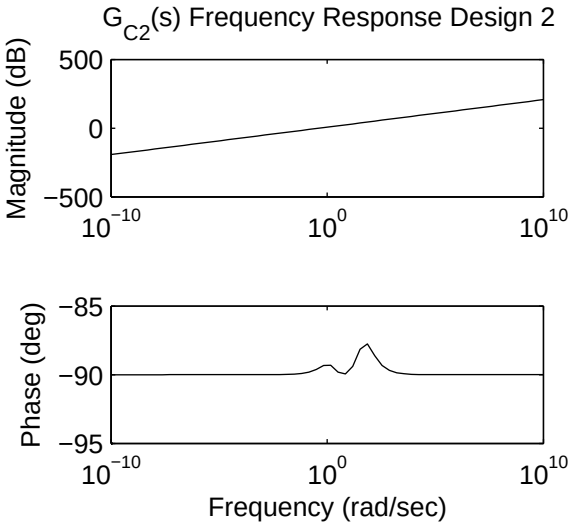
(a) $G_{C1}(j\omega)$ design 1.



(b) $G_{C1}(j\omega)$ design 2.

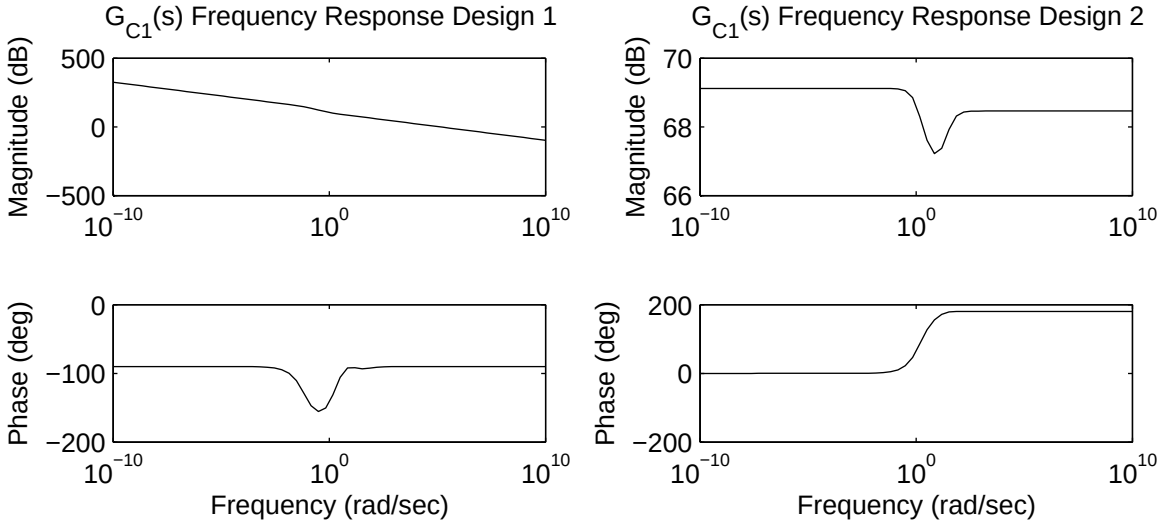


(c) $G_{C2}(j\omega)$ design 1.



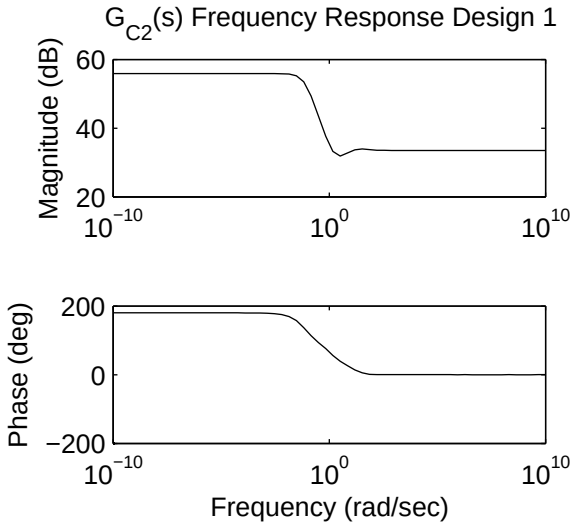
(d) $G_{C2}(j\omega)$ design 2.

Figure C.3: Design Scenario 1: Frequency response designs for $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$.

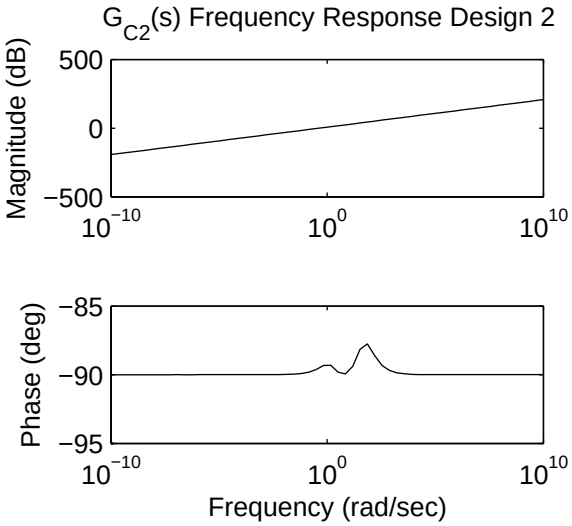


(a) $G_{C1}(j\omega)$ design 1.

(b) $G_{C1}(j\omega)$ design 2.



(c) $G_{C2}(j\omega)$ design 1.



(d) $G_{C2}(j\omega)$ design 2.

Figure C.4: Design Scenario 2: Frequency response designs for $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$.

C.4.3 Design scenario 3

This design is for first-order tracking specifications on both controllers, while using the interacting model without time-delay and the integrators approximated by fast first-order responses. The time-delay can be brought back through a Smith predictor structure. The model used for this design scenario is given by

$$G_{D3}(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)} & \frac{0.00055}{(1+0.24s)} \\ \frac{-0.29}{(s+0.001)} & \frac{0.42}{(s+0.001)} \end{bmatrix} \tag{C.33}$$

The first-order tracking specifications are given by

$$\frac{Y_1(s)}{R_1(s)} = \frac{1}{(1+0.021s)} \tag{C.34}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1}{(1+0.05s)} \tag{C.35}$$

The frequency responses for the two controllers $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ are given in Figure C.5, which shows that design 2 for $G_{C1}(j\omega)$ has an integrating response, but it does not have the correct sign and design 2 of $G_{C2}(j\omega)$ has an integrating response, but also with the incorrect sign. The method, therefore, does not provide a feasible design for the model in equation (C.33) with the design specifications given in equation (C.34) and (C.35).

C.4.4 Design scenario 4

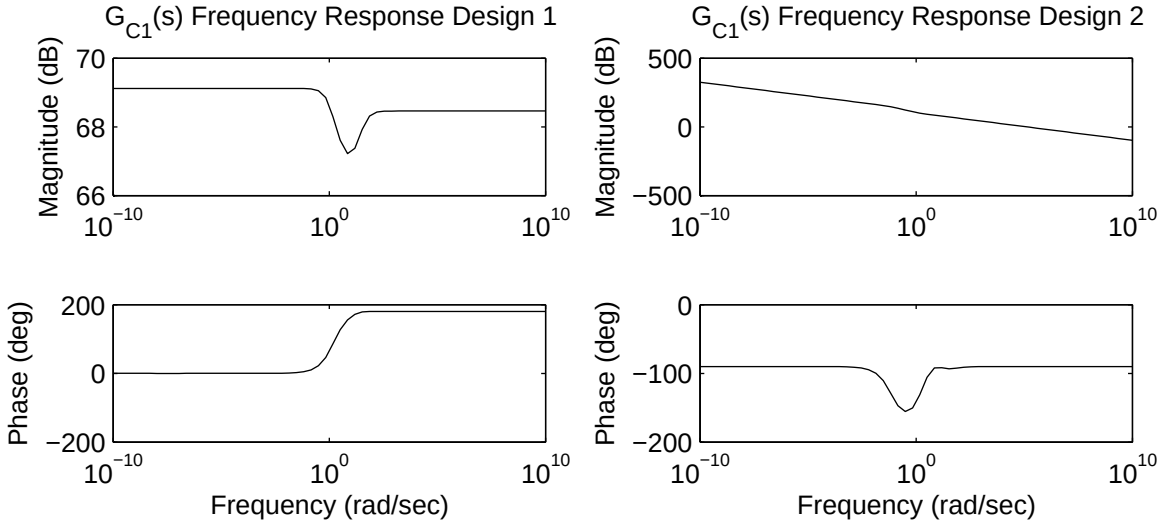
This design defines a second-order non-minimum phase closed-loop tracking specification for the PSE-CFF loop and a first-order closed-loop tracking specification for the SLEV-SFW loop, while using the interacting model without time-delay and the integrators approximated by fast first-order responses. The time-delay can be brought back through a Smith predictor structure. The model used for this design scenario is given by

$$G_{D4}(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)} & \frac{0.00055}{(1+0.24s)} \\ \frac{-0.29}{(s+0.001)} & \frac{0.42}{(s+0.001)} \end{bmatrix} \tag{C.36}$$

The closed-loop tracking specifications are given by

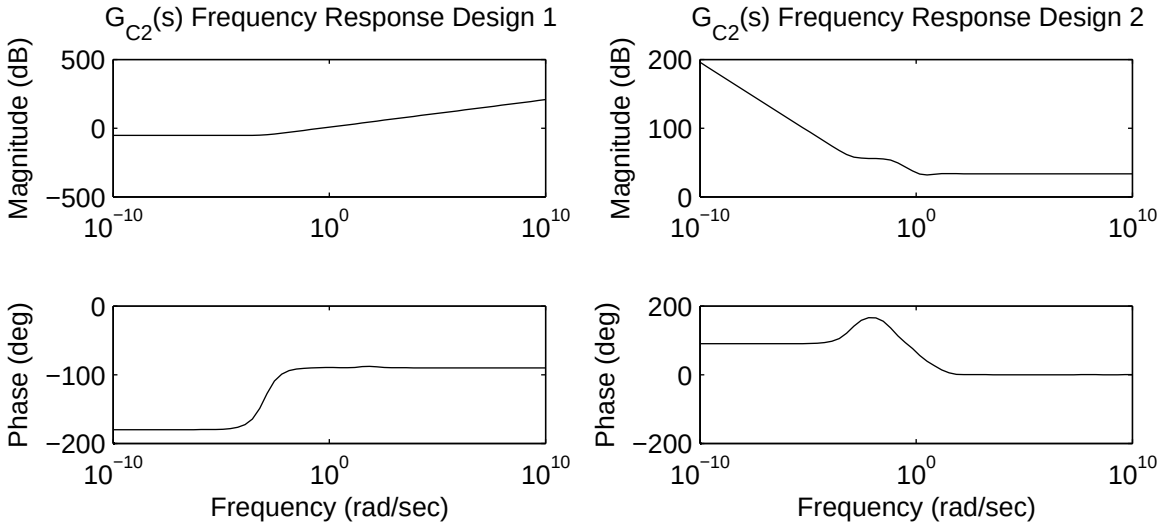
$$\frac{Y_1(s)}{R_1(s)} = \frac{(1-0.021s)}{(1+0.021s)(1+0.021s)} \tag{C.37}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1}{(1+0.05s)} \tag{C.38}$$



(a) $G_{C1}(j\omega)$ design 1.

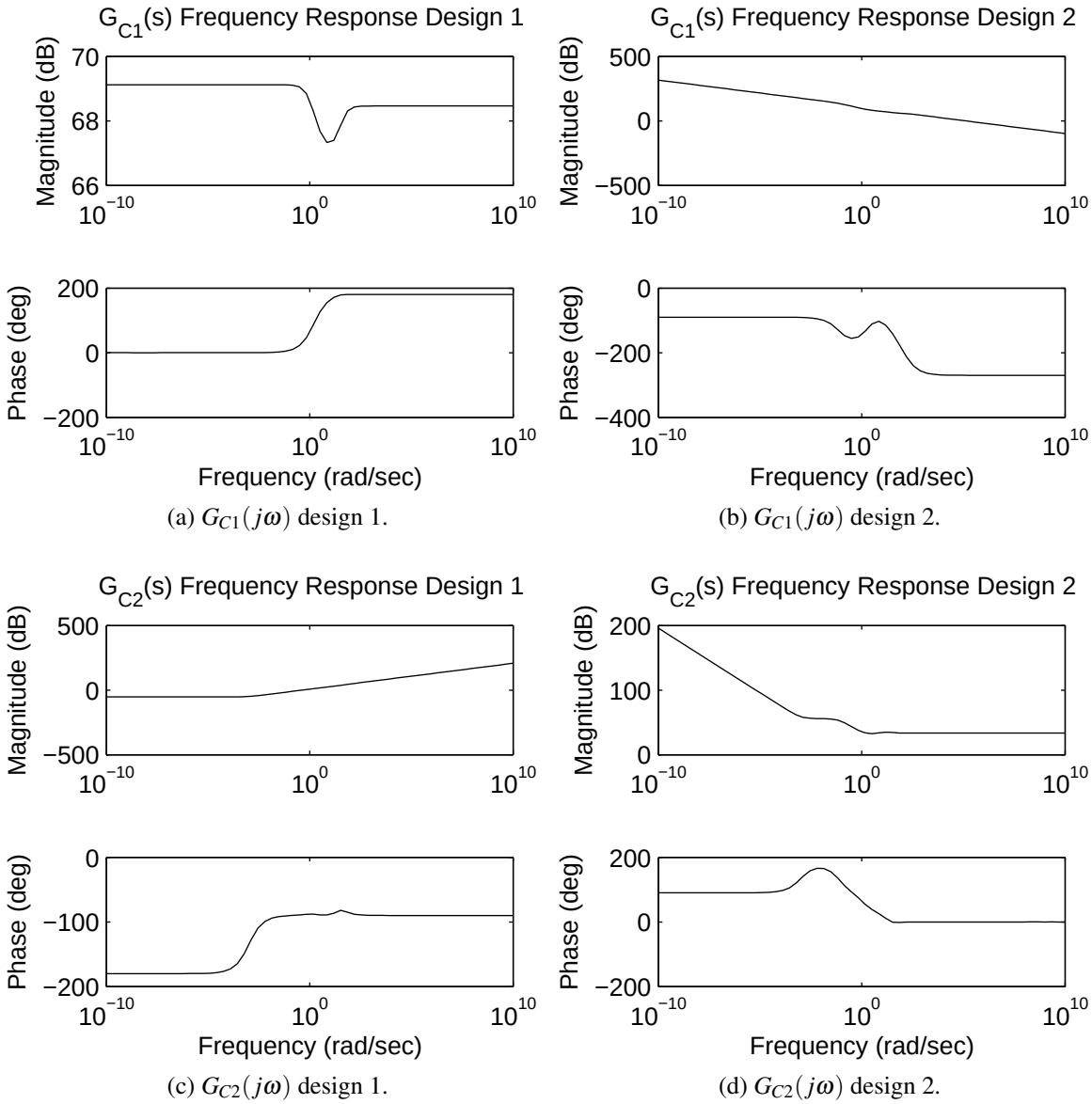
(b) $G_{C1}(j\omega)$ design 2.



(c) $G_{C2}(j\omega)$ design 1.

(d) $G_{C2}(j\omega)$ design 2.

Figure C.5: Design Scenario 3: Frequency response designs for $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$.



(a) $G_{C1}(j\omega)$ design 1.

(b) $G_{C1}(j\omega)$ design 2.

(c) $G_{C2}(j\omega)$ design 1.

(d) $G_{C2}(j\omega)$ design 2.

Figure C.6: Design Scenario 4: Frequency response designs for $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$.

The frequency responses for the two controllers $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ are given in Figure C.6. Design 2 for $G_{C1}(j\omega)$ has an integrating response, but it does not have the correct sign and the change to a second-order non-minimum phase closed-loop tracking specification only influences the high frequency phase response. Design 2 of $G_{C2}(j\omega)$ has an integrating response, but does not have the correct sign. The method, therefore, does not provide a feasible design for the model in equation (C.36) with the design specifications given in equation (C.37) and (C.38).

C.4.5 Design scenario 5

This design defines a second-order non-minimum phase closed-loop tracking specification for the PSE-CFF loop and a first-order closed-loop tracking specification for the SLEV-SFW

loop that is faster than the PSE-CFF loop. The interacting model without time-delay and the integrators approximated by fast first-order responses is used in the design. The time-delay can be brought back through a Smith predictor structure. The model used for this design scenario is given by

$$G_{D5}(s) = \begin{bmatrix} -0.00035\frac{(1-0.63s)}{(1+0.54s)} & \frac{0.00055}{(1+0.24s)} \\ \frac{-0.29}{(s+0.001)} & \frac{0.42}{(s+0.001)} \end{bmatrix} \tag{C.39}$$

The closed-loop tracking specifications are given by

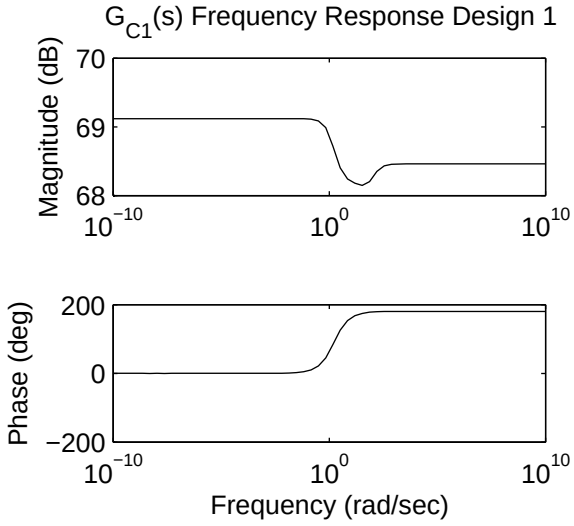
$$\frac{Y_1(s)}{R_1(s)} = \frac{(1-0.021s)}{(1+0.021s)(1+0.021s)} \tag{C.40}$$

$$\frac{Y_2(s)}{R_2(s)} = \frac{1}{(1+0.01s)} \tag{C.41}$$

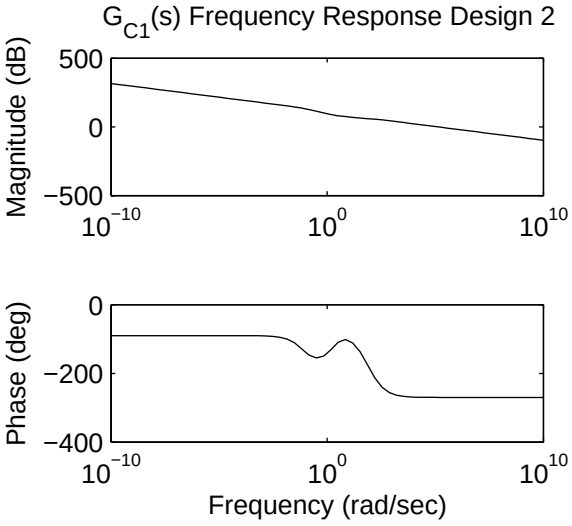
The frequency responses for the two controllers $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$ are given in Figure C.7. This specification change made no material change compared to design scenario 4, shown in Section C.4.4. The method, therefore, does not provide a feasible design for the model in equation (C.39) with the design specifications given in equation (C.40) and (C.41).

C.5 CONCLUSION

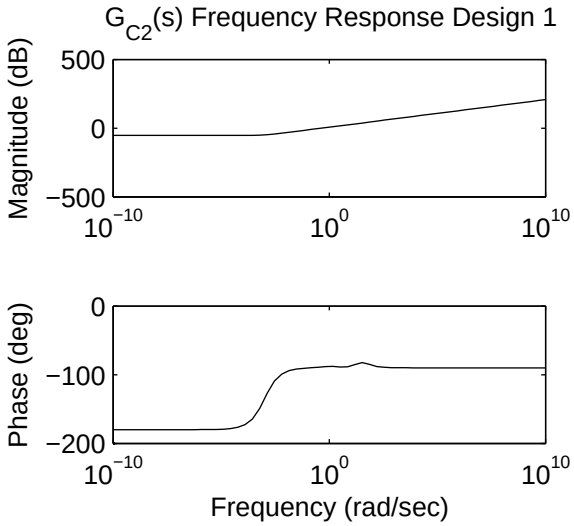
This addendum aimed to tune the decentralised PID controllers while explicitly taking interactions into consideration for the PSE-CFF and SLEV-SFW loops. A number of design scenarios were investigated in Section C.4 as well as other scenarios that were not documented here that investigated second-order closed-loop tracking specifications with and without non-minimum phase zeros for both controllers as well as slowing down the tracking specifications. Controller designs were found for both controllers with integrating responses when the integrators were approximated by fast first-order responses, but none of the model or specification changes produced controllers with integrating responses and the correct signs. This method, therefore, does not produce feasible controllers for the required closed-loop tracking specifications and valid interacting models.



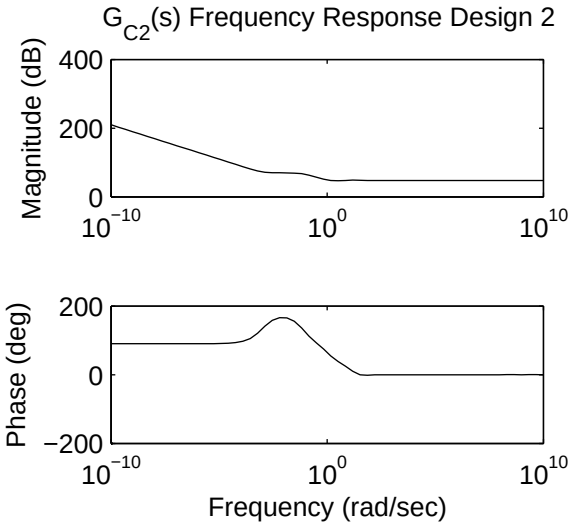
(a) $G_{C1}(j\omega)$ design 1.



(b) $G_{C1}(j\omega)$ design 2.



(c) $G_{C2}(j\omega)$ design 1.



(d) $G_{C2}(j\omega)$ design 2.

Figure C.7: Design Scenario 5: Frequency response designs for $G_{C1}(j\omega)$ and $G_{C2}(j\omega)$.

ADDENDUM D

SIMULATION SUMMARY

A summary of the simulation results are given in Table B.1 that details the tracking performance of the controller with regard to the particle size (PSE), mill load level (LOAD) and throughput. The simulation scenario is outlined in terms of the controlled variable weighting and setpoint values, as well as the step disturbances. The relevant changes in each simulation scenario are highlighted in bold. The dashes in the table represent values that are either zero or not applicable. The performance metrics are described in Section 5.2.

The headings of Table B.1 are defined as

PSE	Particle Size Estimate. [$\% < 75\mu$m]
LOAD	The volumetric filling of the mill. [%]
Throughput	The amount of solids discharged at the cyclone overflow. [$^{\text{tons}}/_{\text{hour}}$]
SLEV	Sump level. [m^3]
Power	The electrical power draw of the mill motor. [kW]
Rheology	An indication of the fluidity of the slurry inside the mill. [fraction]
U	The manipulated variables.
Disturbances	Describes the step disturbances in ore hardness, fraction of rock in the feed ore and SFW.
Time	Describes the average and maximum iteration time of the simulations [seconds].

The subheadings of Table B.1 are defined as

Δ	The sum of the squares of the error from the setpoint.
S	The setpoint of the variable.
W	The weight of the variable in the objective function.
A	The average value of the variable over the simulation duration.
RH	The increase in the hardness of the feed ore. [%]
AR	The increase in the fraction of rock in the feed ore. [%]
SFW	The increase in SFW. [$m^3/hour$]
T	The time when the disturbance is introduced.
M	The maximum value of the variable over the simulation duration.

The controllers are identified next to the figure numbers in Table B.1 by

R	Robust Nonlinear Model Predictive Controller.
N	Nonlinear Model Predictive Controller.
P	Proportional-Integral-Differential Controller.

The simulations show that RNMPC and NMPC are capable of tighter control of PSE, especially when constraints are active, because the multivariable controllers can leverage the multivariable nature of the milling circuit to increase the control envelope. The RNMPC and NMPC controllers are, however, not capable of improving throughput while maintaining PSE at the desired setpoint. The PI controllers performed well, because they were tuned very aggressively. In certain milling circuits there are large time delays that result in less aggressive tuning of the PI controllers and degraded performance. MPC controllers were found in practice to perform well over longer periods compared to PI controllers (Chen *et al.*, 2007*b*, Ramasamy *et al.*, 2005).

Table D.1: All simulations summary.

			PSE			LOAD			Throughput			SLEV		Power		Rheology		U	Disturbances						Time	
Simulation			Δ	S	W	Δ	S	W	A	S	W	S	W	S	W	S	W	W	RH	T	AR	T	S	T	A	M
Figures 5.4 & 5.5	a	R	0.05	80	100	3.20E-05	45	100	106.72	99.9	**0**	5	0	2000	0	0.51	0	**0.01**	—	—	—	—	—	—	12.8	132
Figures B.1 & B.2	a	N	0.05	80	100	1.36E-03	45	100	106.31	99.9	0	5	0	2000	0	0.51	0	0.01	—	—	—	—	—	—	0.1	2.25
Figures 5.4 & 5.5	b	R	0.05	80	100	3.60E-05	45	100	111	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	—	—	—	—	22.6	186
Figures B.1 & B.2	b	N	0.07	80	100	6.34E-04	45	100	110.27	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	—	—	—	—	0.1	1.6
Figures 5.4 & 5.5	c	P	0.15	80	—	5.35E-04	45	—	106.28	—	—	5	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Figures 5.6 & 5.7	a	R	0.07	80	100	4.70E-05	45	100	88.71	99.9	0	5	1	2000	0	0.51	0	0.01	**50**	**180**	—	—	—	—	21.7	318
Figures 5.6 & 5.7	b	N	0.08	80	100	6.36E-04	45	100	88.97	99.9	0	5	1	2000	0	0.51	0	0.01	**50**	**180**	—	—	—	—	0.1	1.52
Figures 5.6 & 5.7	c	P	0.97	80	—	4.58E-04	45	—	95.67	—	—	5	—	—	—	—	—	—	**50**	**180**	—	—	—	—	—	—
Figures 5.8 & 5.9	a	R	0.05	80	100	1.14E-04	45	100	103.5	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**180**	—	—	24.5	204
Figures 5.8 & 5.9	b	N	0.09	80	100	8.72E-04	45	100	102.86	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**180**	—	—	0.1	1.56
Figures 5.8 & 5.9	c	P	0.18	80	—	5.82E-04	45	—	105.53	—	—	5	—	—	—	—	—	—	—	—	**50**	**180**	—	—	—	—
Figures 5.10 & 5.11	a	R	0.06	80	100	2.28E-04	45	100	103.26	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**10**	—	—	27.1	291
Figures 5.10 & 5.11	b	N	0.23	80	100	7.08E-03	45	100	101.77	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**50**	**10**	—	—	0.1	1.47
Figures 5.10 & 5.11	c	P	0.41	80	—	6.32E-04	45	—	106.96	—	—	5	—	—	—	—	—	—	—	—	**50**	**10**	—	—	—	—
Figures 5.12 & 5.13	a	R	0.05	80	100	2.70E-05	45	100	110.89	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**0**	—	**50**	**180**	26.7	340
Figures 5.12 & 5.13	b	N	0.08	80	100	4.71E-04	45	100	110.28	99.9	0	5	1	2000	0	0.51	0	0.01	—	—	**0**	—	**50**	**180**	0.1	1.59
Figures 5.12 & 5.13	c	P	0.16	80	—	7.71E-04	45	—	110.30	—	—	5	—	—	—	—	—	—	—	—	**0**	—	**50**	**180**	—	—
Figures B.9 & B.10	a	R	0.10	80	100	1.59E-03	45	100	73.04	99.9	0	5	1	2000	50	0.51	50	0.01	50	10	50	100	50	30-80	36.2	665
Figures B.11 & B.12	a	N	0.10	80	100	1.42E-03	45	100	73.03	99.9	0	5	1	2000	50	0.51	50	0.01	50	10	50	100	50	30-80	0.2	2.07
Figures B.9 & B.10	b	R	0.27	80	100	9.50E-04	45	100	71.43	99.9	0	5	1	2000	50	0.51	50	0.1	50	10	50	100	50	30-80	61.3	1168
Figures B.11 & B.12	b	N	0.42	80	100	3.84E-02	45	100	71.93	99.9	0	5	1	2000	50	0.51	50	0.1	50	10	50	100	50	30-80	0.2	2.13
Figures B.9 & B.10	c	R	1.63	80	100	3.49E-01	45	100	74.81	99.9	0	5	1	2000	50	0.51	50	1	50	10	50	100	50	30-80	34.4	204
Figures B.11 & B.12	c	N	1.81	80	100	5.75E-01	45	100	74.7	99.9	0	5	1	2000	50	0.51	50	1	50	10	50	100	50	30-80	0.2	2.99
Figures 5.14 & 5.15	a	R	0.23	80	100	3.10E-03	45	100	72.22	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	34.9	1014
Figures 5.14 & 5.15	b	N	0.30	80	100	5.45E-03	45	100	72.54	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.2	1.77
Figures 5.14 & 5.15	c	P	0.77	80	—	1.52E-04	45	—	74.59	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figures B.13 & B.14	b	R	0.29	80	100	3.50E-03	45	100	73.82	99.9	1	5	1	2000	0	0.51	0	0.1	50	10	50	100	50	30-80	50.0	1295
Figures B.15 & B.16	b	N	0.28	80	100	9.11E-02	45	100	73.79	99.9	1	5	1	2000	0	0.51	0	0.1	50	10	50	100	50	30-80	0.2	2.41
Figures B.13 & B.14	c	R	1.53	80	100	3.97E-01	45	100	76.07	99.9	1	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	25.8	348
Figures B.15 & B.16	c	N	1.31	80	100	6.07E-01	45	100	75.1	99.9	1	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	0.2	3.75

Table D.3: All simulations summary (continued).

			PSE			LOAD			Throughput			SLEV		Power		Rheology		U	Disturbances					Time	
Simulation			Δ	S	W	Δ	S	W	A	S	W	S	W	S	W	S	W	W	RH	T	AR	T	S	T	A	M
Figures 5.16 & 5.17	b	R	0.12	75	100	3.35E-03	45	100	74.51	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	35.9	1132
Figures B.3 & B.4	b	N	0.19	75	100	8.36E-03	45	100	74.33	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.23
Figures 5.18 & 5.19	b	P	0.24	75	—	1.62E-04	45	—	77.98	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figures 5.16 & 5.17	c	R	0.12	70	100	3.88E-03	45	100	81.35	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	33.4	1136
Figures B.3 & B.4	c	N	0.17	70	100	6.73E-03	45	100	81.87	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.08
Figures 5.18 & 5.19	c	P	1.50	70	—	1.40E-04	45	—	81.20	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figures 5.20 & 5.21	a	R	0.15	80	100	2.65E-03	45	100	74.04	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	33.8	688
Figures 5.20 & 5.21	b	N	0.18	80	100	5.22E-03	45	100	74.22	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	9.6	58
Figures 5.20 & 5.21	c	P	0.39	80	—	1.64E-04	45	—	76.43	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figures 5.22 & 5.23	a	R	0.15	80	100	1.80E-03	45	100	76.73	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	39.3	1127
Figures 5.22 & 5.23	b	N	0.14	80	100	4.74E-03	45	100	78.11	99.9	1	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	10.3	70
Figures 5.22 & 5.23	c	P	0.92	80	—	1.74E-04	45	—	78.52	—	—	5	—	—	—	—	—	—	50	10	50	100	50	30-80	—	—
Figures 5.24 & 5.25	a	R	2.79	80	100	5.87E-02	45	100	71.14	99.9	5	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	21.0	813
Figures B.5 & B.6	a	N	3.31	80	100	1.06E-01	45	100	71.45	99.9	5	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	2.04
Figures 5.24 & 5.25	b	R	7.09	80	100	1.70E-01	45	100	70.76	99.9	10	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	15.2	619
Figures B.5 & B.6	b	N	7.89	80	100	3.43E-01	45	100	69.25	99.9	10	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	1.77
Figures 5.26 & 5.27	b	R	5.52	80	100	8.20E-02	45	100	74.85	99.9	10	5	1	2000	0	0.51	0	0.1	50	10	50	100	50	30-80	49.2	1215
Figures B.7 & B.8	b	N	4.71	80	100	5.86E-02	45	100	75.56	99.9	10	5	1	2000	0	0.51	0	0.1	50	10	50	100	50	30-80	0.2	2.3
Figures 5.26 & 5.27	c	R	4.11	80	100	1.21E-01	45	100	75.6	99.9	10	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	33.2	987
Figures B.7 & B.8	c	N	4.79	80	100	7.87E-01	45	100	75	99.9	10	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	0.2	6.43
Figures 5.24 & 5.25	c	R	10.01	80	100	5.15E-01	45	100	68.09	99.9	20	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	13.3	682
Figures B.5 & B.6	c	N	13.01	80	100	5.00E-01	45	100	70.83	99.9	20	5	1	2000	0	0.51	0	0.01	50	10	50	100	50	30-80	0.3	1.38
Figures B.17 & B.18	b	R	9.19	80	100	2.69E-02	45	100	76.83	99.9	20	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	44.7	1171
Figures B.19 & B.20	b	N	7.52	80	100	1.32E+00	45	100	75.73	99.9	20	5	1	2000	0	0.51	0	1	50	10	50	100	50	30-80	0.2	3.77