Optimisation of Smart Grid Performance using Centralised and Distributed Control Techniques

Paul Mc Namara

20^{th} Febraury 2012

A Thesis Submitted to the
National University of Ireland
in Fulfillment of the Requirements for
the Degree of
Doctor of Philosophy

Supervisor: Dr. Gordon Lightbody
Head of Department: Prof. Nabeel A. Riza
Department of Electrical and Electronic Engineering,
National University of Ireland, Cork.

Abstract

A massive change is currently taking place in the manner in which power networks are operated. Traditionally, power networks consisted of large power stations which were controlled from centralised locations. The trend in modern power networks is for generated power to be produced by a diverse array of energy sources which are spread over a large geographical area. As a result, controlling these systems from a centralised controller is impractical. Thus, future power networks will be controlled by a large number of intelligent distributed controllers which must work together to coordinate their actions. The term Smart Grid is the umbrella term used to denote this combination of power systems, artificial intelligence, and communications engineering.

This thesis focuses on the application of optimal control techniques to Smart Grids with a focus in particular on iterative distributed MPC. A novel convergence and stability proof for iterative distributed MPC based on the Alternating Direction Method of Multipliers is derived. Distributed and centralised MPC, and an optimised PID controllers' performance are then compared when applied to a highly interconnected, nonlinear, MIMO testbed based on a part of the Nordic power grid. Finally, a novel tuning algorithm is proposed for iterative distributed MPC which simultaneously optimises both the closed loop performance and the communication overhead associated with the desired control.

Acknowledgements

Getting through and finishing this PhD thesis has been one of the toughest but ultimately most rewarding experiences of my life so far. It would have been impossible to see this through were it not for the support I had from a number of people. There were too many people who helped me out to name them all so I'll try and remember as many people as possible here, and apologise in advance to anyone I've forgotten.

In true control engineering fashion Gordon Lightbody seems to have a very good model for how to allow a PhD student to reach his or her goals. I can't honestly say if I'd have gotten through this process with another supervisor and he was at all times highly supportive, understanding, and encouraging. Similarly I received a massive amount of help from Rudy Negenborn and Bart de Schutter in Delft, and thoroughly enjoyed working and meeting with them at various stages over the last couple of years. Also, it helps with your work when you have people to help with the non-engineering side of things. Geraldine, Ralph, Mary, and Niamh were all life savers at various stages over the last 4 and a half years and were always more than helpful in any dealings with the college administration.

I've made some great friends too over the course of this couple of years, most formed over the course of endless tea breaks. There are too many of the (mostly) guys to name here so ye know who ye are. It was great to have such a great bunch of people in the same boat as me over the years. It was also great to get the opportunity to talk control, and other things, with Julien Cretel and Samira Roshany over the years.

Last but not least I'd like to thank all my other friends, my parents and family, and most of all my wonderful girlfriend Emily who put up with the brunt of a lot of my worse off days in the PhD, and were delighted for me things finally came together. I couldn't have done this without ye,

Paul Mc Namara

Published Papers

"Improving Distributed Model Predictive Control Performance Via Weight Optimization using PSO", Paul Mc Namara, Gordon Lightbody, 2nd IFAC International Conference on Intelligent Control Systems and Signal Processing (ICONS), September 21-23, 2009, Istanbul, Turkey.

"PSO optimized PID parameters for coupled HVDC control", Paul Mc Namara, Gordon Lightbody, Irish Signals and Systems Conference, June 23-24, 2010, Cork, Ireland.

"Coordination of a Multiple Link HVDC System Using Local Communications Based Distributed Model Predictive Control", Paul Mc Namara, Rudy R. Negenborn, Bart de Schutter, Gordon Lightbody, IFAC World Congress, Milan, Italy, August 28-September 2, 2011.

"Optimal coordination of a multiple HVDC link system using centralised and distributed control", Paul Mc Namara, Rudy R. Negenborn, Bart de Schutter, Gordon Lightbody, accepted for publication in IEEE Transactions on Control Systems Technology.

"Weight Optimisation for iterative distributed Model Predictive Control applied to Power Networks", Paul Mc Namara, Rudy R. Negenborn, Bart de Schutter, Gordon Lightbody, submitted to Engineering Applications of Artificial Intelligence.

"Convergence and Stability proof for unconstrained distributed Model Predictive Control", Paul Mc Namara, Rudy R. Negenborn, Bart de Schutter, Gordon Lightbody, currently being written for submission to Automatica.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern society is built upon a foundation of complex, highly interconnected systems. The traffic and transportation systems, international economic markets, information superhighways, and electrical systems, to name but a few, are the invisible infrastructures which have enabled the rapid development seen in societies across the globe in the last few decades. In fact, it is only on the occasions when something goes wrong with these systems that it becomes obvious to the general public just how reliant the smooth operation of these systems is to the functioning of society. The failure of these systems can be all the more confusing to the average person given the apparent simplicity of these systems at the individual action level, e.g., the turning on of a light switch, looking for some information in a search engine, or buying shares in the stock market. However, it is when hundreds or thousands of these actions are taken collectively together with the dynamic interactions between various system elements, that the complex and often unpredictable nature of these systems arises. The goal of systems and control engineers has, thus, always been to comprehend the complexities inherent in these systems and to minimise the potential effects of unpredictable system elements on the operation of these systems.

1.1 From Centralised to Distributed Control

The origins of classical feedback control date back to 1868 when J.C. Maxwell published his paper on steam engine regulation using centrifugal governors (Maxwell, 1868). It was Nyquist's 1932 paper, however, that provided principles which could be applied to virtually any feedback system (Nyquist, 1932) and over the next 20 years a solid theoretical foundation in frequency domain methods was established through the work of Nyquist, Bode, Nichols, and Evans (Bennett, 1996). These classical techniques were based on continuous-time system models. The 1950s saw the dawn of discrete-time control, due to the arrival of digital computers. After half a century of research and implementation, the foundations of digital control theory are firmly established and have permeated across a multitude of fields (Tewari, 2002). It was during the 1950s, too, that the area of optimal control was introduced when Bellman developed the concept of Dynamic Programming (Bellman, 1952; Bryson, 1996). Advanced control techniques such as Reinforcement Learning (Sutton and Barto, 1998) and Model Predictive Control (MPC) (Maciejowski, 2002; Rossiter, 2003) have arisen from the field of optimal control as tractable methods for solving the Dynamic Programming problem.

Typically, decentralised control structures are used when controlling large scale systems. Centralised control is typically problematic in large scale systems due to issues with time delays, and because it is undesirable to have to transmit the large volume of data needed to control all of the subsystems over a network (Baillieul and Antsaklis, 2007). In decentralised control structures, inputs $\boldsymbol{u}$ and outputs $\boldsymbol{y}$ are grouped into disjoint sets. Inputs and outputs are then coupled into non-overlapping pairs, and independent regulators are designed for each of these pairs which operate independently of each other (Scattolini, 2009), as can be seen in Fig. 1.1. Given the relevance of decentralised control, many methods have been designed which offer guaranteed stable closed loop control (Scattolini, 2009). Decentralised control methods have been successfully applied to a wide variety of systems including irrigation networks (Cantoni et al., 2007), chemical plants (Ricker, 1996), electrical power systems (Siljak et al., 2002) and teams of moving vehicles (Siljak and Zecevic, 2005). However, it is known that decentralised control may not be stabilising where there is strong interactions between subsystems (Venkat, 2006; Scattolini, 2009). Also, in systems with Decentralised Fixed Modes (DFMs), the system may not be controllable using a decentralised controller. DFMs are a subset of the systems open-loop eigenvalues

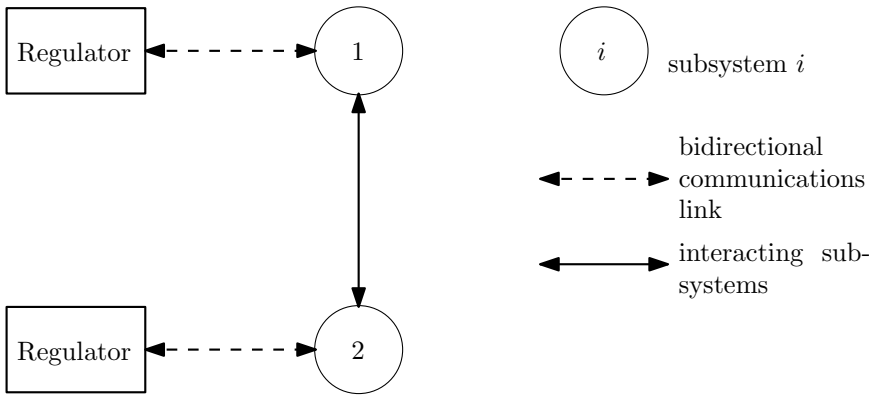


Figure 1.1: Decentralised control of 2 interacting subsystems.

which have the property that they remain fixed, independent of any linear time-invariant controller (subject to the given decentralized control information constraint), which may be applied to the system (Davison and Chang, 1990). Also, many decentralised control techniques rely on the system having a certain type of state-space formulation and it may not always be possible to describe the system in this way (Siljak and Zecevic, 2005).

Distributed control systems can overcome a lot of the disadvantages inherent in decentralised control by allowing a certain degree of communication between regulators, as can be seen for the simple system in Fig. 1.2. The development of distributed control systems in fact, provides the potential for the improved overall performance of networked control systems, due to reduced delays in sensing and actuation in comparison with centralised control systems (Baillieul and Antsaklis, 2007). Hierarchical control systems can also be created where higher level coordinating controllers can provide coordinating setpoints to lower level controllers. This can be seen with the optional coordination layer in Fig. 1.2 (Scattolini, 2009). The development of distributed control was made possible by the development of cheap microprocessors which could relate information to each other over a communications network. This development was further fuelled by the savings which could be made through the potential reduction in the use of expensive wiring and the flexibility with which additional components could be introduced to systems as their needs changed. One of the earliest efforts at distributed networked control began in 1983 when Bosch GmbH began a feasibility study into the use of networked devices to control different functions in passenger vehicles. From this the Control Area Network (CAN) communications protocol was introduced in 1986 and today networked control systems are found in abundance in industry. The use of wireless communications with these technologies

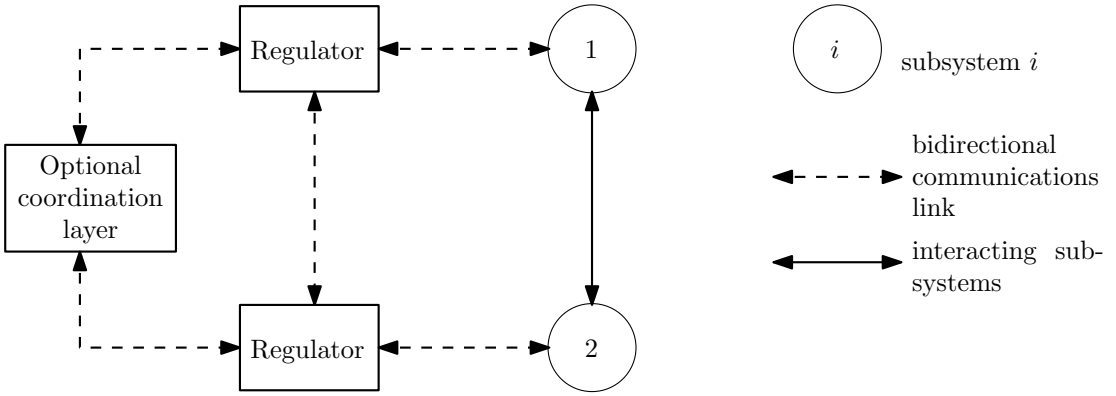


Figure 1.2: Distributed control of 2 interacting subsystems.

introduces further advances and challenges for networked and distributed control systems (Baillieul and Antsaklis, 2007). MPC has proved to be particularly suited for distribution over a number of processors, typically using methods based on distributed optimisation, game theory or using contracting constraints in order to coordinate the actions of the different controllers (Scattolini, 2009). Distributed control has been applied in a number of areas including process control (Christofides, 2001), power networks (Negenborn, 2007; Camponogara et al., 2002), coordinated vehicle control (Dunbar and Murray, 2002), water network control (Trnka et al., 2011; Javalera et al., 2010), and supply chain management (Maestre et al., 2009).

1.2 Methods for Smart Grid Control

A number of challenges face electrical energy providers and regulators at this moment that cannot be addressed adequately with the current power grid infrastructure. The grid in its current form limits the amount of clean, renewable sources such as wind and solar that can be integrated into the grid and currently is therefore heavily reliant on polluting, carbon based fuels. Also, in its current form, the grid does not allow for the extensive integration of the small scale energy production sources, such as micro-CHP units and small scale solar panels and wind turbines. In addition to this, the current grid is limited in its ability to react to serious grid faults which are becoming increasingly common (Farhangi, 2010). The next generation electricity grid, which is known as the Smart Grid, will enable greater control of the grid to make up for its current shortcomings.

Smart Grid technology combines the areas of power systems, control, and communications engineering. It maximises the potential utilisation that can be made out of the established power system architecture and allows individual consumers significantly more autonomy in both how they consume energy and interact with the electrical grid.

Modern optimisation and control techniques will be used extensively with new Smart Grids. The use of advanced control techniques and optimisation for power system control is already quite common. Stochastic optimisation algorithms such as Genetic Algorithms (Rerkpreedapong et al., 2003b; Chiang, 2005; Liu et al., 2010) and Particle Swarm Optimisation (AlRashidi and El-Hawary, 2009) have been used extensively to tune controller parameters in power systems. Neural Networks allow non-linearities to be modelled in power systems and can be used to improve power system control over linear control techniques (Anis Ibrahim and Morcos, 2002). Fuzzy systems also allow non-linearities to be modelled and can allow knowledge from practitioners to be incorporated into the power systems control algorithms (Shayeghi et al., 2009). There is much potential for the use for Reinforcement Learning in power grids, due to its ability to learn how to control systems based on experience and without the need for a model (Ernst et al., 2004; Nanduri and Das, 2007; Momoh, 2009).

Centralised Model Predictive Control has also been very successful in providing a high level of control for a variety of power system problems. MPC is particularly attractive for power systems control due to the ease with which constraints can be dealt with and the intuitive manner in which it is formulated (Xie and Ilic, 2009; Negenborn et al., 2009; Rerkpreedapong et al., 2003a). However, centralised MPC is not suited for the control of large scale power systems as the control problem becomes too complex to handle in real-time. Distributed MPC on the other hand is well suited for use in large scale power networks and can provide the level of flexibility in control that is desired for use with Smart Grids (Negenborn, 2007; Camponogara et al., 2002; Camacho et al., 2011; Moradzadeh et al., 2011). However, there is still a considerable amount of work needed on distributed MPC techniques before they become viable for use in power networks. Issues relating to the application of distributed MPC in Smart Grids will be discussed in great detail in the following chapter.

1.3 Scope and Aims of this Thesis

The Smart Grid will enable the simultaneous solution of a number of power grid control goals in real-time. The satisfaction of these goals will be based on the manipulation of both continuous and discrete control variables. Optimisation based techniques are the most promising methods for allowing these different objectives to be fulfilled.

In situations where linear models provide an accurate approximation to the real system model, full advantage can be taken of the host of deterministic optimisation techniques that are available in order to control the system. In cases where an accurate model of the system cannot be attained, and often in cases where the system is highly nonlinear, stochastic optimisation methods can be used instead of deterministic optimisation methods for the design of suitable control systems.

However, one the most desirable features of the Smart Grid is that individual subsystems would have autonomy over their decisions. Also, in modern deregulated power networks, separate companies provide electricity to different parts of the grid. In these deregulated systems there may be more than one control system responsible for the control of the grid. Individual countries will also typically have their own control centres, even though grids may cross the borders of adjacent countries. Therefore, non-centralised multi-agent control techniques are needed to control these grids. Also, the level of communication possible between individual controllers may be limited, particularly for processes that take place at small time scales, such as frequency control. Stability guarantees and methods of tuning controllers to give the desired control performance will also be necessary.

This thesis investigates the optimal control of Smart Grids. Optimal control techniques are applied to a number of systems which include discrete, continuous, linear and non-linear systems. Of particular interest is distributed Model Predictive Control for which stability and tuning issues are addressed.

1.4 Thesis Outline

This thesis consists of seven chapters. This chapter establishes the progression from centralised to distributed control systems and briefly outlines the reasons for the application of distributed control techniques to future smart power networks. The scope and aims of the thesis were then presented.

Chapter 2 describes the manner in which power networks were originally constructed and how it is proposed that they will develop in the coming years. The chapter then provides an introduction to the power electronic devices that will enable the advanced control of power networks. Issues surrounding the control of future power grids are then discussed at length.

Chapter 3 introduces Model Predictive Control in detail. A simulation is carried out in which MPC is applied to a highly interconnected 3-area Load Frequency Control (LFC) problem.

In chapter 4, the iterative distributed MPC algorithm developed in (Negenborn, 2007) is presented. A novel convergence and stability proof is given for the unconstrained linear version of this algorithm. In the final section, the stability proof is applied to a 2-area discrete time Load Frequency control problem and the effects of various system parameters on the determination of the closed loop stability are discussed.

In chapter 5, a multiple link HVDC power system based on part of the Nordic power grid is introduced. This is a highly interconnected, nonlinear, MIMO system. A multi-loop PID controller, whose performance has been optimised using Particle Swarm Optimisation (PSO), is first applied to this system. The performance of the optimised PID controller is then compared to the performance of a centralised and distributed MPC controller. It is also shown how the distributed MPC can be extended to deal with systems where controllers have shared inputs.

In chapter 6, a novel PSO-based tuning algorithm is presented for iterative distributed MPC. This algorithm is capable of simultaneously optimising both the closed-loop performance and communication overhead of the distributed MPC, via the tuning of the

distributed MPC weights. This tuning algorithm is applied in two situations where systems are being controlled using distributed MPC. One of the systems is a 20 area discrete time LFC and the other system is the multiple link HVDC system which will be presented in chapter 4.

In chapter 7, a brief summary will be given of the results in the thesis, and some conclusions will be made. Also, potential future work that could be carried out in the areas outlined in this thesis will be given.

References

M.R. AlRashidi and M.E. El-Hawary. A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation*, 13(4):913 –918, Aug. 2009.

W.R. Anis Ibrahim and M.M. Morcos. Artificial intelligence and advanced mathematical tools for power quality applications: a survey. *IEEE Transactions on Power Delivery*, 17(2):668 –673, Apr. 2002.

J. Baillieul and P.J. Antsaklis. Control and communication challenges in networked real-time systems. *Proceedings of the IEEE*, 95(1):9 –28, Jan. 2007.

R. Bellman. On the theory of dynamic programming. Technical report, The RAND Corporation, Santa Monica, California, 1952.

S. Bennett. A brief history of automatic control. *IEEE Control Systems*, 16(3):17 –25, June 1996.

Jr. Bryson, A.E. Optimal control-1950 to 1985. *IEEE Control Systems*, 16(3):26 –33, June 1996.

E. F Camacho, A. J. Del Real, C. Bordons, and A. Arce. Solar Thermal Plants Integration in Smart Grids. In *Proceedings of the 18th IFAC World Congress*, volume 18, pages 4939–4944, Sept 2011.

E. Camponogara, D. Jia, B.H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22(1):44–52, Feb 2002.

M. Cantoni, E. Weyer, Y. Li, S.K. Ooi, I. Mareels, and M. Ryan. Control of large-scale irrigation networks. *Proceedings of the IEEE*, 95(1):75 –91, Jan. 2007.

C.L. Chiang. Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels. *IEEE Transactions on Power Systems*, 20(4):1690–1699, Nov. 2005.

P. D. Christofides. Control of nonlinear distributed process systems: Recent developments and challenges. *AIChE Journal*, 47(3):514–518, 2001.

E.J. Davison and T.N. Chang. Decentralized stabilization and pole assignment for general proper systems. *IEEE Transactions on Automatic Control*, 35(6):652–664, June 1990.

W.B. Dunbar and R.M. Murray. Model predictive control of coordinated multi-vehicle formations. In *Proceedings of the 41st IEEE Conference on Decision and Control*, volume 4, pages 4631–4636, California, USA, Dec. 2002.

D. Ernst, M. Glavic, and L. Wehenkel. Power systems stability control: reinforcement learning framework. *IEEE Transactions on Power Systems*, 19(1):427 – 435, Feb. 2004.

H. Farhangi. The path of the smart grid. *IEEE Power and Energy Magazine*, 8(1):18 –28, Jan. 2010.

V. Javalera, B. Morcego, and V. Puig. Negotiation and Learning in distributed MPC of Large Scale Systems. In *American Control Conference*, Baltimore, Maryland, July 2010.

S.C. Liu, J.H. Zhang, Z.Q. Liu, and H.Q. Wang. Reactive power optimization and voltage control using an improved genetic algorithm. In *International Conference on Power System Technology (POWERCON)*, pages 1–5, Oct. 2010.

J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, England, 2002.

J.M. Maestre, D.M. de la Pea, and E.F. Camacho. Distributed MPC: a supply chain case study. In *Proceedings of the 48th IEEE Conference on Decision and Control*, pages 7099 –7104, Dec. 2009.

J.C. Maxwell. On governors. *Proceedings of the Royal Society of London*, 16:270–283, 1868.

J.A. Momoh. Smart grid design for efficient and flexible power networks operation and control. In *IEEE/PES Power Systems Conference and Exposition*, pages 1–8, Washington DC, USA, Mar. 2009.

M. Moradzadeh, L. Bhojwani, and R. Boel. Coordinated voltage control via distributed model predictive control. In *Chinese Control and Decision Conference (CCDC)*, pages 1612 –1618, Mianyang, China, May 2011.

V. Nanduri and T. K. Das. A Reinforcement Learning Model to Assess Market Power Under Auction-Based Energy Pricing. *IEEE Transactions on Power Systems*, 22(1): 85–95, Feb. 2007.

R. R. Negenborn. *Multi-Agent Model Predictive Control with Application to Power Networks*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 2007.

R.R. Negenborn, S. Leirens, B. De Schutter, and J. Hellendoorn. Supervisory nonlinear mpc for emergency voltage control using pattern search. *Control Engineering Practice*, 17(7):841–848, 2009.

H. Nyquist. Regeneration theory. *Bell Systems Technical Journal*, 11:126–147, 1932.

D. Rerkpreedapong, N. Atic, and A. Feliachi. Economy oriented model predictive load frequency control. In *Large Engineering Systems Conference on Power Engineering*, pages 12–16, May 2003a.

D. Rerkpreedapong, A. Hasanovic, and A. Feliachi. Robust load frequency control using genetic algorithms and linear matrix inequalities. *IEEE Transactions on Power Systems*, 18(2):855 – 861, May 2003b.

N.L. Ricker. Decentralized control of the tennessee eastman challenge process. *Journal of Process Control*, 6(4):205 – 221, 1996.

J.A. Rossiter. *Model Based Predictive Control-A Practical Approach*. CRC Press, Florida, 2003.

R. Scattolini. Architectures for distributed and hierarchical Model Predictive Control - A review. *Journal of Process Control*, 19(5):723–731, 2009.

H. Shayeghi, H.A. Shayanfar, and A. Jalili. Load frequency control strategies: A state-of-the-art survey for the researcher. *Energy Conversion and Management*, 50(2):344 – 353, 2009.

D.D. Siljak and A.I. Zecevic. Control of large-scale systems: Beyond decentralized feedback. *Annual Reviews in Control*, 29(2):169 – 179, 2005. ISSN 1367-5788.

D.D. Siljak, D.M. Stipanovic, and A.I. Zecevic. Robust decentralized turbine/governor control using linear matrix inequalities. *IEEE Transactions on Power Systems*, 17(3): 715 – 722, Aug. 2002.

Sutton and Barto. *Reinforcement Learning*. 1998.

A. Tewari. *Modern Control Design with Matlab and Simulink*, volume 1. John Wiley & Sons, Ltd., New York, USA, 2002.

P. Trnka, J. Pekar, and V. Havlena. Application of Distributed MPC to Barcelona Water Distribution Network. In *Proceedings of the 18th IFAC World Congress*, Milan, Italy, Aug.-Sept. 2011.

A. Venkat. *Distributed Model Predictive Control: Theory and Applications*. PhD thesis, University of Wisconsin-Madison, Wisconsin, 2006.

L. Xie and M.D. Ilic. Model predictive economic/environmental dispatch of power systems with intermittent resources. In *IEEE Power Energy Society General Meeting*, pages 1–6, Pittsburgh, Pennyslvania, USA, July 2009.

Chapter 2

Modern Power Networks and Control

2.1 Power Systems: Tradition and Evolution

Power networks are large, complex, highly interconnected systems. The high level of interconnectivity in power networks arises from the dynamic interactions between system components such as generators, transmission lines, energy storage devices, and loads (Kundur, 1994). The basic objective of a power system is to provide active and reactive power to consumers, within tight frequency and voltage bounds. In addition, this power production must be scheduled so as provide the desired amount of power in the right place, at the right time. Traditionally, power systems consisted of regulated utilities which had a monopoly franchise over the areas they provided to (Masters, 2004). A small number of large, centralised power stations generated electrical power, which was then consumed by end users, as can be seen in Fig. 2.1. Typically the directions of power flow in such systems were predictable, flowing from the large scale producers to the smaller scale consumers. However, radical changes in power systems structure have arisen in recent years for a number of reasons.

Global warming has been a large factor in the need for these changes. The dominant

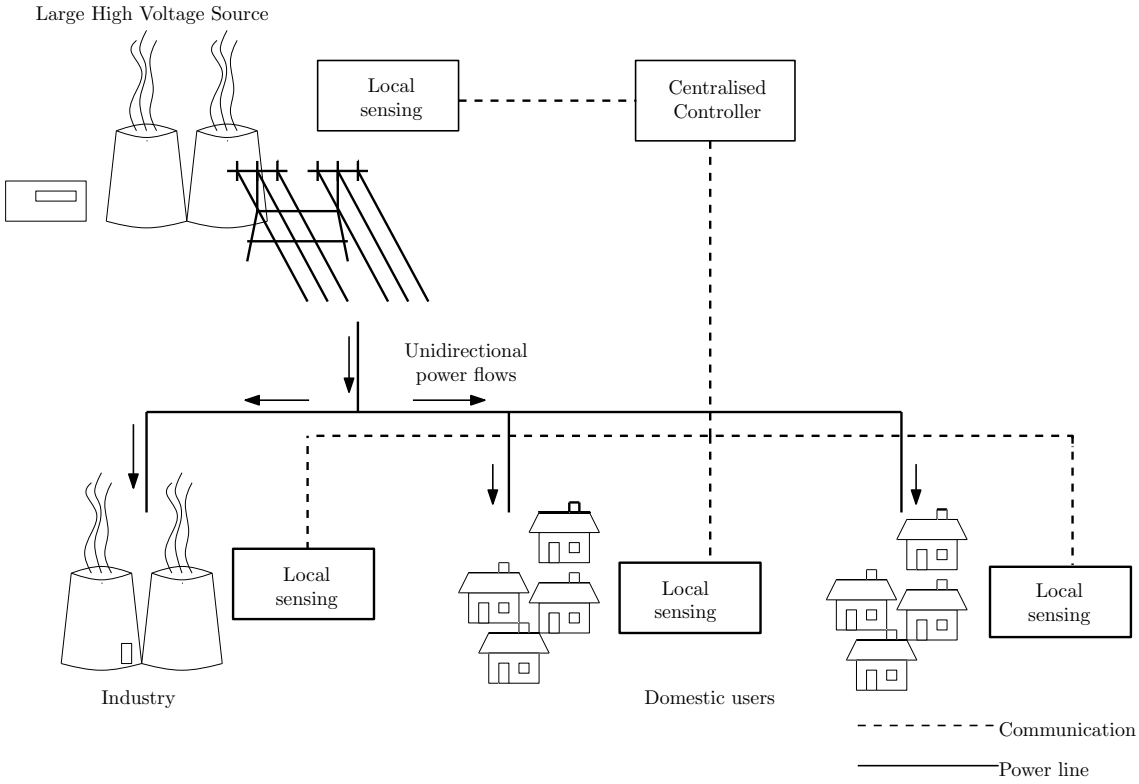


Figure 2.1: Traditional centralised monopolistic power distribution.

forms of power generation have typically involved the burning of large quantities of fuels such as coal, oil or gas. In the process of burning these fuels, large quantities of CO_2 gas and other greenhouse gases are released. The overwhelming scientific consensus is that these gases are a major contributor to global warming. In response to this, there has been a concerted effort around the world to reduce the consumption of these fuels, with many countries having signed up to the Kyoto protocol (United Nations, 1998). Those countries who have not (most notably the United States), have still been investing major amounts of money in green energy research, with global clean energy investment reaching a record US$ 243 billion in 2010 (Liebreich et al., 2011). As a result, there has been an increasing penetration of new clean forms of energy production into electricity markets across the world. However, many of these new forms of energy production, such as wind and solar, are quite different from traditional sources. They cannot be called upon to generate desired levels of energy on demand and so can be less predictable than traditional energy sources. Often they are combined with some sort of storage (Korpas and Holen, 2006), or combined with backup generation (Dufo-López and Bernal-Agustín, 2005) to compensate for disturbances in energy production that may occur using these forms of energy harvesting. For example, with wind turbines, if the wind speed falls

outside certain bounds, then the turbine will shut down until the wind speed is within these bounds again. Also, the areas most suited to the production of clean forms of energy, particularly in the cases of wind and solar power generation, may not be located near the areas of highest electricity demand (Tande, 2000), resulting in the generation of power taking place over vast geographical areas.

Another driving force behind the change in power network structure has been the deregulation of power generation. This began in the 1980's, where the success in deregulating traditional monopolies such as the telecommunications, airline and gas industries provided evidence that the electricity industry could also be successfully deregulated. This started with the introduction of small-scale power plants, especially gas turbines and combined-cycle plants, that offered both reduced first cost and operating costs compared with almost all of the generation facilities already on line (Masters, 2004). Cogeneration or Combined Heat and Power (CHP), in particular, allowed the generation of both heat and electricity simultaneously and so industries producing waste heat could then use it for electricity production. The introduction in recent years of small scale micro-CHP units, capable of generating up to 50kW, now enable individual households to also become electricity producers (Paepea et al., 2006). Similarly, small scale wind, solar and micro-hydroelectric generators allow small scale production of electricity by individual households or communities (Masters, 2004; Kellogg et al., 1998). Distributed Generation (DG) is the term used to describe these small scale energy sources that are typically located near their loads.

A number of solutions have been proposed for coordinating these disparate generation sources into small scale coherent blocks of power production and consumption. The Microgrid concept, as depicted in Fig. 2.2, assumes a cluster of loads and microsources operating as a single controllable system that provides both power and heat to its local area (Lasseter, 2002). Distributed Storage (DS) devices are also used in microgrid applications where the generation and load capacities of the microgrid cannot be met (Kroposki et al., 2008). These forms of storage can include supercapacitors, batteries and flywheels. These storage devices require power electronic devices in order to convert their stored DC energy to AC energy. These devices will usually be bidirectional to also allow the charging of the devices' power. DS devices can also be used to improve power quality of the microgrid area and allow fluctuations from sources such as wind and solar devices to be smoothed out. One of the main features of Microgrids is their ability to connect and disconnect from the main grid. In the grid connected mode of operation, the microgrid

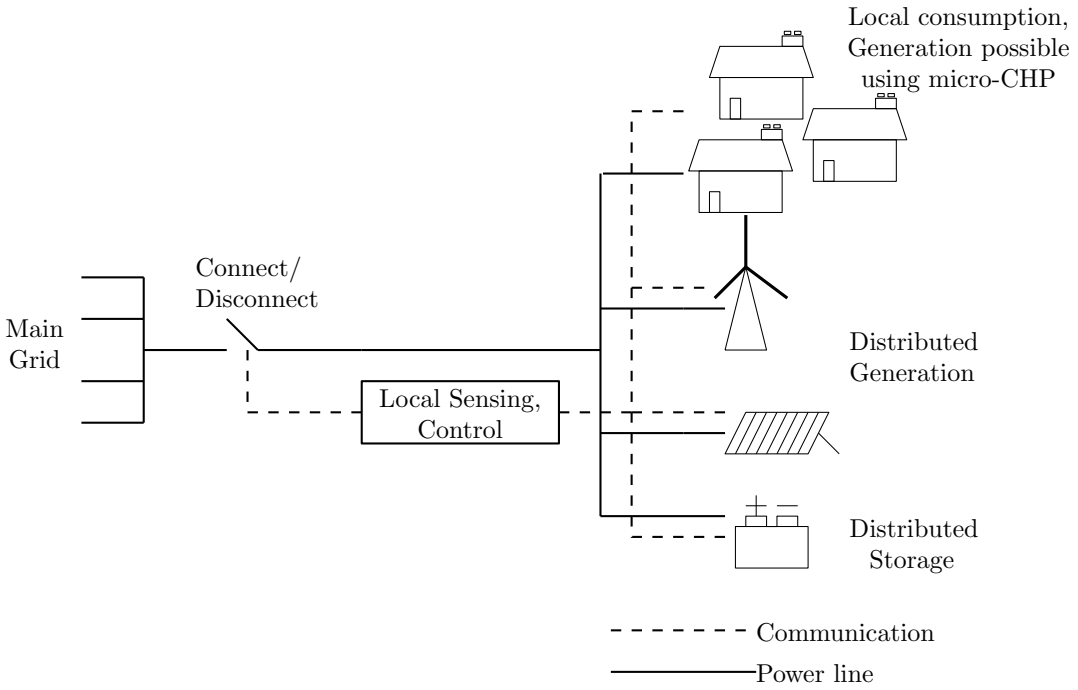


Figure 2.2: Microgrid which can disconnect from the main grid during instances of grid instability.

can both send and receive power to/from the main grid. When disconnected from the grid, in what is called the 'islanded' mode of operation, the microgrid acts independently from the rest of the grid, using power from its own generation sources and from storage devices to supply the demand within the local power grid. Typically, the microgrid would disconnect from the main grid in times of instability, reconnecting when the main grid has been restabilised. Special power electronic switches have been developed for this connection and disconnection in order to coordinate the switching in and out of the microgrid to the main power supply grid (Kroposki et al., 2008). Concepts such as the Virtual Power Plant (VPP) (Pudjianto et al., 2007) and energy hubs (Camacho et al., 2011) coordinate groups of generation sources and loads, and can be used to coordinate the distribution of power such that preference is given to the use of renewable sources over non-renewable sources of energy. VPPs can also be used to facilitate trading in energy markets, and can give consumers the option of disconnecting from the grid at peak times or times when the grid is near instability, allowing them to save money and conserve electricity usage.

Looking at the aforementioned developments, power grids can be seen to be evolving on a number of fronts. The new grid could be expected to look like the grid presented in Fig. 2.3. Where generation and control were once centralised and highly regulated, they are

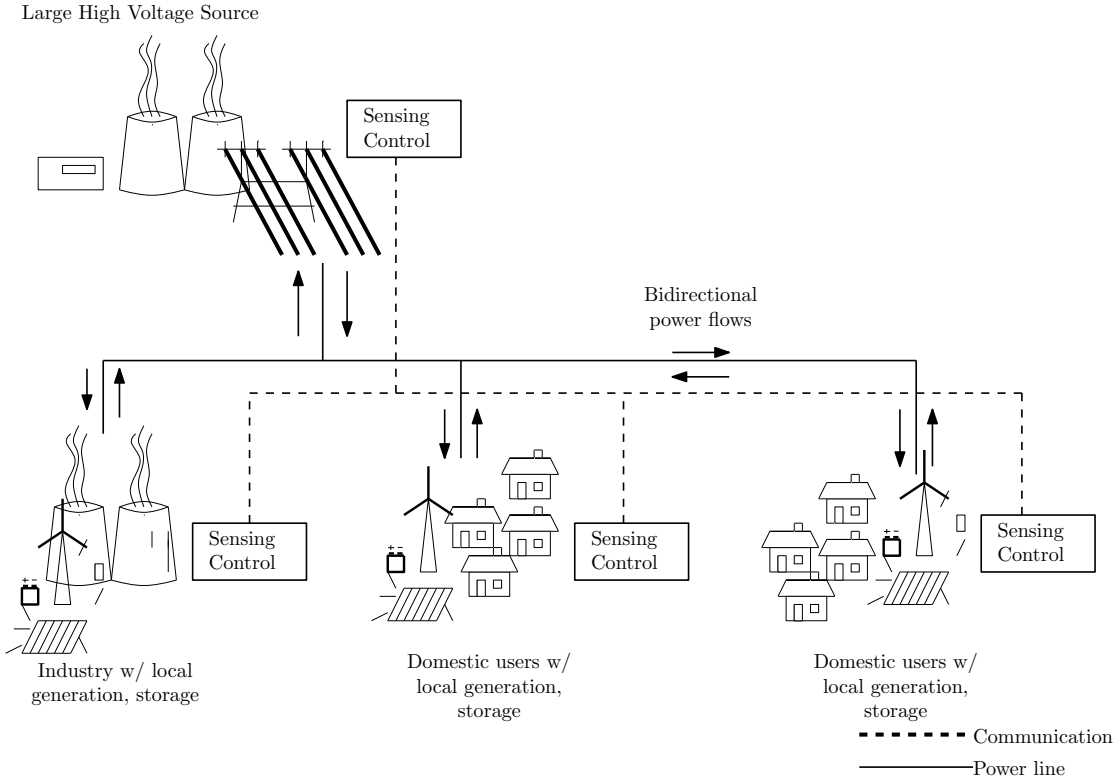


Figure 2.3: Future of power distribution.

now decentralised and deregulated. Where typically, in the past, generation sources would have been in few locations and close to their loads, now they are numerous and spread over vast geographical areas. Where the source and direction of power flows would have previously have been predictable, now they are becoming increasingly unpredictable. A new grid infrastructure is developing to enable the transition to this new power system. The Smart Grid is the umbrella term given to the new grid which is currently evolving. Unlike the traditional grid, which was typically passive, and 'dumb' in terms of only having a limited amount of real time system control, the Smart Grid enables far more flexilibity in terms of where and what power production can be supplied to the grid. It also enables the grid to automatically overcome serious contingencies, and allows consumers to become more active in terms of how they interact with the grid.

The installation of improved sensing and communications equipment, embedded generation, and storage devices will encourage the development of the current power grid towards this new system. As regards controlling these grids, two things will be necessary: First, an improvement in the ability to control the physical power system parameters themselves, i.e., the voltages, currents, powers, etc., will be needed. This is being enabled by the

installation and continuing development of power electronics devices. Second, the use of advanced control techniques that enable non-centralised control of the grid, such that areas of the grid can communicate with each other to coordinate the control of the overall grid, while maintaining their own autonomous decision making capabilities, will be necessary. The next two sections will discuss power electronics devices and some of the issues involved with the non-centralised control of smart grids.

2.2 Power Electronics Devices

While power electronics devices have a widespread range of applications, it is Flexible AC Transmission Systems (FACTS) devices which are of interest in this thesis. The original FACTS technologies were based on Thyristor valves and converters (Kundur, 1994). In recent years FACTS devices now utilise the more advanced technology of voltage source converters which are based on Insulated Gate Bipolar Transistors (IGBT) or Insulated Gate Commutated Transistors (IGCT). FACTS devices are said to be dynamic, to describe their fast controllability, and static, which means they have no moving parts like mechanical switches to perform dynamic controllability. They achieve advanced power system control through switched or controlled shunt compensation, series compensation and phase shift control. These devices facilitate very fast current, voltage or impedance control (Zhang et al., 2006). FACTS devices have been applied in the following areas:

- Power flow control (Zhihui et al., 2010),
- Transmission capacity enhancement (Ying et al., 2003),
- Voltage control (Rao et al., 2000),
- Reactive power compensation (Dizdarevic and Majstrovic, 2003),
- Stability improvement (Haque, 2005),
- Power quality improvement (Grunbaum, 2008),
- Power conditioning (Singh et al., 2011),
- Flicker mitigation (Zhang et al., 2004),

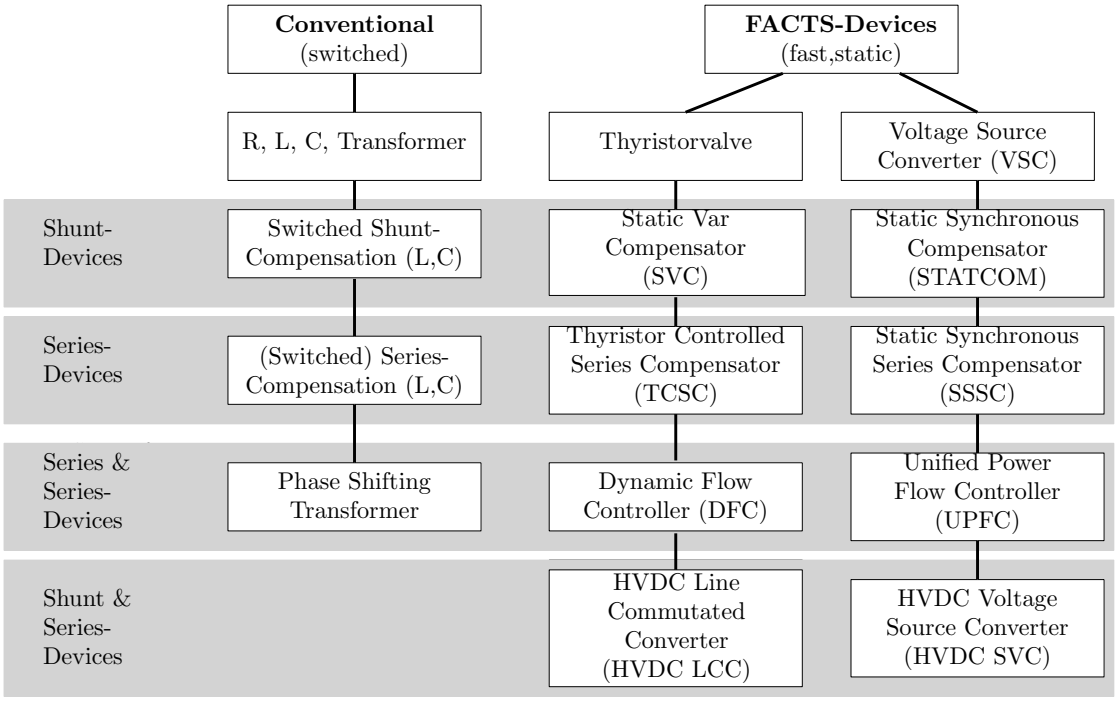


Figure 2.4: Overview of FACTS devices (Zhang et al., 2006).

- Interconnection of renewable and distributed generation and storages (Strunz and Brock, 2003).

In Fig. 2.4, the leftmost column shows conventional devices made from fixed or mechanically switched components such as resistors, capacitors, and conductors used with transformers. The left column of FACTS devices shows those devices constructed using the original thyristor valve technology. The right column of FACTS devices shows those devices made using the newer more advanced IGBT and IGCT technologies. A brief discussion of each of these technologies will now be given.

2.2.1 FACTS Devices

Shunt Devices

Static Var Compensators (SVC) and Static Compensators (STATCOM) operate as reactive power compensators and so have become vital in modern power systems for maintain-

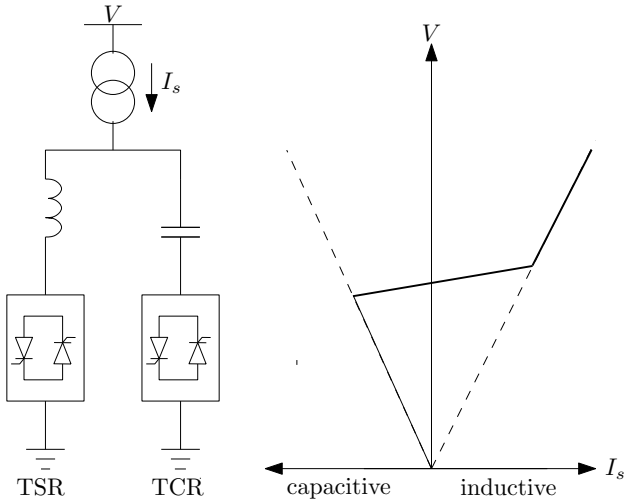


Figure 2.5: Static Var Compensator setup and its voltage/current characteristic.

ing high quality power. They have also become an integral component in dynamic reactive power compensation, which can help minimise the impact of random voltage variations on power systems. This is quite useful in industrial situations where a good quality voltage supply is needed (Chen et al., 2001), and is proving incredibly useful in offshore wind farms, where they are used to provide a balanced reactive power level to keep voltages within limits across the wind farm and at the point of interconnection to the grid, despite power production fluctuations (Wang and Hsiung, 2011).

SVCs consist of shunt connected variable capacitances and inductances called Thyristor Switched Capacitors (TSC) and Thyristor Controlled Reactors (TCR), respectively. When these devices are connected in parallel with each other they form a Static Var System (SVS), which can be seen in Fig 2.5 (Kundur, 1994). The variation of the thyristor angle allows the effective capacitance or inductance in each branch to be varied. The coordinated control of branches of these components is what enables reactive power control. An illustrative example of how this system operates to regulate voltage is given in Fig. 2.6. The inductor voltage-current characteristic slope gets steeper as the inductance increases. For the purposes of illustration, the capacitance is kept constant. Between the maximum and minimum ranges of the inductance the combination of the inductance and capacitance together produces a fixed voltage V independent of the shunt current I_s. In a real system capacitances can also be varied with a TSC and so using a number of parallel shunt devices the desired system regulation can be achieved. An SVS is capable of providing the reactive power required to control dynamic voltage oscillations under various system

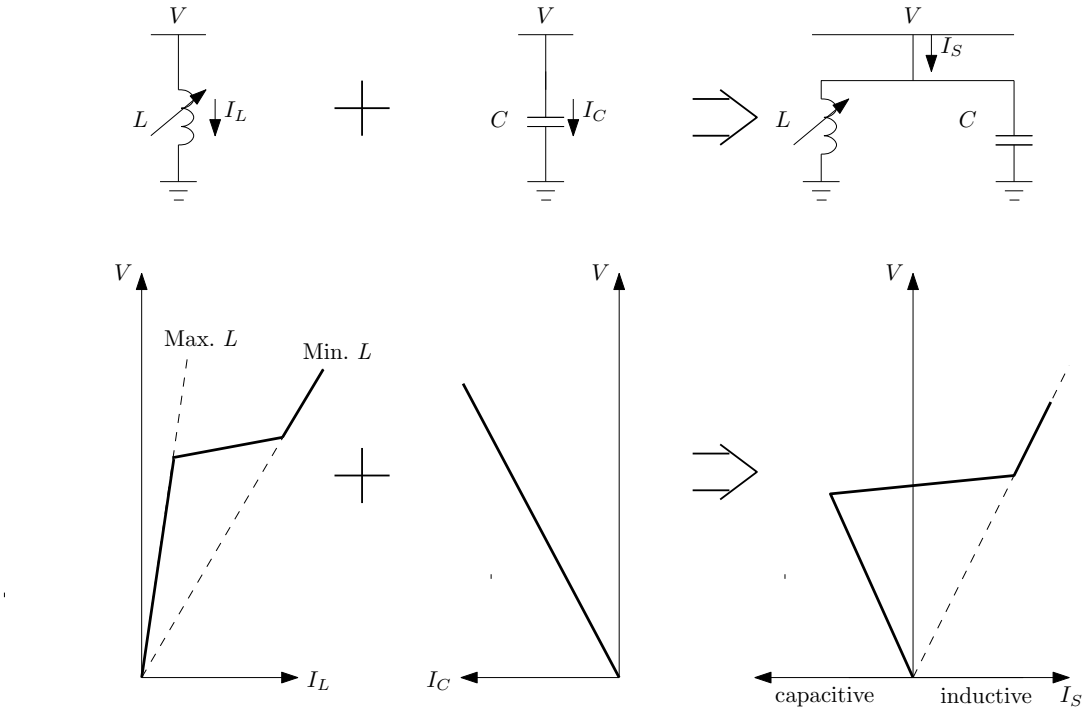


Figure 2.6: Illustration of voltage regulation using a shunt variable inductor and fixed capacitor connected in parallel (Kundur, 1994).

conditions thus enabling an improvement in power system transmission and distribution stability.

STATCOMs were first introduced in 1999 and are based on VSC technology, which allows the voltages to be controlled at a greater frequency than is the case with an SVC and so allow for faster reaction times. In STATCOMs a compensating voltage is connected in shunt with the transmission line through a tie reactance, as in Fig. 2.7 (Mohan, 2006; Sen and Sen, 2009). The shunt current I_s flowing through the inductor can be controlled so as to absorb or provide reactive power to the line. STATCOMs are treated as controllable voltage sources. Another advantage of STATCOMs over VSCs is that the controllable capacitive or inductive currents can be provided independently of the voltage V, and it can provide independent active and reactive power control. Its V-I_s characteristic can be seen in Fig. 2.7 where the vertical lines represent the current rating of the device. These vertical lines imply that the maximum allowable currents are independent of the voltage V and so means that even during the most severe contingencies the STATCOM will keep its full capacity (Zhang et al., 2006). While STATCOMs are currently more expensive to install than SVCs, limiting their use to scenarios that demand very fast voltage regulation,

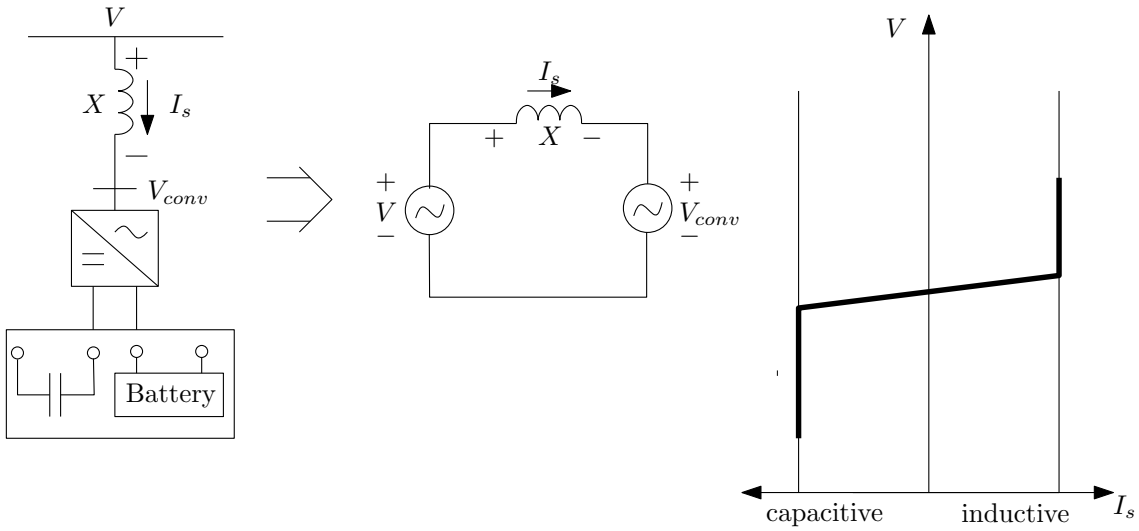


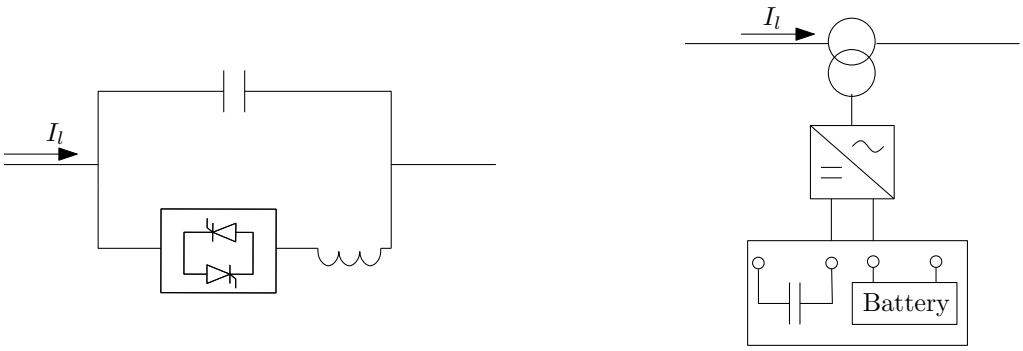
Figure 2.7: Static Compensator setup and its voltage/current characteristic (Mohan, 2006).

it is predicted that in time as the price of the semiconductors used in STATCOM reduces that STATCOMs will become the preferred device for voltage regulation (Noroozian et al., 2003).

Series Devices

Series devices can help overcome series voltage decline in magnitude and phase, and reduce voltage fluctuations. Series FACTS devices also have the capability to limit fault currents on the lines to which they are connected.

Thyristor Controlled Series Capacitors (TCSC) have two main uses. First, they can be used to increase the damping between large interconnected electrical systems by controlling the line reactance between two areas. Second, they can be used to overcome Sub-Synchronous Resonance (SSR); a problem that can arise due to interactions between large thermal generating units and series compensated transmission systems. TCSCs operate by varying a capacitive or inductive reactance, using the thyristor to vary this reactance, as can be seen in Fig. 2.8 for the case of a variable inductance (Mohan, 2006). The variance of this reactance enables electromechanical damping between large electrical systems. They can also change their apparent impedance (as seen by the line current) for



TCSC SSSC

Figure 2.8: Thyristor Controlled Series Compensator (Mohan, 2006) and Static Synchronous Series Compensator setup (Zhang et al., 2006).

sub-synchronous frequencies, in order to avoid sub-synchronous resonance. Both of these goals can be achieved simultaneously with TCSCs using appropriate control algorithms (Zhang et al., 2006).

While TCSCs are modelled as series impedances, Static Synchronous Series Compensators (SSSC) are series voltage sources. SSSCs are configured in a similar fashion to STATCOMs, as can be seen in Fig. 2.8. However they work in this case by injecting an almost sinusoidal compensating voltage, of variable magnitude, in series with a transmission line. This has the effect of emulating a series inductance or capacitance in series with the transmission line (Sen and Sen, 2009). While these devices are more expensive than TCSCs and cannot be used at the transmission level, they are primarily a device for power quality applications, with demonstrated potential in maintaining stability in deregulated power systems (Menniti et al., 2004). In this scenario they are called Dynamic Voltage Restorers (DVR). They can be used to keep voltages constant, for a factory infeed for example and can be used to mitigate voltage dips and flicker. With a charging mechanism or DC side battery they can work as uninterruptable power supplies (Zhang et al., 2006).

Shunt and Series Devices

As restrictions grow for new power lines, and power flows become more volatile due to energy market activities, increasing control of power flow is becoming more important.

Dynamic Flow Controllers (DFC) and Unified Power Flow Controllers (UPFC) enable increased control over power flows via shunt and series devices.

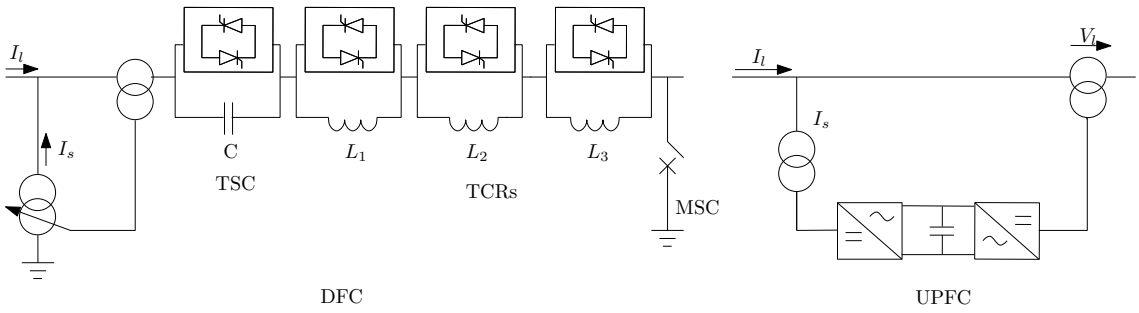


Figure 2.9: Dynamic Flow Controller and Unified Power Flow Controller setup (Zhang et al., 2006).

A DFC is a hybrid device, which combines the capabilities of a Phase Shifting Transformer (PST) and a switched series compensator, and can be seen in Fig. 2.9. It can provide series and shunt compensation and can actively adjust its internal parameters in order to control active and reactive power flow while regulating voltage, thus enabling better use of existing generation capacities in power networks (Ahmadi et al., 2008). A number of series connected TSCs and TCRs allow the control of the line reactance with the mechanically switched shunt capacitor (MSC) utilised to provide support in the case of overloads and other conditions. The Phase Shifting transformer (PST), which has a tap changer, allows a voltage to be injected into the line in quadrature with the node voltage and acts as a controllable reactance. The manipulation of each of the different elements of the DFC allows for the control of the active and reactive power flowing in the line to which it is attached in series (Zhang et al., 2006).

A UPFC, as seen in Fig. 2.9, is a combination of a static compensator and static series compensator. It consists of a shunt and series transformer, connected back-to-back via two VSCs with a common DC link capacitor. The DC capacitor in combination with the back-to-back VSCs allows independent bidirectional active power and reactive power exchange between the transformers allowing control of the phase shift of the series voltage. This allows full controllability for the voltage and power flow in the line.

High Voltage DC Transmission Devices

High Voltage Direct Current (HVDC) lines can be used to transfer large quantities of power between subsystems in power networks. They operate by converting AC power to DC power at one side of the line and reconverting the DC power back to AC power on the other side. It is widely recognised as having an advantage over AC transmission for long-distance bulk power delivery, asynchronous power connections, e.g., transporting power between an area with a 50Hz frequency and an area with 60 Hz, and long submarine cable crossings (Bahrman and Johnson, 2007). HVDC lines also have the ability to rapidly control the transmitted power between areas, and so have a significant impact on their associated AC power systems (Kundur, 1994).

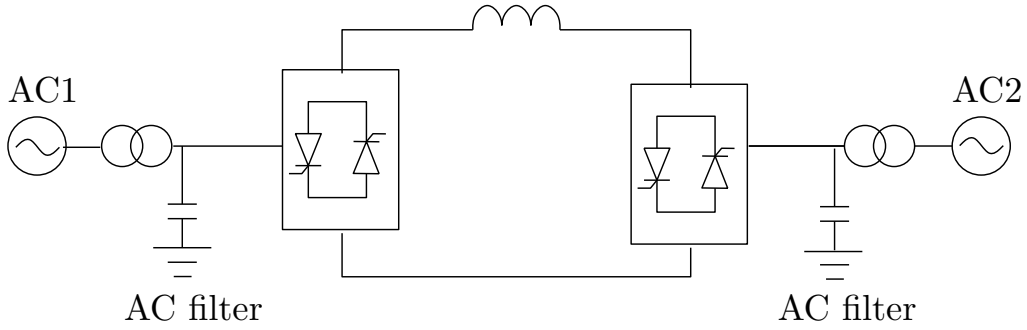


Figure 2.10: HVDC LCC system (Mohan, 2006).

The original Line Commutated Converter HVDC lines (HVDC LCC) use thyristor valves for the conversion process. The AC voltage is stepped up to a higher voltage using a transformer and the thyristors then convert the AC power to DC power at one side of the link at what is called the rectifier end. The thyristor bridge at the other end of the line, which is called the inverter, then converts the DC power back to AC power. The thyristors can only operate with the AC current lagging the voltage and so the conversion process requires the supply of reactive power (Zhang et al., 2006). As a result, the reactive power is not controllable in these lines. The reactive power is supplied by AC filters and any surplus reactive power must be reabsorbed by the grid. The weaker the AC system or the further the converter is away from a generation source, the tighter the reactive power exchange must be to stay within the desired voltage tolerances. This can be a limiting factor on the installation of HVDC LCC in certain areas (Bahrman and Johnson, 2007).

HVDC transmission using VSCs with Pulse Width Modulation (HVDC VSC) was intro-

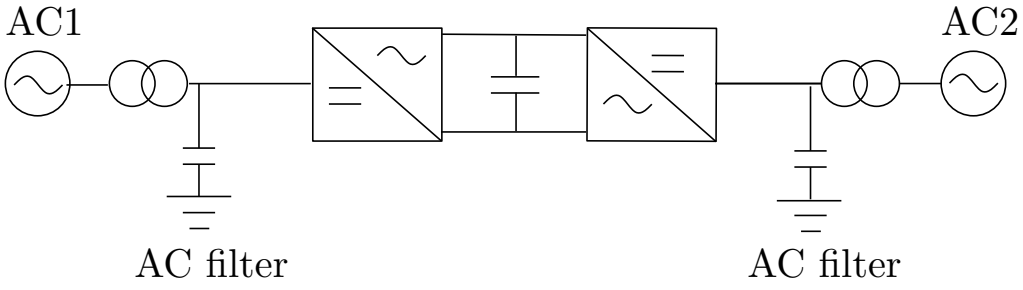


Figure 2.11: HVDC VSC system (Mohan, 2006).

duced in the 1990s under the name HVDC Light (Zhang et al., 2006). These are based on IGBT technology and as a result can be used to control both active and reactive power independently. The conversion process for HVDC VSC works on the same AC-DC-AC conversion as with the LCC system. It should be noted that while the active power entering the rectifier will roughly equal that leaving the inverter, this does not have to be the case with the reactive power. This significantly increases the overall system controllability and allows the HVDC lines to be used in order to increase the transmission capability of surrounding transmission lines in addition to balancing power flows. Also HVDC VSC, unlike HVDC LCC, does not require a source of reactive power and so overcomes many of the shortfalls of HVDC LCC by being capable of operating in almost any part of the grid, and can even start from a black start, i.e., the converter can be used to synthesize a balanced set of three phase voltages like a virtual synchronous generator (Bahrman and Johnson, 2007). This makes HVDC VSC particularly useful when connecting to weak areas of the grid which may require extra reactive power support.

Having looked at the modern power electronics devices that will enable the control of future power grids, a discussion on the advanced control techniques that will be used for the control of the grid will now be undertaken.

2.3 Non-centralised Control of Smart Grids

Classical control techniques were typically developed for centralised or decentralised control situations. In modern computer science when centralised control is used it is referred to as single-agent control. For clarity the definition of an agent, as understood in this thesis, will now be provided. An agent is defined here as an entity responsible for the

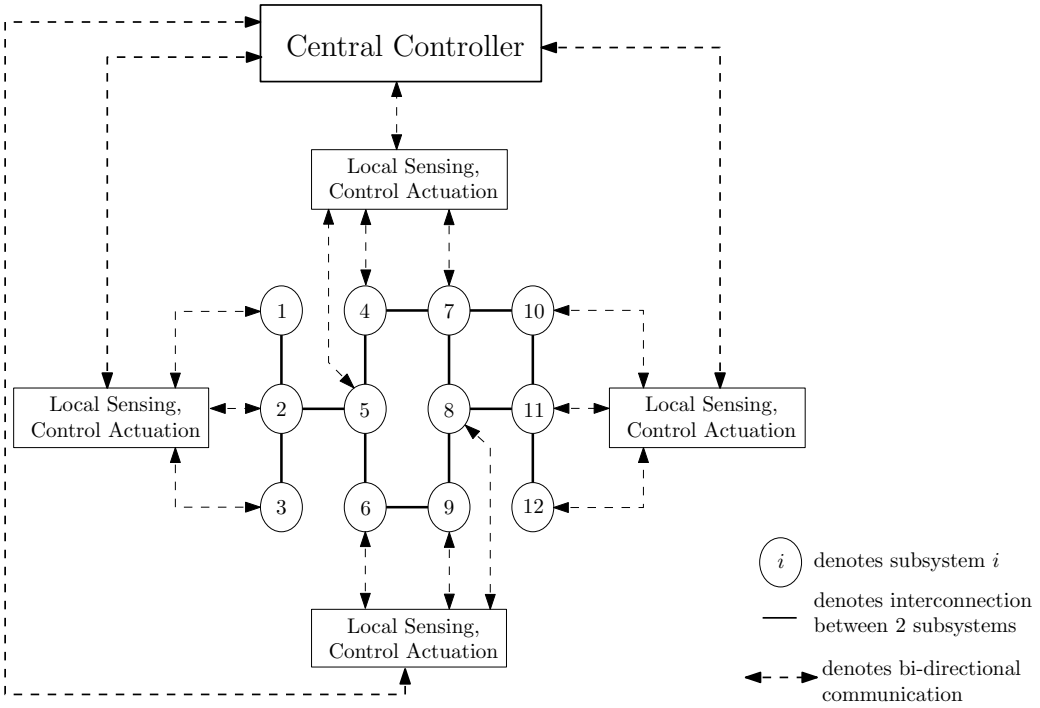


Figure 2.12: Centralised control of power system consisting of 12 subsystems with various interconnections.

control of a system or subsystem, with access to the current state of the system or subsystem it controls. The agent's local states are accessed by direct measurement or estimation. Agents have access to a model of the local system or subsystem and in the distributed case, agents are able to communicate with other agents who share a common variable. Agents compute values for their control inputs at discrete time steps based on the information available to them.

In centralised control, as shown in Fig. 2.12, the control agent has access to all information relevant to the control of each subsystem in the overall network, i.e., the states, goals, constraints, etc. Information is exchanged between the central controller and local sensors and actuators for each of the subsystems. Typically, centralised control systems give the best performance attainable by a control system. However a number of issues arise with the use of centralised control systems:

- They may exhibit poor robustness qualities and, in cases where the location of the control agent is a large distance from the object being controlled, issues regarding communication delays can arise.

- Centralised control systems can scale badly, i.e., as the size of the system increases, a centralised problem may become too large to solve in real time. This is often the case in large scale power systems.
- It may not always be possible to control an area using only one agent. When power networks are deregulated or when power networks cross country borders, there can be more than one control agent responsible for different parts of the grid.

In order to overcome the disadvantages inherent in the use of a single control agent, several control agents can be used to control different parts of the system. Where multiple control agents are used, the control system is typically referred to as a multi-agent control system. In principle these control systems can (Negenborn, 2007):

- Improve the robustness and reliability of systems. If one controller fails, the other controllers continue to maintain responsibility for the control of the remaining parts of the grid.
- Enable the scalability of control problems by dividing the original centralised problem into a number of smaller problems.
- Reduce communication delays. Agents can therefore operate near the area they are controlling and hence have faster access to both sensors and actuators.
- Overcome the problems inherent in systems where more than one control agent is needed in a system, as is the case in a deregulated power market.
- Allow individual agents autonomy over their control goals while ensuring overall system stability. This capability is vitally important in the smart grid, where consumers will be encouraged to take more active decisions as to how they consume and produce power.

If interactions between control areas are weak, it is presumed that a decentralised approach, where agents do not take the effect of interactions with other subsystems into account, is sufficient to control the overall system (Venkat, 2006). In these situations, agents are designated to control individual subsystems in a system without communicating with each other, under the assumption that the effect of feedback is sufficient to overcome the effect of interactions between subnetworks.

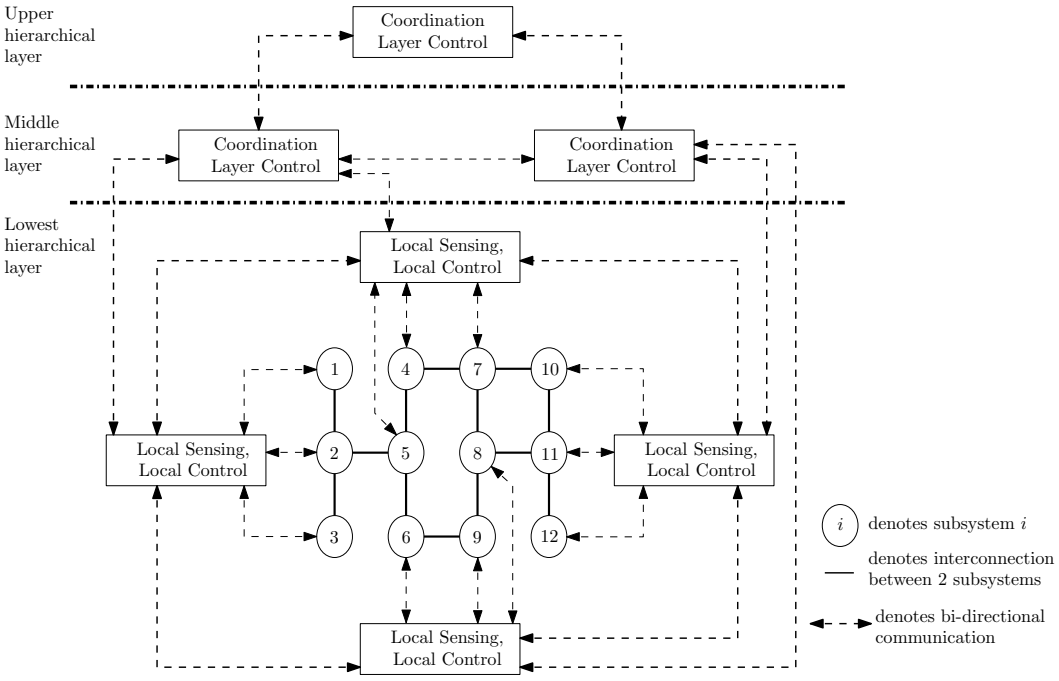


Figure 2.13: Distributed control architecture with coordination layers for hierarchical control.

However, the necessity for coordination between controllers in power networks was highlighted most dramatically in the North American blackout on August 14, 2003 . A decentralised control structure prevented the interconnected control areas from taking emergency actions such as selective load shedding. This had a knock-on effect across the network with successive subnetworks overloading, ultimately leading to a blackout. The U.S.-Canada Power System Outage Task Force reported that the extent of the system failure was so dramatic that within 7 minutes the blackout had spread from the Cleveland Akron area in northern Ohio to much of northeastern USA and Canada (U.S.-Canada Power System Outage Task Force, 2004).

Alternatives to a completely decentralised control structure include hierarchical and distributed control systems. In hierarchical systems there are a number of layers of controller agents which pass down setpoints to agents at lower layers, as can be seen in Fig. 2.13 (In this example there are 3 layers; an upper, middle, and lower layer, are given but depending on the hierarchical system there can be as many layers as needed). Typically lower layers deal with real time control of the system and the higher layers are used to calculated longer term goals, and will usually determine the setpoints for the lower level control layers to follow. Distributed control systems use communication between agents, that are usually

in the same hierarchical control layer, in order to coordinate their responses, as can be seen in Fig. 2.13. Distributed control techniques have combined aspects of control theory, optimisation, and game theory to enable stable multi-agent control. In recent years there has been many developments in this area, with distributed Model Predictive Control in particular showing much promise.

2.3.1 Model Predictive Control

Model Predictive Control (MPC) is an optimisation-based technique which uses a prediction model in order to determine the control inputs for a system. An attractive feature of MPC is the way in which system constraints can be catered for in the problem in a straightforward manner. It is also an intuitively attractive technique from which stable performance can be attained without extensive tuning (Rawlings and Mayne, 2009; Wang, 2009; Rossiter, 2003).

Given the attractive properties of MPC, it is no wonder that it is finding success in a wide number of applications. MPC has been utilised successfully in manufacturing and industry (Camacho and Bordons, 2003), transportation networks such as water (Negenborn et al., 2009; Ocampo-Martinez et al., 2010), urban traffic networks (Tettamanti et al., 2008), air traffic control (Chaloulos et al., 2010), and in medical applications (Wang et al., 2010). MPC is already proving incredibly useful as a solution for a wide range of Smart Grid control problems, such as balancing grid production and consumption (Trangbaek et al., 2011), coordinating the different elements of the grid at different time scales (Ulbig et al., 2011), coordination of small scale power sources (Camacho et al., 2011), minimizing power losses in an electrical network subject to voltage and power constraints (Lavaei et al., 2011), and predictive charging of large volumes of plug-in electric vehicles (Ma et al., 2011).

However, for large systems such as the electricity grid, it is often impractical to implement MPC from a central controller, due to computational constraints. Likewise, it is often necessary to use multi-agent techniques when using MPC with electricity grids, due to the issues associated with deregulation and the degree to which power subsystems are interconnected, as was outlined previously. These multi-agent systems consist of agents that communicate and cooperate with each other to approximate the behaviour of a centralised

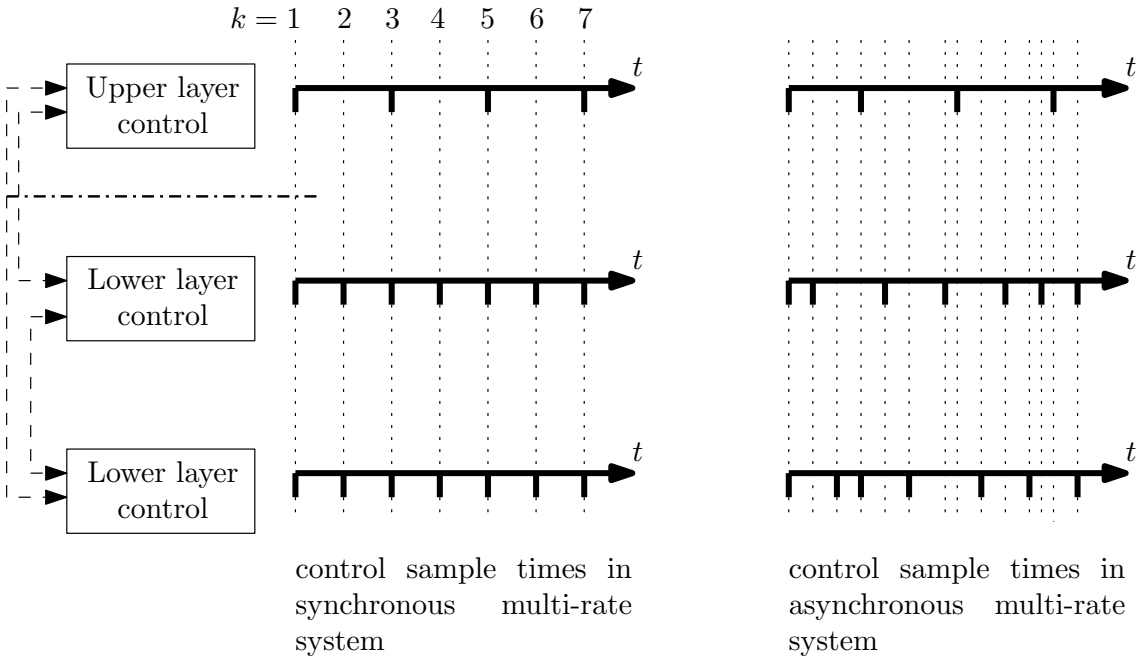


Figure 2.14: Hierarchical and distributed control architecture showing multi-rate synchronous and asynchronous updating.

MPC system. Agents may be responsible for the control of subsystems and the agent architecture can also be of a hierarchical nature with agents in higher layers coordinating the actions of lower layer agents (Scattolini, 2009). In hierarchical MPC an upper layer will usually determine setpoints or constraints that are sent to lower layer MPC agents (Falcone et al., 2008; Bendtsen et al., 2010). In (Negenborn, 2007), for example, higher layer objectives were used to consider the slow dynamics of the system and pass setpoints to the lower layer controllers which would try and achieve these setpoints while dealing with the faster system dynamics. These systems may also be multi-rate systems, with agents performing control actions at different times. These multi-rate systems typically fall into two categories; synchronous multi-rate systems, and asynchronous multi-rate systems, as can be seen in Fig. 2.14. In synchronous multi-rate systems some agents perform optimisations at fast sample rates and others perform optimisations over multiples of these fast sample intervals. In asynchronous systems, agents do not share a common sample time or sample period, as can be seen on the asynchronous update timeline in Fig. 2.14.

This thesis concentrates primarily on single-layer distributed MPC systems. There has been much research interest in recent years in this area (Scattolini, 2009; Sanchez et al., 2011; Liu et al., 2010). Distributed MPC algorithms are typically iterative or non-iterative.

Also their implementations are usually carried out in series or in parallel (although it may be possible to parallelise the actions of agents not directly connected in serial implementations). Of the non-iterative techniques, many Lyapunov-based distributed MPC techniques have been developed (Camponogara et al., 2002; Liu et al., 2010; Hermans et al., 2010), which use a Lyapunov constraint function to ensure system stability. Also in (Javalera et al., 2010) a non-iterative method is developed where a reinforcement learning based "negotiator" agent is used between distributed MPC agents to provide values of interconnecting variables to the agents which they then use in their MPC problems. There are many iterative distributed MPC methods that have been developed based on game-theoretic approaches that search for optimal equilibria (Li et al., 2005; Sanchez et al., 2011). Other decomposition-coordination methods decompose the original control problem into several smaller optimisation problems and use communication between agents to coordinate their solutions. Examples of decomposition methods used include Jacobian decomposition (Venkat, 2006), Gauss-Seidel decomposition (Negenborn et al., 2008), and Bender's decomposition (Moroş Andan et al., 2010). The above methods are all synchronous methods where each agent optimises at each time step. However, asynchronous methods have been developed in (Camponogara and Talukdar, 2007; Venkat, 2006) and multi-rate methods have been developed in (Heidarinejad et al., 2011; Venkat, 2006; Roshany-Yamchi et al., 2011). Robust forms of distributed MPC have also been developed in (Trodden and Richards, 2006; Al-Gherwi et al., 2011). In all of these techniques, there is a general performance trade off between the level of control performance achievable and the level of communication needed between agents at each control cycle. In general, better control performance can be achieved with increasing levels of communication.

2.3.2 Game Theoretic Perspectives on Centralised and Non-centralised MPC Control Goals

The goal of a centralised MPC scheme for a system of n subsystems can be stated as follows:

$$\tilde{\boldsymbol{u}}(k) = \arg\min_{\tilde{\boldsymbol{u}}} \sum_{a=1}^{n} w_a J_a^{\text{local}}(\boldsymbol{x}_a(k), \tilde{\boldsymbol{u}}_a(k)), \tag{2.1}$$

where $\tilde{\boldsymbol{u}}_a(k)$ are the control inputs to subsystem a over the prediction horizon at sample k, $\boldsymbol{x}_a(k)$ is the state of subsystem a at sample k, $\tilde{\boldsymbol{u}}(k)$=$[\tilde{\boldsymbol{u}}_1(k), \dots \tilde{\boldsymbol{u}}_n(k)]$ is the vector of all predicted inputs, and $J_a^{\text{local}}(\boldsymbol{x}_a(k), \tilde{\boldsymbol{u}}_a(k))$ is the local cost function of the a^{th} subsys-

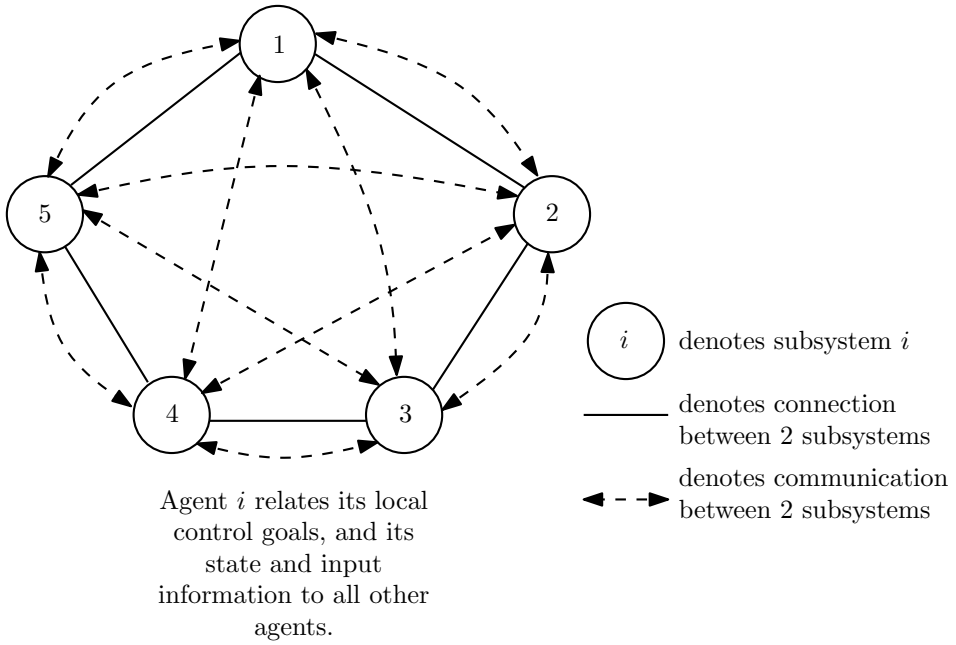


Figure 2.15: Communications necessary in a system composed of 5 subsystem in order to achieve a Pareto equilibrium performance.

tem. The performance of a centralised MPC scheme reaches what is known as a Pareto equilibrium. It is possible to achieve this performance in a distributed manner by using an iterative procedure (Venkat, 2006; Sanchez et al., 2011). Here, each agent optimises the following cost function in parallel at each iteration:

$$\tilde{\boldsymbol{u}}_a(k,l) = \arg\min_{\tilde{\boldsymbol{u}}_a} \sum_{a=1}^{n} w_a J_a^{\text{local}}(\boldsymbol{x}_a(k), \tilde{\boldsymbol{u}}_1(k,l-1), \ldots, \tilde{\boldsymbol{u}}_a(k,l), \ldots, \tilde{\boldsymbol{u}}_n(k,l-1)), \quad (2.2)$$

where $\tilde{\boldsymbol{u}}_a(k,l)$ is the vector of inputs to subsystem a at sample time k and iteration l of the iterative process. The algorithm can terminate either after a pre-determined number of iterations or when the optimised inputs have converged. Each agent optimises for its own input, based on the previous value of inputs which have been calculated by all the other agents. In game theory, a Pareto equilibrium is the best control performance that can be achieved by a system. This control can only be achieved, however, if all agents in a system have all the information of a centralised MPC made available to them (assuming convex cost functions), i.e., the goals of all agents in a system, knowledge of the full state-space of the system, all system constraints, etc. as can be seen in Fig. 2.15 for a system consisting of 5 subsystems. However, in vast complex systems, such as electricity grids, the communication of this volume of information is impractical or may not even be possible, e.g., in a deregulated power market control agents may not be willing to share this level of information with other control agents.

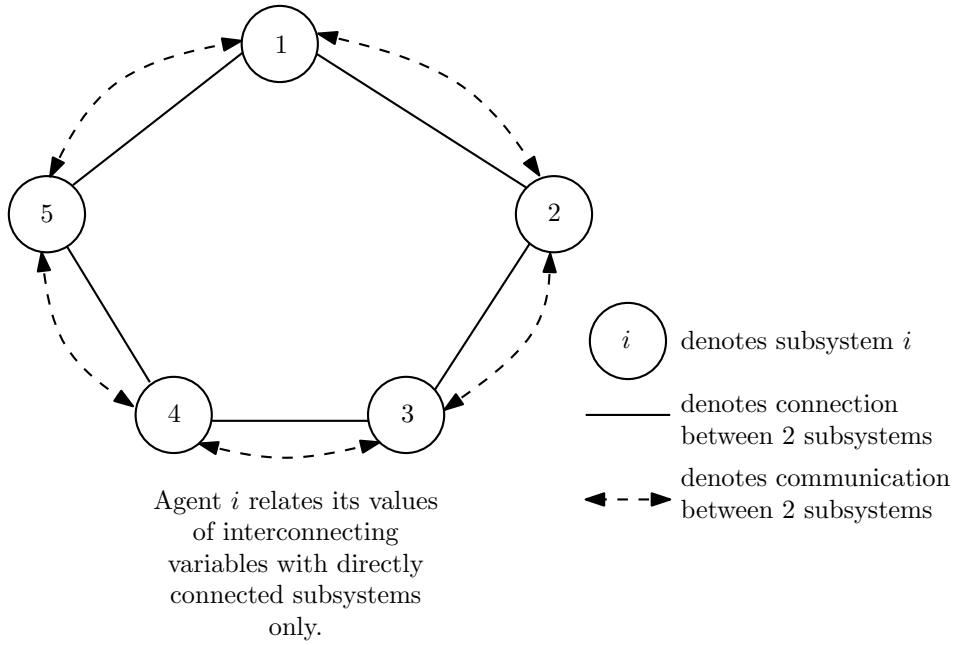


Figure 2.16: Communications necessary in a system composed of 5 subsystem in order to achieve a Nash equilibrium performance.

However, when communication of interconnecting variables is allowed between adjacent agents, a Nash equilibrium can be achieved (Sanchez et al., 2011; Venkat, 2006). These algorithms can be both iterative and non-iterative (Scattolini, 2009). Nash equilibria are typically achieved using cost functions similar to the following:

$$\tilde{\boldsymbol{u}}_a(k,l) = \arg\min_{\tilde{\boldsymbol{u}}_a(k,l)} J_a^{\text{local}}(\boldsymbol{x}_a(k), \tilde{\boldsymbol{u}}_a(k,l), \tilde{\boldsymbol{v}}_a(k,l)) + J_a^{\text{inter}}(\boldsymbol{x}_a(k), \tilde{\boldsymbol{u}}_a(k,l), \tilde{\boldsymbol{v}}_a(k,l)), \quad (2.3)$$

where $\tilde{\boldsymbol{v}}_a(k,l)$ is a vector of interconnecting inputs over the prediction horizon coming from other subsystems affecting subsystem a at sample k and iteration l (in the case of iterative algorithms), and $J_a^{\text{inter}}(\boldsymbol{x}_a(k), \tilde{\boldsymbol{u}}_a(k,l), \tilde{\boldsymbol{v}}_a(k,l))$ is an interconnection cost that arises due to agent a's interactions with other subnetworks. Usually $J_a^{\text{inter}}(\boldsymbol{x}_a(k), \tilde{\boldsymbol{u}}_a(k,l), \tilde{\boldsymbol{v}}_a(k,l))$ is used in order to form consensus with adjacent agents on values of the interconnecting variables. As the achievement of a Nash equilibrium only requires the communication of the interconnecting variables shared by connected subsystems (as can be seen in Fig. 2.16 for the 5 area system used previously in Fig. 2.15), it means that significantly less inter-agent communication is needed to achieve systemwide control when compared to the Pareto seeking system. While Nash equilibrium seeking control systems in general do not achieve the performance of a Pareto equilibrium seeking control system, their performance will usually be significantly better than that achieved by a decentralised control system with no communications (Venkat, 2006). Indeed, promising results have already been

achieved controlling power networks, which include FACTS devices, with control agents that only communicate with agents to which they are connected by a common variable (Negenborn et al., 2010; Venkat, 2006; Talukdar et al., 2005).

2.4 Summary

This chapter examined how power grids have been evolving in recent years. Originally power systems were centralised monolithic structures where a few large generation centres provided energy to consumers. Power typically flowed in one direction which was from the large scale production stations to the end consumers. This predictablility as regards the direction of power flows allowed for simpler control systems to provide effective control of power systems. However, the increasing penetration of distributed generation sources and the decentralisation of both power generation and control has meant that more sophisticated control techniques are needed to control future power grids. It has been seen that local communications based distributed MPC techniques and FACTS devices provide a means of enabling the transition to the new Smart Grid. However many issues still remain to be addressed with these techniques and large-scale real-time deployment has yet to be carried out using this combination of techniques.

References

S. Abraham and R.J. Efford. Final report on the august 14, 2003 blackout in the united states and canada: Causes and recommendations. Technical report, U.S.-Canada Power System Outage Task Force, April 2004.

R. Ahmadi, A. Sheykholeslami, A.N. Niaki, and H. Ghaffari. Power flow control and solutions with Dynamic Flow Controller. In *IEEE Canada Electric Power Conference*, pages 1–6, Vancouver, BC, Canada, Oct. 2008.

W. Al-Gherwi, H. Budman, and A. Elkamel. A robust distributed model predictive control algorithm. volume 21, pages 1127 – 1137, 2011.

M.P. Bahrman and B.K. Johnson. The ABCs of HVDC transmission technologies. *IEEE Power and Energy Magazine*, 5(2):32–44, March-April 2007.

J. Bendtsen, K. Trangbaek, and J. Stoustrup. Hierarchical model predictive control for resource distribution. In *49th IEEE Conference on Decision and Control (CDC)*, pages 2468 –2473, Dec. 2010.

E. F. Camacho and C. Bordons. *Model Predictive Control, 2nd Edition*. Springer, London, UK, 2003.

E. F Camacho, A. J. Del Real, C. Bordons, and A. Arce. Solar Thermal Plants Integration in Smart Grids. In *Proceedings of the 18th IFAC World Congress*, volume 18, pages 4939–4944, Sept 2011.

E. Camponogara and S.N. Talukdar. Distributed Model Predictive Control: Synchronous and Asynchronous Computation. *IEEE Transactions on Systems, Man and Cybernetics*, 37(5):732 –745, Sept. 2007.

E. Camponogara, D. Jia, B.H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22(1):44–52, Feb 2002.

G. Chaloulos, P. Hokayem, and J. Lygeros. Distributed hierarchical MPC for conflict resolution in air traffic control. In *American Control Conference (ACC), 2010*, pages 3945 –3950, July 2010.

C.S. Chen, H.J. Chuang, C.T. Hsu, and S.M. Tseng. Mitigation of voltage fluctuation for an industrial customer with arc furnace. In *Power Engineering Society Summer Meeting*, volume 3, pages 1610 –1615, Vancouver, BC, Canada, July 2001.

N. Dizdarevic and M. Majstrovic. FACTS-based reactive power compensation of wind energy conversion system. 2:8 pp. Vol.2, June 2003.

R. Dufo-López and J. L. Bernal-Agustín. Design and control strategies of PV-diesel systems using genetic algorithms. *Solar Energy*, 79(1):33 – 46, 2005.

P. Falcone, F. Borrelli, H.E. Tseng, J. Asgari, and D. Hrovat. A hierarchical Model Predictive Control framework for autonomous ground vehicles. In *American Control Conference*, pages 3719 –3724, June 2008.

R. Grunbaum. FACTS for voltage control and power quality improvement in distribution grids. pages 1 –4, June 2008.

M.H. Haque. Stability improvement by FACTS devices: a comparison between STATCOM and SSSC. 2:1708 – 1713, June 2005.

M. Heidarinejad, J. Liu, D. M. de la Pena, J. F. Davis, and P. D. Christofides. Multirate distributed model predictive control of nonlinear systems. In *American Control Conference*, pages 5181 –5188, June 2011.

R.M. Hermans, M. Lazar, and A. Jokic. Almost decentralized lyapunov-based nonlinear model predictive control. In *American Control Conference*, pages 3932–3938, July 2010.

V. Javalera, B. Morcego, and V. Puig. Negotiation and Learning in distributed MPC of Large Scale Systems. In *American Control Conference*, Baltimore, Maryland, July 2010.

W.D. Kellogg, M.H. Nehrir, G. Venkataramanan, and V. Gerez. Generation unit sizing and cost analysis for stand-alone wind, photovoltaic, and hybrid wind/pv systems. *IEEE Transactions on Energy Conversion*, 13(1):70 –75, March 1998.

M. Korpas and A.T. Holen. Operation planning of hydrogen storage connected to wind power operating in a power market. *IEEE Transactions on Energy Conversion*, 21(3): 742 –749, Sept. 2006.

B. Kroposki, R. Lasseter, T. Ise, S. Morozumi, S. Papatlianassiou, and N. Hatziargyriou. Making microgrids work. *Power and Energy Magazine, IEEE*, 6(3):40–53, May-June 2008.

P. Kundur. *Power System Stability and Control*. Mc-Graw Hill, New York, 1994.

R.H. Lasseter. Microgrids. *IEEE Power Engineering Society Winter Meeting*, 1:305 – 308, 2002.

J. Lavaei, A. Rantzer, and S. Low. Power flow optimization using positive quadratic programming. In *Proceedings of the 18th IFAC World Congress*, Milan, Italy, August 2011.

S. Li, Y. Zhang, and Q. Zhu. Nash-optimization enhanced distributed model predictive control applied to the shell benchmark problem. *Information Sciences*, 170:329–349, 2005.

M. Liebreich, E. Zindler, T. Tringas, A. Gurung, and M. von Bismarck. Green investing 2011: Reducing the cost of financing. Technical report, World Economic Forum, April 2011.

J. Liu, X. Chen, D. Muñoz de la Peña, and P. D. Christofides. Sequential and iterative architectures for distributed model predictive control of nonlinear process systems. *AIChE Journal*, 56(8):2137–2149, 2010.

Z. Ma, I. A. Hiskens, and D. Callaway. A Decentralized MPC Strategy for Charging Large Populations of Plug-in Electric Vehicles. In *Proceedings of the 18th IFAC World Congress*, Milan, Italy, August 2011.

G. M. Masters. *Renewable and Efficient Electric Power Systems*. John Wiley and Sons, Inc, 2004.

D. Menniti, A. Pinnarelli, N. Scordino, and N. Sorrentino. Using a FACTS device controlled by a decentralised control law to damp the transient frequency deviation in a deregulated electric power system. *Electric Power Systems Research*, 72(3):289 – 298, 2004.

N. Mohan. *First Course on Power Systems*. MNPere, 2006.

P.-D. Moroş Andan, R. Bourdais, D. Dumur, and J. Buisson. Distributed model predictive control based on Benders' decomposition applied to multisource multizone building temperature regulation. In *49th IEEE Conference on Decision and Control*, pages 3914 –3919, Dec. 2010.

United Nations. Kyoto protocol to the United Nations framework convention on climate change. 1998.

R. R. Negenborn. *Multi-Agent Model Predictive Control with Application to Power Networks*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 2007.

R. R. Negenborn, B. De Schutter, and J. Hellendoorn. Multi-agent model predictive control for transportation networks: Serial versus parallel schemes. *Engineering Applications of Artificial Intelligence*, 21(3):353–366, April 2008.

R. R. Negenborn, R. R. Hug-Glanzmann, B. De Schutter, and G. Andersson. A novel coordination strategy for multi-agent control using overlapping subnetworks with application to power systems. In J. Mohammadpour and K. M. Grigoriadis, editors, *Efficient Modeling and Control of Large-Scale Systems*, pages 251–278. Springer, Norwell, Massachusetts, 2010.

R.R. Negenborn, P.-J. van Overloop, T. Keviczky, and B. De Schutter. Distributed model predictive control of irrigation canals. *Networks and Heterogeneous Media*, 4(2):359380, June 2009.

M. Noroozian, A.N. Petersson, B. Thorvaldson, B.A. Nilsson, and C.W. Taylor. Benefits of SVC and STATCOM for electric utility application. In *IEEE PES Transmission and Distribution Conference and Exposition*, volume 3, pages 1192 – 1199, Sept. 2003.

C. Ocampo-Martinez, V. Fambrini, D. Barcelli, and V. Puig. Model predictive control of drinking water networks: A hierarchical and decentralized approach. In *American Control Conference*, pages 3951 –3956, July 2010.

Michel De Paepea, Peter D'Herdta, and David Mertensb. Micro-CHP systems for residential applications. *Energy Conversion and Management*, 47(18-19):3435 – 3446, 2006.

D. Pudjianto, C. Ramsay, and G. Strbac. Virtual power plant and system integration of distributed energy resources. *Renewable Power Generation, IET*, 1(1):10 –16, March 2007.

P. Rao, M.L. Crow, and Z. Yang. STATCOM control for power system voltage control applications. *IEEE Transactions on Power Delivery*, 15(4):1311 –1317, Oct. 2000.

J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, Wisconsin, 2009.

S. Roshany-Yamchi, R.R. Negenborn, M. Cychowski, B. De Schutter, Connell J., and K. Delaney. Distributed model predictive control and estimation of large-scale multi-rate systems. In *Proceedings of the 18th IFAC World Congress*, pages 416–422, Milan, Italy, August 2011.

J.A. Rossiter. *Model Based Predictive Control-A Practical Approach*. CRC Press, Florida, 2003.

G. Sanchez, L. Giovanini, M. Murillo, and A. Limache. *Advanced Model Predictive Control*, chapter Distributed Model Predictive Control Based on Dynamic Games, pages 1–26. InTech, July 2011.

R. Scattolini. Architectures for distributed and hierarchical Model Predictive Control - A review. *Journal of Process Control*, 19(5):723–731, 2009.

K. Sen and M. Sen. *Introduction to FACTS Controllers: Theory, Models, and Applications*. Wiley-IEEE Press, 2009.

M. Singh, V. Khadkikar, A. Chandra, and R.K. Varma. Grid Interconnection of Renewable Energy Sources at the Distribution Level With Power-Quality Improvement Features. *IEEE Transactions on Power Delivery*, 26(1):307 –315, Jan. 2011.

K. Strunz and E.K. Brock. Hybrid plant of renewable stochastic source and multilevel storage for emission-free deterministic power generation. pages 214 – 218, Oct. 2003.

S. Talukdar, Dong Jia, P. Hines, and B.H. Krogh. Distributed Model Predictive Control for the Mitigation of Cascading Failures. pages 4440 – 4445, December 2005.

John Olav Giver Tande. Exploitation of wind-energy resources in proximity to weak electric grids. *Applied Energy*, 65(1-4):395 – 401, 2000.

T. Tettamanti, I. Varga, B. Kulcsar, and J. Bokor. Model predictive control in urban traffic network management. In *Control and Automation, 2008 16th Mediterranean Conference on*, pages 1538 –1543, June 2008.

Klaus Trangbaek, Jan Bendtsen, and Jakob Stoustrup. Hierarchical control for smart grids. In *Proceedings of the 18th IFAC World Congress*, Milan, Italy, August 2011.

P. Trodden and A. Richards. Robust distributed model predictive control using tubes. In *American Control Conference, 2006*, pages 2034–2039, June 2006.

Andreas Ulbig, Michle Arnold, Spyros Chatzivasileiadis, and Goran Andersson. Framework for multiple time-scale cascaded mpc application in power systems. In *Proceedings of the 18th IFAC World Congress*, Milan, Italy, August 2011.

A. Venkat. *Distributed Model Predictive Control: Theory and Applications*. PhD thesis, University of Wisconsin-Madison, Wisconsin, 2006.

Li Wang and Chia-Tien Hsiung. Dynamic Stability Improvement of an Integrated Grid-Connected Offshore Wind Farm and Marine-Current Farm Using a STATCOM. *IEEE Transactions on Power Systems*, 26(2):690–698, May 2011.

Liuping Wang. *Model Predictive Control System Design and Implementation Using Matlab*. Advances in Industrial Control. Springer, London, 2009.

Y Wang, E. Dassau, and F.J. Doyle. Closed-loop control of artificial pancreatic β -cell in type 1 diabetes mellitus using model predictive iterative learning control. *Biomedical Engineering, IEEE Transactions on*, 57(2):211 –219, Feb 2010.

X. Ying, Y.H. Song, C. C. Liu, and Y.Z. Sun. Available transfer capability enhancement using FACTS devices. *IEEE Transactions on Power Systems*, 18(1):305 – 312, Feb. 2003.

Li Zhang, Yilu Liu, M.R. Ingram, D.T. Bradshaw, S. Eckroad, and M.L. Crow. EAF voltage flicker mitigation by FACTS/ESS. 1:372 – 378, Oct. 2004.

Xiao-Ping Zhang, Christian Rehtanz, and Bikash Pal. *Flexible AC Transmission Systems: Modelling and Control*. Power Systems. Springer, London, 2006.

Y. Zhihui, S.W.H. de Haan, J.B. Ferreira, and D. Cvoric. A FACTS Device: Distributed Power-Flow Controller (DPFC). *IEEE Transactions on Power Electronics*, 25(10):2564 –2572, Oct. 2010.

Chapter 3

Model Predictive Control: Real-Time Optimisation Based Control

3.1 Model Predictive Control

Model Predictive Control (MPC) (Maciejowski, 2002; Rawlings and Mayne, 2009; Camacho and Bordons, 2003; Rossiter, 2003) is an optimal control technique, in which a control agent uses an internal process model to predict the process output over a certain number of sample steps (called the prediction horizon) to calculate optimal control moves for the system. One of the main advantages of this control technique is the systematic and intuitive manner in which constraints are incorporated into the control system and the fact that delays are naturally catered for. It enables the control of systems with multiple inputs and outputs in a straightforward manner. It can adapt to slow changes in system parameters and it is reasonably straightforward to tune MPC systems in order to achieve stable closed loop performance. In order to implement MPC, the practitioner must specify the control goals and supply a prediction model for the system. The measurement of the system state is also needed at each control sample. Another issue is that the MPC optimisation problem must be solvable within the necessary time frame in order to provide

the system inputs at each control sample. It is at this stage a mature technology, with feasibility, stability, and robustness proofs well established (Rawlings and Mayne, 2009).

3.1.1 The History of MPC

The foundations of modern MPC date back to the 1960s within the field of optimal control. Dynamic Programming provided sufficient conditions for optimality and a constructive procedure for determining an optimal feedback controller. The maximum principle provided the necessary conditions for optimality and encouraged researchers to develop computational algorithms capable of determining open-loop control laws for a given initial state (Mayne et al., 2000). Other research, noted in (Mayne et al., 2000), conducted during the 1970s and 80s, used finite control horizons for control (as opposed to infinite horizons which would have been used in H_∞ control but which were difficult to implement for real-time on-line control), and ensured stability of the system with the use of Lyapunov functions and a stability constraint $x(T) = 0$, where $x(T)$ is the final state in the prediction horizon. While these ideas lay the theoretical groundwork for MPC, the original MPC implementations were carried out in industry, particularly in the petro-chemical and process industries (Mayne et al., 2000; Camacho and Bordons, 2003). Existing control techniques, such as linear quadratic control, were not used due to their inability to handle constraints, nonlinearities, and uncertainty. The ability of MPC to deal with constraints was one of its key strengths, as due to economic considerations, operating points in plants are often situated on the boundary of the set of operating points satisfying all constraints (Mayne et al., 2000).

Two of the earliest versions of MPC were Model Predictive Heuristic Control (MPHC) (Richalet et al., 1978) and Dynamic Matrix Control (Cutler and Ramaker, 1980). MPHC, whose software was called IDCOM (IDentification and COMmand), uses an impulse response, while DMC uses a step response in order to form predictions. Because of the use of these responses for control, applications are restricted in both cases to stable open loop systems. Both systems use finite control horizons and quadratic cost functions, that balance tracking performance against control effort, in the formulation of the predictive control problem. Both systems account for unknown disturbances by assuming that the disturbance remains constant over the full prediction horizon. This disturbance is taken as the difference between what the output was predicted to have been in the previous sample

step and what the actual output was at the current sample step. With the first generation of these systems, input and output constraints are handled in an ad hoc manner (Mayne et al., 2000). However, this was overcome in the second generation program, Quadratic DMC (QDMC) (García and Morshedi, 1986). This uses quadratic programming to solve the quadratic optimisation problem at each sample step, and can be used when the system is linear and when the control and state constraints are defined by linear inequalities.

Generalised Predictive Control, was proposed in (Clarke et al., 1987) and is based on the use of Controller Auto-Regressive Moving-Average (CARMA) and integrated CARMA (CARIMA) models for predictions. These models allow for more sophisticated modelling of noise and allow GPC to handle unstable and non-minimum phase plants. State-space MPC techniques were also developed around this time (Navratil et al., 1988; Li et al., 1989; Ricker, 1990) and allowed many of the results from state-space theory to be applied to MPC. Since all the states of the system are usually not measured, it is hence necessary to utilise an observer to provide an estimate of the full state vector. In the presence of white noise disturbances, and when the noise covariance matrices of the output are known, a Kalman filter can be used to determine the states that cannot be directly measured (Camacho and Bordons, 2003). However, stability was still not guaranteed with these early versions of MPC. Stable extensions of unconstrained MPC, such as Constrained Receding-Horizon Predictive Control (Clarke and Scattolini, 1991), were later developed which enforced a terminal equality constraint on the state in order to enforce stability. Stable Generalized Predictive Control (Kouvaritakis et al., 1992) stabilises the loop before applying the control, which guarantees closed loop stability. While these provided stabilising proofs for MPC, they were limited to the unconstrained case. Despite the difficulty of the problem at hand, a number of techniques were developed during the 1990s to deal with the constrained case. The use of terminal penalties and/or constraints, Lyapunov functions and the use of invariant sets were all used to ensure guaranteed system stability (Mayne et al., 2000).

Robust techniques have also been developed for MPC. These typically use a number of models to take into account system uncertainties and the control is applied so as to deal with the worst case scenario. MPC techniques have also been developed for nonlinear control, hybrid control and the control of very fast processes (Camacho and Bordons, 2003). Also, in recent years there has been much work on distributed MPC, where a number of agents communicate together to try and approximate the control of a centralised

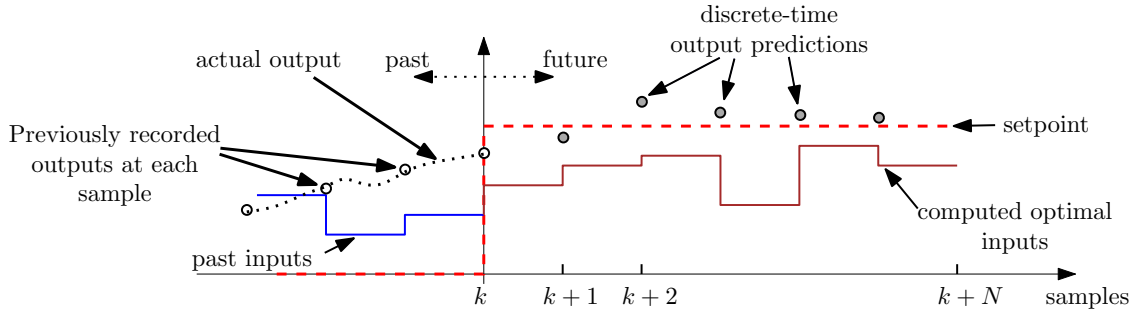


Figure 3.1: Model Predictive Control, showing past inputs and outputs, and predicted optimal inputs and outputs.

MPC agent (Scattolini, 2009). It is distributed MPC that forms the primary focus of the work in this thesis and so it will be dealt with in greater depth in the following chapters.

3.1.2 State-space MPC

In MPC, a control agent uses a discrete-time system model that predicts the system's future trajectory over a prediction horizon in order to calculate optimal inputs for the system over this horizon, as can be seen in Fig. 3.1. Only the input for the first time step is applied. At the next time step a new action is determined. MPC is often called Receding Horizon Control due to the prediction horizon moving forward at each time step. By linearising the system about an operating point, the system can then be described in continuous time using the following dynamic state-space equations:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}^{\mathrm{c}}\boldsymbol{x}(t) + \boldsymbol{B}^{\mathrm{c}}\boldsymbol{u}(t) + \boldsymbol{D}^{\mathrm{c}}\boldsymbol{d}(t), \tag{3.1}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}^{\mathrm{c}}\boldsymbol{x}(t), \tag{3.2}$$

where $\boldsymbol{x}(t) \in \Re^{n_x}$ is the state of the system at time t, $\boldsymbol{u}(t) \in \Re^{n_u}$ are the system's inputs, $\boldsymbol{d}(t) \in \Re^{n_d}$ are known disturbances, $\boldsymbol{y}(t) \in \Re^{n_y}$ are the outputs, and $\boldsymbol{A}^{\mathrm{c}}$, $\boldsymbol{B}^{\mathrm{c}}$, $\boldsymbol{D}^{\mathrm{c}}$, and $\boldsymbol{C}^{\mathrm{c}}$ are the continuous-time state-space matrices.

By converting (3.1) and (3.2) into the discrete domain, the system can be represented in discrete-time as follows:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{D}\boldsymbol{d}(k), \tag{3.3}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k), \tag{3.4}$$

where $\boldsymbol{x}(k) \in \Re^{n_x}$ is the state of the system, $\boldsymbol{u}(k) \in \Re^{n_u}$ are the inputs, $\boldsymbol{d}(k) \in \Re^{n_d}$ are known disturbances, and $\boldsymbol{y}_a(k) \in \Re^{n_y}$ are the outputs at sample step k, and $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{D}$, and $\boldsymbol{C}$ are the discrete-time state-space matrices. In this thesis, systems are discretised using the zero-order hold method. Using the zero-order hold with sample time τ gives:

$$\boldsymbol{A} = e^{\boldsymbol{A}^{\mathrm{c}}\tau} \tag{3.5}$$

$$\boldsymbol{B} = \int_0^{\tau} e^{\boldsymbol{A}^{\mathrm{c}}\eta}\boldsymbol{B}^{\mathrm{c}}\mathrm{d}\eta \tag{3.6}$$

$$\boldsymbol{D} = \int_0^{\tau} e^{\boldsymbol{A}^{\mathrm{c}}\eta}\boldsymbol{D}^{\mathrm{c}}\mathrm{d}\eta, \tag{3.7}$$

and $\boldsymbol{C}$=$\boldsymbol{C}^{\mathrm{c}}$.

Using (3.3) it is possible to predict a value for $\boldsymbol{x}(k+2)$ as follows:

$$\begin{aligned}\boldsymbol{x}(k+2) =& \boldsymbol{A}\boldsymbol{x}(k+1) + \boldsymbol{B}\boldsymbol{u}(k+1) + \boldsymbol{D}\boldsymbol{d}(k+1)\\ =& \boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{D}\boldsymbol{d}(k)) + \boldsymbol{B}\boldsymbol{u}(k+1) + \boldsymbol{D}\boldsymbol{d}(k+1)\\ =& \boldsymbol{A}^2\boldsymbol{x}(k) + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{B}\boldsymbol{u}(k+1) + \boldsymbol{A}\boldsymbol{D}\boldsymbol{d}(k) + \boldsymbol{D}\boldsymbol{d}(k+1)\end{aligned} \tag{3.8}$$

This can extended in a similar fashion in order to make state-space predictions up to N steps into the future. To simplify notation, the prediction vector, over a horizon N is first introduced. For a general vector $\boldsymbol{z}$, its prediction vector is $\tilde{\boldsymbol{z}}(k) = [\boldsymbol{z}^{\mathrm{T}}(k) \dots \boldsymbol{z}^{\mathrm{T}}(k+N-1)]^{\mathrm{T}}$. State-space and output predictions over an N step prediction horizon are then determined as follows:

$$\tilde{\boldsymbol{x}}(k+1) = \boldsymbol{A}^{\mathrm{f}}\boldsymbol{x}(k) + \boldsymbol{B}^{\mathrm{f}}\tilde{\boldsymbol{u}}(k) + \boldsymbol{D}^{\mathrm{f}}\tilde{\boldsymbol{d}}(k) \tag{3.9}$$

$$\tilde{\boldsymbol{y}}(k+1) = \boldsymbol{C}^{\mathrm{f}}\tilde{\boldsymbol{x}}(k+1) \tag{3.10}$$

where $\boldsymbol{A}^{\mathrm{f}}$, $\boldsymbol{B}^{\mathrm{f}}$, $\boldsymbol{D}^{\mathrm{f}}$, and $\boldsymbol{C}^{\mathrm{f}}$ are the state-space prediction matrices given as follows:

$$\tilde{\boldsymbol{x}}(k+1) = \begin{pmatrix}\boldsymbol{x}(k+1)\\ \boldsymbol{x}(k+2)\\ \vdots\\ \boldsymbol{x}(k+N)\end{pmatrix}, \boldsymbol{A}^{\mathrm{f}} = \begin{pmatrix}\boldsymbol{A}\\ \boldsymbol{A}^2\\ \vdots\\ \boldsymbol{A}^N\end{pmatrix}, \boldsymbol{B}^{\mathrm{f}} = \begin{pmatrix}\boldsymbol{B} & 0 & \dots & 0\\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \dots & 0\\ \vdots & & \ddots & \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \dots & \boldsymbol{B}\end{pmatrix},$$

$$\boldsymbol{D}^{\mathrm{f}} = \begin{pmatrix}\boldsymbol{D} & 0 & \dots & 0\\ \boldsymbol{A}\boldsymbol{D} & \boldsymbol{D} & \dots & 0\\ \vdots & & \ddots & \\ \boldsymbol{A}^{N-1}\boldsymbol{D} & \boldsymbol{A}^{N-2}\boldsymbol{D} & \dots & \boldsymbol{D}\end{pmatrix}, \boldsymbol{C}^{\mathrm{f}} = \begin{pmatrix}\boldsymbol{C} & & & \\ & \boldsymbol{C} & & \\ & & \ddots & \\ & & & \boldsymbol{C}\end{pmatrix}.$$

In order to attain integral action with MPC, an augmented state-space model, that describes the evolution of the state from sample to sample in terms of incremental inputs is used for predictions. This augmented state-space model is given by (Wang, 2009):

$$\boldsymbol{x}^{\text{aug}}(k+1) = \begin{bmatrix} \Delta\boldsymbol{x}(k+1) \\ \boldsymbol{x}(k+1) \end{bmatrix} = \hat{\boldsymbol{A}}\boldsymbol{x}^{\text{aug}}(k) + \hat{\boldsymbol{B}}\Delta\boldsymbol{u}(k) + \hat{\boldsymbol{D}}\Delta\boldsymbol{d}(k) \tag{3.11}$$

$$\boldsymbol{y}(k) = \hat{\boldsymbol{C}}\boldsymbol{x}^{\text{aug}}(k), \tag{3.12}$$

where

$$\hat{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0}_{n_x \times n_x} \\ \boldsymbol{A} & \boldsymbol{\mathrm{I}}_{n_x \times n_x} \end{bmatrix}, \ \hat{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{B} \end{bmatrix}, \ \hat{\boldsymbol{D}} = \begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{D} \end{bmatrix}, \ \hat{\boldsymbol{C}} = \begin{bmatrix} \boldsymbol{0}_{n_y \times n_y} \\ \boldsymbol{C} \end{bmatrix}^{\text{T}}.$$

Here the Δ operator is used to denote the change in a given variable from sample to sample, i.e. $\Delta z(k) = z(k) - z(k-1)$. Predictions using these matrices can then be formed in the same way as in (3.9) and (3.10). Matrices $\hat{\boldsymbol{A}}^{\text{f}}$, $\hat{\boldsymbol{B}}^{\text{f}}$, $\hat{\boldsymbol{D}}^{\text{f}}$, $\hat{\boldsymbol{V}}^{\text{f}}$, and $\hat{\boldsymbol{C}}^{\text{f}}$ are used to denote the prediction matrices in the incremental case.

3.1.3 MPC Implementation

In a system of n subsystems, MPC problems are constructed to fulfill control objectives for subsystems $a = 1, \ldots, n$ based on knowledge of $\boldsymbol{x}(k)$, and $\tilde{\boldsymbol{d}}(k)$ if there are measurable disturbances. When there are no system constraints, a cost function, $J(\boldsymbol{x}^{\text{aug}}(k), \Delta\tilde{\boldsymbol{u}}(k), \Delta\tilde{\boldsymbol{d}}(k))$ (which will henceforth be denoted by $J(k)$), embodies the control objectives of the system. The control agent determines the best incremental control inputs $\Delta\tilde{\boldsymbol{u}}(k)$ to be applied over the horizon by minimising this cost function as follows:

$$\begin{gathered} \Delta\tilde{\boldsymbol{u}}(k) = \arg\min_{\Delta\tilde{\boldsymbol{u}}} J(k) \\ \text{which is the solution of } \frac{\partial}{\partial\Delta\tilde{\boldsymbol{u}}(k)} J(k) = 0. \end{gathered} \tag{3.13}$$

The cost at sample time k is,

$$J(k) = \sum_{p=0}^{N-1} J^{\text{stage}}(k, p), \tag{3.14}$$

where $J^{\text{stage}}(k, p)$ is the cost at the p^{th} step of the prediction horizon at sample step k, typically defined as the following quadratic cost function:

$$J^{\text{stage}}(k, p) = \boldsymbol{e}^{\text{T}}(k+p+1)\boldsymbol{Q}\boldsymbol{e}(k+p+1) + \Delta\boldsymbol{u}^{\text{T}}(k+p)\boldsymbol{R}\Delta\boldsymbol{u}(k+p), \tag{3.15}$$

where $\boldsymbol{e}(k+p)$ is the vector of errors in the MPC problem at the p^{th} stage of the prediction horizon, at sample time k. The error, $\boldsymbol{e}(k+p) = \boldsymbol{y}(k+p) - \boldsymbol{r}(k+p)$, where $\boldsymbol{r}(k+p)$ is a vector of the setpoints. Using the stage cost in (3.15), $J(k)$ represents the desire to minimise the square of the error over the prediction horizon, i.e., to follow as closely as possible the setpoint over the prediction horizon. The weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ determine the relative importance of the minimisation of the error and the control effort from sample to sample during optimisation, respectively. The tuning of these parameters significantly influences the behaviour of the control system. For the unconstrained case, and using an incremental model, an analytical solution for the inputs can then be found by finding the value of $\Delta\tilde{\boldsymbol{u}}(k)$ that minimises $J(k)$:

$$\begin{aligned} J(k) \;=\; & ((\boldsymbol{C}^{\mathrm{f}}(\boldsymbol{A}^{\mathrm{f}}(k)\boldsymbol{x}(k) + \boldsymbol{B}^{\mathrm{f}}\Delta\tilde{\boldsymbol{u}}(k) + \boldsymbol{D}^{\mathrm{f}}\Delta\tilde{\boldsymbol{d}}(k)) - \tilde{\boldsymbol{r}}(k+1))^{\mathrm{T}}\boldsymbol{Q}((\boldsymbol{C}^{\mathrm{f}}(\boldsymbol{A}^{\mathrm{f}}(k)\boldsymbol{x}(k) \\ & + \boldsymbol{B}^{\mathrm{f}}\Delta\tilde{\boldsymbol{u}}(k) + \boldsymbol{D}^{\mathrm{f}}\Delta\tilde{\boldsymbol{d}}(k)) - \tilde{\boldsymbol{r}}(k+1)) + \Delta\tilde{\boldsymbol{u}}^{\mathrm{T}}(k)\boldsymbol{R}\Delta\tilde{\boldsymbol{u}}(k)) \\ =\; & \Delta\tilde{\boldsymbol{u}}^{\mathrm{T}}(k)\boldsymbol{H}\Delta\tilde{\boldsymbol{u}}(k) + \Delta\tilde{\boldsymbol{u}}^{\mathrm{T}}(k)\boldsymbol{f} + \mu \end{aligned} \tag{3.16}$$

where the square symmetric matrix $\boldsymbol{H}{=}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{C}\boldsymbol{B} + \boldsymbol{R}$, the vector $\boldsymbol{f}{=}2(\boldsymbol{C}(\boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{D}\Delta\tilde{\boldsymbol{d}}(k)) - \tilde{\boldsymbol{r}}(k+1))^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{C}\boldsymbol{B}$, and the scalar $\mu{=}(\boldsymbol{C}(\boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{D}\Delta\tilde{\boldsymbol{d}}(k)) - \tilde{\boldsymbol{r}}(k+1))^{\mathrm{T}} \times \boldsymbol{Q}(\boldsymbol{C}(\boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{D}\Delta\tilde{\boldsymbol{d}}(k)) - \tilde{\boldsymbol{r}}(k+1))^{\mathrm{T}}$. μ does not depend on $\Delta\tilde{\boldsymbol{u}}(k)$.

The optimal choice of controls $\Delta\tilde{\boldsymbol{u}}(k)$ is obtained when,

$$\frac{\partial}{\partial\Delta\tilde{\boldsymbol{u}}(k)} {=}2\boldsymbol{H}\Delta\tilde{\boldsymbol{u}}(k) + \boldsymbol{f} = 0 \tag{3.17}$$

This yields the solution,

$$\begin{aligned} \Delta\tilde{\boldsymbol{u}}(k) &= -\frac{1}{2}\boldsymbol{H}^{-1}\boldsymbol{f} \\ &= -(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{C}\boldsymbol{B} + \boldsymbol{R})^{-1}((\boldsymbol{C}(\boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{D}\Delta\tilde{\boldsymbol{d}}(k)) - \tilde{\boldsymbol{r}}(k+1))^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{C}\boldsymbol{B}) \end{aligned} \tag{3.18}$$

Only the value for the optimal $\Delta\boldsymbol{u}(k)$ is applied to the subsystem after optimisation, and this process is repeated every time step, with the new prediction horizon moving forward one time step. Because of the use of a quadratic error function based on linear state-space models, this unconstrained controller is similar to a Linear Quadratic Regulator (LQR). The use of this state-space based formulation means it is necessary to use an observer or Kalman filter in real applications in order to determine unmeasured system states. In this thesis, however, it is assumed that there is access to all system states for all applications.

3.1.4 Constrained MPC

While the unconstrained version of MPC enables powerful system control, one of the main advantages of modern MPC is its ability to incorporate system constraints into its problem. Typically, there would be constraints on the upper and lower levels of the systems outputs, $\boldsymbol{y} \geq \boldsymbol{y}_{\min}$, $\boldsymbol{y} \leq \boldsymbol{y}_{\max}$, where $\boldsymbol{y}_{\min}$ and $\boldsymbol{y}_{\max}$ are the lower and upper bounds on the output, respectively. Similarly, there would be constraints on the inputs, $\boldsymbol{u} \geq \boldsymbol{u}_{\min}$, $\boldsymbol{u} \leq \boldsymbol{u}_{\max}$, where $\boldsymbol{u}_{\min}$ and $\boldsymbol{u}_{\max}$ are the lower and upper bounds on the output, respectively. Rate constraints are also common and determine the limits as to how fast a system variable is allowed to change from sample to sample. These are given by, $\Delta\boldsymbol{p} \geq \Delta\boldsymbol{p}_{\min}$, $\Delta\boldsymbol{p} \leq \Delta\boldsymbol{p}_{\max}$, where p is the variable in question, and $\Delta\boldsymbol{p}_{\min}$ and $\Delta\boldsymbol{p}_{\max}$ are the maximum and minimum allowable changes in $\boldsymbol{p}$ from sample to sample. These constraints can also be divided into hard constraints, which are constraints that absolutely must be satisfied, and soft constraints, which are constraints that should be satisfied if possible (Rossiter, 2003). An example of a hard constraint would be the limit on an actuator or valve, which must lie between 0 and 100% open. There is no point therefore in asking a controller to go beyond this point. An example of a soft constraint would be a limit that ensures a piece of equipment is not overly stressed so that it does not get worn out over time. It is desirable to satisfy soft constraints but they can be exceeded if necessary. The general constrained MPC problem can be formulated in the following way:

$$\begin{aligned} \tilde{\boldsymbol{u}}(k) = &\arg\min_{\tilde{\boldsymbol{u}}} J(\boldsymbol{x}(k), \tilde{\boldsymbol{u}}(k), \tilde{\boldsymbol{d}}(k)) \\ &\text{subject to constraints} \begin{cases} \kappa_i(\boldsymbol{x}(k), \tilde{\boldsymbol{u}}(k), \tilde{\boldsymbol{d}}(k)) \geq 0, & i \in \mathcal{I} \\ \kappa_j(\boldsymbol{x}(k), \tilde{\boldsymbol{u}}(k), \tilde{\boldsymbol{d}}(k)) = 0, & j \in \mathcal{E}, \end{cases} \end{aligned} \tag{3.19}$$

where $\mathcal{I}$ and $\mathcal{E}$ are two finite sets of indices, and $\kappa_i(\boldsymbol{x}(k), \tilde{\boldsymbol{u}}(k), \tilde{\boldsymbol{d}}(k))$, for $i \in \mathcal{I}$, and $\kappa_j(\boldsymbol{x}(k), \tilde{\boldsymbol{u}}(k), \tilde{\boldsymbol{d}}(k))$, for $j \in \mathcal{E}$, are the system's inequality and equality constraints, respectively. A non-incremental model based on $\boldsymbol{x}(k)$, $\tilde{\boldsymbol{u}}(k)$, and $\tilde{\boldsymbol{d}}(k)$ is used in (3.19) for notational convenience. In order to solve this problem, constrained optimisation methods must be used. Some background on deterministic constrained optimisation techniques can be found in Appendix A. When a quadratic cost function is used, Quadratic Programming techniques can be used to efficiently solve the linear quadratic forms of (3.19). Many commercial packages, such as quadprog found in Matlab, are available for efficiently solving this problem.

Many stable constrained MPC systems use infinite horizons and terminal constraints in

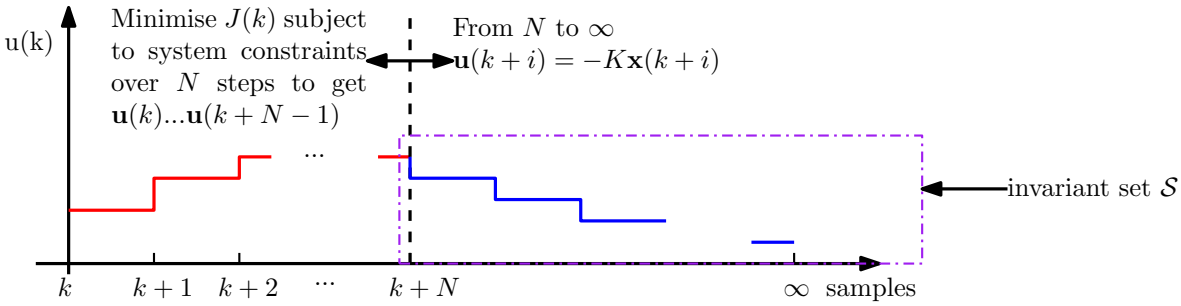


Figure 3.2: Stability constrained dual mode MPC.

order to ensure stable control. For example, dual mode type MPC algorithms use two modes of operation in the prediction horizon to provide infinite horizon stable control. The first mode uses constrained optimisation over the first N prediction steps. A second mode of operation, which uses a fixed feedback law, $\boldsymbol{u}(k+i) = -K\boldsymbol{x}(k+i)$, is employed for states between N and ∞, as can be seen in Fig. 3.2. The cost over this part of the prediction horizon is evaluated using a terminal cost, $\boldsymbol{x}(k+N)\boldsymbol{P}\boldsymbol{x}(k+N)$. The concept of an invariant set is used with the infinite horizon control to ensure asymptotic stability of the system. An invariant set $\mathcal{S}$ is a set where it is guaranteed that if $\boldsymbol{x}(k) \in \mathcal{S}$, then $\boldsymbol{x}(k+1) \in \mathcal{S}$, i.e. once a state has entered this set it does not leave it again. An invariant set can be constructed such that the evolution of states under the second mode of operation, $\boldsymbol{u}(k+i) = -K\boldsymbol{x}(k+i)$, satisfies this set, which in turn is designed to satisfy the system constraints. The system will be then be stable provided that the system guides $\boldsymbol{x}(k+N)$ into this invariant set in the first mode of operation (Rossiter, 2003). Many other stable constrained versions of MPC take a similar approach to this (Mayne et al., 2000). However in this thesis, only finite horizon MPC is used. An example of the use of centralised MPC in a smart grid control scenario will now be given.

3.2 Model Predictive Control for Load Frequency Control

In both this chapter and the previous one, the argument has been made that modern control techniques and power electronics devices will enable the transition to the Smart Grid. A simple example will now be given demonstrating the application of MPC to a 3 area Load Frequency Control (LFC) System, and the difference in the achievable control performance, both with and without a Series Synchronous Static Compensator, will be

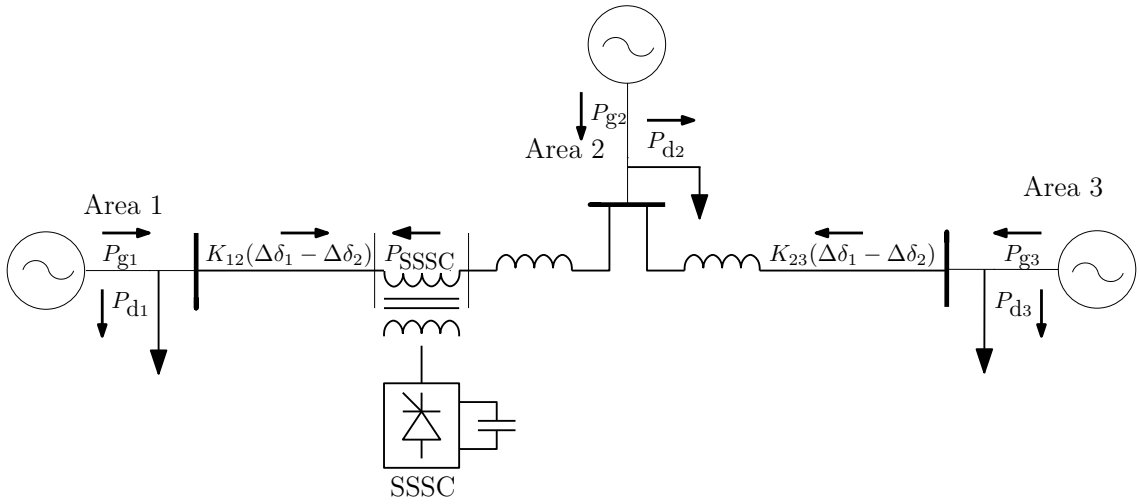


Figure 3.3: The 3 area LFC system with a Static Synchronous Series Compensator between areas 1 and 2.

illustrated. The quick reaction times of the SSSC allows for the increased controllability of the power network and significantly improves system damping for LFC (Bhatt et al., 2010; Zareiegovar et al., 2010; Bhatt et al., 2009). In LFC, it is desired to maintain the frequency of a power system as close to 50 Hz (or 1 per unit (pu) frequency, which is the normalised frequency) as possible at all times. This is done by ensuring that the supplied power closely tracks the demanded power at all times in the network. Agents must be capable of returning the frequency in the area they control to the 1 pu setpoint after disturbances such as load disturbances and line faults. The agents' individual problems are coupled due to power flowing between subsystems through AC or DC line connections.

3.2.1 System Description

The 3 area LFC system is given in Fig. 3.3. Areas 2 and 3 are connected by a normal AC line and areas 1 and 3 are connected by an AC line with an SSSC. Non-reheat turbines are used in the models of the generators. The continuous time linearised dynamics of the system about a particular operating point are given as follows (Kundur, 1994):

$$\dot{\delta_a}(t) = \omega_a(t), \tag{3.20}$$

$$\dot{\omega}_a(t) = \frac{1}{M_a}(P_{\mathrm{G}a}(t) + b_a P_{\mathrm{sssc}}(t) - P_{\mathrm{d}a}(t) - D_a\omega_a(t) - \sum_{j\in\mathcal{N}_a} K_{aj}(\delta_a(t) - \delta_j(t))), \tag{3.21}$$

$$\dot{P}_{\text{sssc}}(t) = \frac{1}{T_{\text{sssc}}}(u_{\text{sssc}}(t) - P_{\text{sssc}}(t)), \tag{3.22}$$

$$\dot{P}_{\text{G}a}(t) = \frac{1}{T_{ta}}(-P_{\text{G}a}(t) + X_{\text{G}a}(t)), \tag{3.23}$$

$$\dot{X}_{\text{G}a}(t) = \frac{1}{T_{\text{G}a}}(-\frac{\omega(t)}{R_a} - X_{\text{G}a}(t) + u_a(t)), \tag{3.24}$$

where $\delta_a(t)$ is the rotor position of generator a, $\omega_a(t)$ is the pu rotor frequency at generator a, $P_{\text{G}a}(t)$ is the pu active power generated by generator a, $P_{\text{sssc}}(t)$ is the pu active power controlled by the SSSC, $P_{\text{d}a(t)}$ is the pu power load in area a, $u_{\text{sssc}}(t)$ is the control input to the SSSC, $X_{\text{G}a}(t)$ is the governor valve position and $u_a(t)$ is the load reference setpoint control input at time t. The constant M_a=$2H_a$ where H_a is the inertia of generator a in seconds, D_a is the load damping constant, K_{aj} is a synchronising coefficient between areas a and j, $\mathcal{N}_a$ is the set of area indices that are AC connected to area a, T_{sssc} is a time constant for the SSSC in seconds, $T_{\text{t}a}$ is the turbine time constant in seconds, and $T_{\text{G}a}$ is the governor time constant in seconds, and R_a gives the governors' in-built speed regulation due to governor action. All variables represent small deviations from their respective operating points.

The coefficient $b_a \in \{-1, 0, 1\}$ determines the direction of the SSSC power flow. Here, $b_a = 1$ implies that an area receives power from the SSSC, $b_a = -1$ implies that an area loses power from the SSSC, and $b_a = 0$ implies an area is not connected to the SSSC. In the simulations where there is no SSSC between areas 1 and 2, the equations used are the same, except $P_{\text{sssc}}(t)$-related variables are removed from the equations, and equation (3.22) is not considered.

SSSCs are modelled here as series power sources. Using a linearised model, the power travelling along an AC line between 2 connected areas is $P_{aj} = K_{aj}(\delta_a - \delta_j)$. Where an SSSC is present, the power travelling between the 2 areas can be given as $P_{aj} = K_{aj}(\delta_a - \delta_j) + P_{\text{sssc}}$ (Bhatt et al., 2010). In reality, the power P_{sssc} is controlled by manipulation of the SSSC voltage, but in this example P_{sssc} itself is calculated and applied directly to the system.

3.2.2 State-space MPC Formulation

The following state-space model can be constructed from equations (3.20)-(3.24) as follows:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \tag{3.25}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t), \tag{3.26}$$

where $\boldsymbol{x}(t)$=[$\delta_1(t)$, $\omega_1(t)$, $P_{\mathrm{G1}}(t)$, $P_{\mathrm{sssc}}(t)$, $X_{\mathrm{G1}}(t)$, $\delta_2(t)$, $\omega_2(t)$, $P_{\mathrm{G2}}(t)$, $X_{\mathrm{G2}}(t)$, $\delta_3(t)$, $\omega_3(t)$, $P_{\mathrm{G3}}(t)$, $X_{\mathrm{G3}}(t)$]$^{\mathrm{T}}$, $\boldsymbol{u}(t)$ = [u_1, u_{sssc}, u_2, u_3]$^{\mathrm{T}}$, and $\boldsymbol{y}(t)$ = [$\omega_1(t)$, $\omega_2(t)$, $\omega_3(t)$]$^{\mathrm{T}}$. Matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are constructed based on (3.20)-(3.24). As disturbances are not known, they are not included in the state-space model for control. System parameters are given in Table 3.1.

Area a	1	2	3
M_a (s)	0.15	0.2	0.167
b_a	1	-1	0
D_a	0.005	0.0083	0.006
R_a	2.4	2.4	2.4
T_{ta} (s)	0.3	0.25	0.3
T_{Ga} (s)	0.1	0.1	0.1
K_{12}=K_{21}	1	K_{23}=K_{32}	1
T_{sssc} (s)	0.05		

Table 3.1: Parameters in per unit form of the LFC SSSC system

The continuous-time state-space model was then discretised using a the zero-order-hold method described in Section 3.1.2, with a sample time τ=25ms. An incremental state-space model based on the augmented state $\boldsymbol{x}^{\mathrm{aug}}(k)$, and incremental input $\Delta\boldsymbol{u}(k)$ was constructed from the discretised state-space model, as in (3.11) and (3.12).

Using the augmented state-space model, state-space predictions were made as in (3.9) and (3.10), with a prediction horizon of $N = 10$ to give:

$$\tilde{\boldsymbol{x}}^{\mathrm{aug}}(k+1) = \boldsymbol{A}^{\mathrm{f}}\boldsymbol{x}^{\mathrm{aug}}(k) + \boldsymbol{B}^{\mathrm{f}}\Delta\tilde{\boldsymbol{u}}(k), \tag{3.27}$$

$$\tilde{\boldsymbol{y}}(k+1) = \boldsymbol{C}^{\mathrm{f}}\tilde{\boldsymbol{x}}^{\mathrm{aug}}(k+1), \tag{3.28}$$

where $\boldsymbol{A}^{\mathrm{f}}$, $\boldsymbol{B}^{\mathrm{f}}$ and $\boldsymbol{C}^{\mathrm{f}}$ are the prediction matrices associated with $\boldsymbol{x}^{\mathrm{aug}}(k)$, $\Delta\tilde{\boldsymbol{u}}(k)$, and $\tilde{\boldsymbol{y}}(k+1)$, respectively.

Using the prediction matrices, the following cost function was used for the MPC optimisations which balances the minimisation of the square of the error at each of the outputs over the prediction horizon and against the effort cost expended:

$$\begin{aligned} J(k) &= (\tilde{\boldsymbol{y}}(k+1) - \tilde{\boldsymbol{r}}(k+1))\boldsymbol{Q}(\tilde{\boldsymbol{y}}(k+1) - \tilde{\boldsymbol{r}}(k+1)) + \Delta\tilde{\boldsymbol{u}}(k)\boldsymbol{R}\Delta\tilde{\boldsymbol{u}}(k) \\ &= (\boldsymbol{C}^{\mathrm{f}}(\boldsymbol{A}^{\mathrm{f}}\boldsymbol{x}^{\mathrm{aug}}(k) + \boldsymbol{B}^{\mathrm{f}}\Delta\tilde{\boldsymbol{u}}(k)) - \tilde{\boldsymbol{r}}(k+1))\boldsymbol{Q}(\boldsymbol{C}^{\mathrm{f}}(\boldsymbol{A}^{\mathrm{f}}\boldsymbol{x}^{\mathrm{aug}}(k) + \boldsymbol{B}^{\mathrm{f}}\Delta\tilde{\boldsymbol{u}}(k)) - \tilde{\boldsymbol{r}}(k+1)) \\ &\quad + \Delta\tilde{\boldsymbol{u}}(k)\boldsymbol{R}\Delta\tilde{\boldsymbol{u}}(k) \\ &= (\boldsymbol{P}\boldsymbol{x}^{\mathrm{aug}}(k) + \boldsymbol{G}\Delta\tilde{\boldsymbol{u}}(k) - \tilde{\boldsymbol{r}}(k+1))\boldsymbol{Q}(P\boldsymbol{x}^{\mathrm{aug}}(k) + \boldsymbol{G}\Delta\tilde{\boldsymbol{u}}(k) - \tilde{\boldsymbol{r}}(k+1)) \\ &\quad + \Delta\tilde{\boldsymbol{u}}(k)\boldsymbol{R}\Delta\tilde{\boldsymbol{u}}(k), \end{aligned} \tag{3.29}$$

where $\boldsymbol{P} = \boldsymbol{C}^{\mathrm{f}}\boldsymbol{A}^{\mathrm{f}}$, $\boldsymbol{G} = \boldsymbol{C}^{\mathrm{f}}\boldsymbol{B}^{\mathrm{f}}$, and $\boldsymbol{Q}$ and $\boldsymbol{R}$ are diagonal matrices that determine the relative importance of minimising the square of the errors in each area and the control effort at each step over the prediction horizon.

It is then possible to represent $J(k)$ in a quadratic cost term as

$$J(k) = \Delta\tilde{\boldsymbol{u}}^{\mathrm{T}}(k)\boldsymbol{H}\Delta\tilde{\boldsymbol{u}}(k) + \Delta\tilde{\boldsymbol{u}}^{\mathrm{T}}(k)\boldsymbol{f} + \mu, \tag{3.30}$$

where matrix $\boldsymbol{H}$=$\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{G} + \boldsymbol{R}$, vector $\boldsymbol{f} = 2\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{P}\tilde{\boldsymbol{x}}^{\mathrm{aug}}(k+1) - 2\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}_F\tilde{\boldsymbol{r}}(k+1)$ and μ is a scalar which is not dependent on $\Delta\tilde{\boldsymbol{u}}(k)$, and therefore does not affect the optimisation. This then allows the MPC problem to be solved using a standard quadratic solver at each sample as follows:

$$\Delta\tilde{\boldsymbol{u}}(k) = \arg\min_{\Delta\tilde{\boldsymbol{u}}} J(k) \quad \text{subject to constraints,} \tag{3.31}$$

The system used here is a linearised representation of the real system about an operating point. The operating point for the frequency is 1 pu. Therefore the goal of linearised system is to keep the frequency deviations ω_a, for $a = 1, 2, 3$, as close to 0 as possible at all times. Thus, $\tilde{\boldsymbol{r}}(k+1)$ here is a vector of zeros.

An equal weighting was given to the minimisation of the error in each of the outputs and so $\boldsymbol{Q}$=diag(10,...,10) is a diagonal matrix of size $Nn_y \times Nn_y$. The control effort cost weight $\boldsymbol{R}$=diag(0.001,...,0.001) is a matrix of size $Nn_u \times Nn_u$. This is kept small and it's

function here is more to ensure full rank of the optimisation problem than to minimise the control effort from sample to sample.

Several constraints are placed on the real system and these are also applied to the MPC optimisations. These are given in Table 3.2.

Variable	Lower bound	Upper bound
u_{sssc} pu	-0.1	0.1
P_{G1} pu	-0.02	0.02
P_{G2} pu	-0.04	0.04
P_{G3} pu	-0.02	0.02
$\Delta P_{\text{G}a}$ pu, for $a = 1, 2, 3$	-0.003	0.003
$\Delta\delta_a$ pu, for $a = 1, 2, 3$	-0.005	0.005
ω_a pu, for $a = 1, 2, 3$	-0.016	0.016

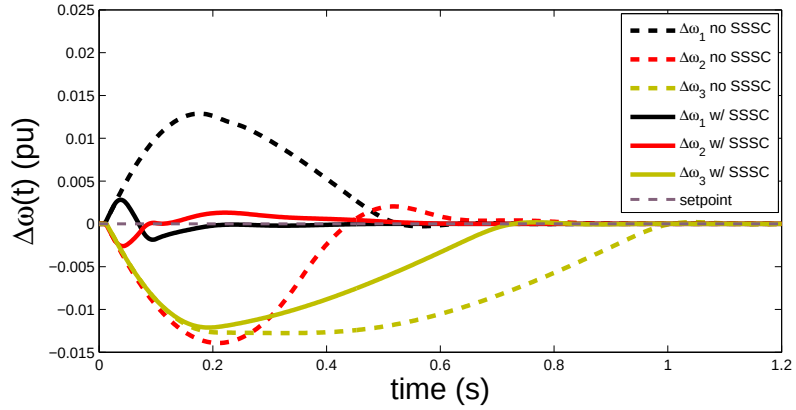
Table 3.2: Constraints for the LFC SSSC system

3.2.3 Experimental Results

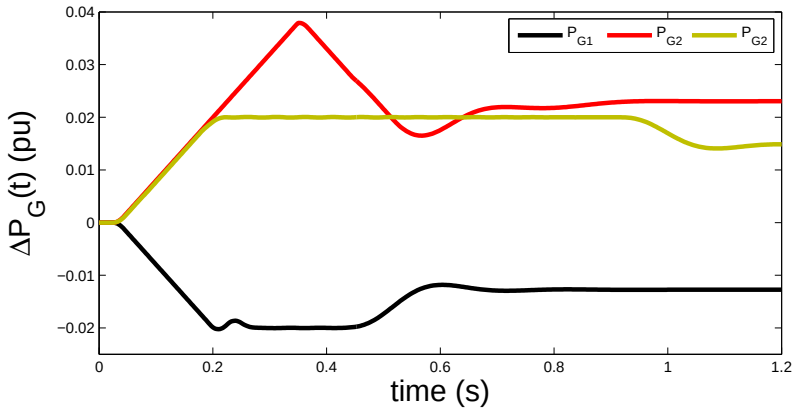
Two different simulations were conducted, one in which the SSSC was positioned between areas 1 and 2 and another in which there was no SSSC between the areas. Load power disturbances of magnitude -0.02, 0.025, and 0.02 pu were applied to areas 1,2 and 3, respectively, at times t=0.01, 0.011, and 0.013 seconds, respectively. Simulations were run on a computer with an Intel® Core™ 2 6400 operating at 2.13 GHz and with 3 GB of RAM in Matlab 7.6.0 (2008a). All MPC calculations were performed using quadprog. The results of the simulations can be seen in Fig. 3.4.

In both simulations, the MPC managed to control the system within the bounds on states and variables. The addition of the SSSC significantly improved the system damping as can be seen in Fig. 3.4(a). This is an illustration of how the FACTS devices and MPC give significant controllability of the system.

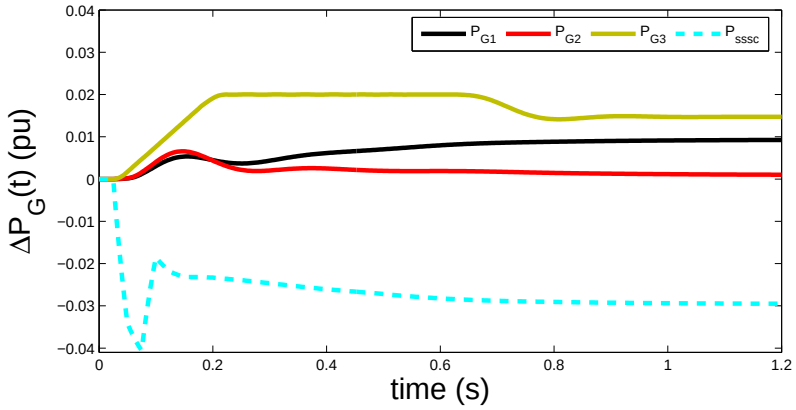
The average and maximum cpu times taken to run MPC optimisations at each sample



(a) Plot of the frequencies both with the SSSC and without the SSSC between areas 1 and 2.



(b) Plot of the power generated by each generator for the simulation with no SSSC.



(c) Plot of the power generated by generator and the SSSC power when the SSSC is between areas 1 and 2 .

Figure 3.4: Plots from simulations both with the SSSC and without the SSSC between areas 1 and 2.

for the non-SSSC case was 0.0672s and 0.7188s, respectively. The average and maximum cpu times taken to run MPC optimisations at each sample with the SSSC between areas 1 and 2 was 0.0439s and 0.6094s, respectively (the cpu time is the total time measured in all processor cores on a computer. The actual time taken is usually equal to the cpu time divided by the number of cores). Longer average and maximum cpu times would be expected with the SSSC in place rather than in the the non-SSSC case, due to an increase in the dimension of the optimisation problem resulting from the extra input variable. However, these are both reduced when the SSSC is in place. This could be due to the fact that when the SSSC is in place the ΔP_{G1} bounds are not active for much of the simulation, unlike in the case without the SSSC in place.

It should be noted here, that the times taken to calculate the MPC optimisations exceed the sample time of 25 ms. This highlights the fact that in power systems, it is necessary that any optimisations are calculated as efficiently as possible so that inputs can be calculated and applied within each sample step. For larger centralised MPC problems it becomes even more difficult to perform the optimisations within the given time constraints and so this motivates the use of efficient non-centralised control techniques in power systems.

3.3 Summary

In this section Model Predictive Control was looked at in depth. It was seen how MPC can conveniently incorporate system constraints into its formulation while being capable of controlling a wide variety of systems. The chapter was then concluded with an example of the application of MPC to a 3 area LFC system. Two different scenarios were examined, one in which an SSSC was connected between two areas and another where the SSSC was not present. The MPC provided adequate performance in both cases but was capable of significantly improved control when the SSSC was in place. This was an illustrative example of how the combination of MPC and an SSSC allows for a high level of controllability in highly interconnected power systems. It was also noted that as the size of the MPC problem grows, that it is not possible to use centralised MPC in power systems, due to the time constraints placed on the calculation of the control inputs, which motivates the development of non-centralised MPC techniques which are more efficient

than the centralised problem.

References

P. Bhatt, S.P. Ghoshal, and R. Roy. Optimized Automatic Generation Control by SSSC and TCPS in Coordination with SMES for Two-Area Hydro-hydro Power System. In *International Conference on Advances in Computing, Control, and Telecommunication Technologies*, pages 474 –480, Trivandrum, Kerala, Dec. 2009.

P. Bhatt, R. Roy, and S.P. Ghoshal. Coordinated control of SSSC and SMES in competitive electricity market for load frequency control. In *11th IEEE International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 42 –47, Singapore, June 2010.

E. F. Camacho and C. Bordons. *Model Predictive Control, 2nd Edition*. Springer, London, UK, 2003.

D.W. Clarke and R. Scattolini. Constrained receding-horizon predictive control. *Control Theory and Applications, IEE Proceedings D*, 138(4):347–354, July 1991.

D.W. Clarke, C. Mohtadi, and P.S. Tuffs. Generalized predictive controlpart i. the basic algorithm. *Automatica*, 23(2):137 – 148, 1987.

C.R. Cutler and B.C. Ramaker. Dynamic Matrix Control-A computer control algorithm. In *Proceedings of the joint automatic control conference*, San Franciso, USA, 1980.

C.E. García and A.M. Morshedi. Quadratic programming solution of dynamic matrix control (qdmc). *Chemical Engineering Communications*, 46:73–87, 1986.

B. Kouvaritakis, J.A. Rossiter, and A.O.T. Chang. Stable generalised predictive control: an algorithm with guaranteed stability. *IEE Proceeceeding in Control Theory and Applications, Part D*, 139(4):349 – 362, July 1992.

P. Kundur. *Power System Stability and Control*. Mc-Graw Hill, New York, 1994.

Sifu Li, Kian Y. Lim, and D. Grant Fisher. A state space formulation for model predictive control. *AIChE Journal*, 35(2):241–249, 1989.

J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, England, 2002.

D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

J. P. Navratil, K. Y. Lim, and D. G. Fisher. Disturbance feedback in model predictive control systems. In *Model-Based Process Control-Proceedings of the 1988 IFAC Workshop*, pages 63–68, Oxford, UK, 1988. Pergamon Press.

J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, Wisconsin, 2009.

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

N. Lawrence Ricker. Model predictive control with state estimation. *Industrial & Engineering Chemistry Research*, 29(3):374–382, 1990.

J.A. Rossiter. *Model Based Predictive Control-A Practical Approach*. CRC Press, Florida, 2003.

R. Scattolini. Architectures for distributed and hierarchical Model Predictive Control - A review. *Journal of Process Control*, 19(5):723–731, 2009.

Gholamreza Zareiegovar, Hosein Shayeghi, Aidin Sakhavati, and Vahid Nabaei. A new scheme to control SSSC in interconnected power systems. In *9th International Conference on Environment and Electrical Engineering (EEEIC)*, pages 202 –205, Prague, Czech Republic, May 2010.

Chapter 4

Distributed Model Predictive Control

4.1 Introduction

In previous chapters the effectiveness of the use of Model Predictive Control (MPC) for the control of power systems has been demonstrated. However, as has previously been stated in Chapter 2, centralised MPC is not viable for the control of large scale power networks. The size of the MPC problem for a full power network is in general not solvable within the time scales involved. Also, because of deregulation, a number of control agents may be responsible for the control of different sections of the grid. Because of the scale of the problem involved and the impracticality of all agents sharing internal network information, such as their respective state vectors and the weights used in the MPC problem, Nash equilibrium seeking algorithms are more appropriate for use for power network control. This involves the communication of interconnecting variables between the agents in control of subsystems which share a variable in their respective control problems (Camponogara et al., 2002; Liu et al., 2010; Hermans et al., 2010; Negenborn et al., 2008). These algorithms have proven to provide adequate levels of control when applied to power network situations.

The distributed MPC algorithm used in this thesis was originally developed in (Negenborn, 2007). It is based on the decomposition of the original MPC problem into a number of smaller subproblems which are then coordinated in an iterative fashion using the Alternating Direction Method of Multipliers (ADMOM) (Tosserams, 2008; Bertsekas and Tsitsikilis, 1989). The distributed MPC technique will now be presented and then a new stability proof for the linear unconstrained version of this algorithm will be given.

4.2 Distributed MPC

Before showing how the distributed MPC is derived, some important variables will first be defined. The discrete-time, linear, time-invariant state-space model used to model the system dynamics is given again here for convenience:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{D}\boldsymbol{d}(k), \tag{4.1}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k), \tag{4.2}$$

This system is taken as consisting of n subsystems, where each subsystem consists of a set of nodes (a node being an individual point in a network that can be described using a combination of variables and equations) and the interconnections between these nodes. A number of factors can affect the way in which the system is decomposed into the different subsystems. Subsystems may be decomposed in a manner that minimises the level of interaction between different areas in order to simplify the control of the system (Ocampo-Martinez et al., 2011). There may be information or communication constraints in a system which results in a natural division into different subsystems. Subsystems could also arise through commercial reasons, e.g., different companies being responsible for the control of different areas of the grid.

In this thesis subsystems are assumed to be non-overlapping, i.e., nodes do not appear in 2 different subsystems. The vectors and matrices in (4.1) and (4.2) are constructed from the individual subsystem states and describe the ways in which these subsystem states and inputs interact with each other. The state $\boldsymbol{x}(k) = [\boldsymbol{x}_1^{\mathrm{T}}(k), \boldsymbol{x}_2^{\mathrm{T}}(k), \ldots, \boldsymbol{x}_n^{\mathrm{T}}(k)]^{\mathrm{T}}$, where $\boldsymbol{x}_a$ denotes the state of subsystem a, inputs $\boldsymbol{u}(k) = [\boldsymbol{u}_1^{\mathrm{T}}(k), \boldsymbol{u}_2^{\mathrm{T}}(k), \ldots, \boldsymbol{u}_n^{\mathrm{T}}(k)]^{\mathrm{T}}$, where $\boldsymbol{u}_a$ are the inputs to subsystem a, disturbances $\boldsymbol{d}(k) = [\boldsymbol{d}_1^{\mathrm{T}}(k), \boldsymbol{d}_2^{\mathrm{T}}(k), \ldots, \boldsymbol{d}_n^{\mathrm{T}}(k)]^{\mathrm{T}}$, where $\boldsymbol{d}_a$ are the disturbances that affect subsystem a, and outputs $\boldsymbol{y}(k) = [\boldsymbol{y}_1^{\mathrm{T}}(k), \boldsymbol{y}_2^{\mathrm{T}}(k), \ldots, \boldsymbol{y}_n^{\mathrm{T}}(k)]^{\mathrm{T}}$,

where $\boldsymbol{y}_a$ are the outputs from subsystem a. The state space matrices are constructed as follows:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_{12} & \dots & \boldsymbol{A}_{1n} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_2 & \dots & \boldsymbol{A}_{2n} \\ \vdots & & \ddots & \\ \boldsymbol{A}_{n1} & \boldsymbol{A}_{n2} & \dots & \boldsymbol{A}_n \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_{12} & \dots & \boldsymbol{B}_{1n} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_2 & \dots & \boldsymbol{B}_{2n} \\ \vdots & & \ddots & \\ \boldsymbol{B}_{n1} & \boldsymbol{B}_{n2} & \dots & \boldsymbol{B}_n \end{pmatrix},$$
$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{D}_1 & \boldsymbol{D}_{12} & \dots & \boldsymbol{D}_{1n} \\ \boldsymbol{D}_{21} & \boldsymbol{D}_2 & \dots & \boldsymbol{D}_{2n} \\ \vdots & & \ddots & \\ \boldsymbol{D}_{n1} & \boldsymbol{D}_{n2} & \dots & \boldsymbol{D}_n \end{pmatrix}, \boldsymbol{C} = \begin{pmatrix} \boldsymbol{C}_1 & & & \\ & \boldsymbol{C}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{C}_n \end{pmatrix}$$

where $\boldsymbol{A}_i$ is the matrix that describes how $\boldsymbol{x}_i(k)$ affects $\boldsymbol{x}_i(k+1)$, and $\boldsymbol{A}_{ij}$ is the matrix that describes how $\boldsymbol{x}_j(k)$ affects $\boldsymbol{x}_i(k+1)$. The same terminology applies to the subsystem matrices that form $\boldsymbol{B}$, $\boldsymbol{D}$, and $\boldsymbol{C}$, where the subsystem matrices describe the dynamic effects of different subsystems inputs, disturbances, and states on the different subsystems' states and inputs. The diagonal elements of each of the above matrices describe the effect of the local states and inputs on each of the subsystems. The off diagonal elements describe the effect of non-local states on each of the subsystems. Typically in large multi-agent networks the agent responsible for the control of each subsystem measures and controls its local variables directly. In a completely decentralised control system where agents do not communicate with each other, it is only the local dynamics and states that are used to determine control inputs to the system. In distributed systems, where agents receive information on non-local states, it is possible to use the off diagonal matrices and the associated non-local states to improve each agent's control. Each individual subsystem's state-space can then be modelled by the following equation:

$$\boldsymbol{x}_a(k+1) = \boldsymbol{A}_a\boldsymbol{x}_a(k) + \boldsymbol{B}_a\boldsymbol{u}_a(k) + \boldsymbol{D}_a\boldsymbol{d}_a(k) + \boldsymbol{V}_a\boldsymbol{v}_a(k) \tag{4.3}$$

$$\boldsymbol{y}_a(k) = \boldsymbol{C}_a\boldsymbol{x}_a(k). \tag{4.4}$$

The vector $\boldsymbol{v}_a$ is used to group together non-local state variables affecting subsystem a and matrix $\boldsymbol{V}_a$ shows how $\boldsymbol{v}_a$ affects $\boldsymbol{x}_a$. Before proceeding further some more variables that will be used in the formulation of the distributed MPC problem will be defined.

Let there be a set of agents, with indices $j \in \mathcal{N}_a$, that are connected to agent a. The interconnecting input vector, $\boldsymbol{w}_{ja}^{\text{in}}$, is defined as the vector of inputs to control problem a

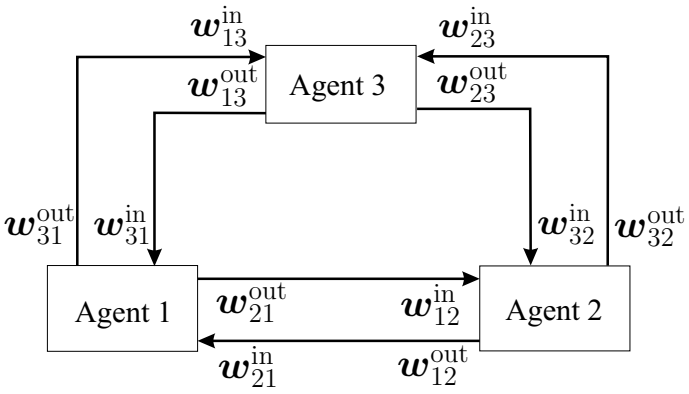


Figure 4.1: Interconnecting inputs and outputs for 3 connected subsystems.

from agent j and the interconnecting output vector $\boldsymbol{w}_{ja}^{\text{out}}$ is defined as the vector of outputs to control problem j from agent a, as in Fig. 4.1. The vector of all interconnecting inputs $\boldsymbol{w}_a^{\text{in}}(k)$, and all interconnecting outputs $\boldsymbol{w}_a^{\text{out}}(k)$ to agent a are typically defined as follows:

$$\begin{aligned}\boldsymbol{w}_a^{\text{in}}(k) &= \left[\boldsymbol{w}_{\mathcal{N}_a^{\text{in}}\{1\}a}^{\text{inT}}(k)\dots\boldsymbol{w}_{\mathcal{N}_a^{\text{in}}\{m_a\}a}^{\text{inT}}(k)\right]^{\text{T}} = \boldsymbol{v}_a(k),\\ \boldsymbol{w}_a^{\text{out}}(k) &= \left[\boldsymbol{w}_{\mathcal{N}_a^{\text{out}}\{1\}a}^{\text{outT}}(k)\dots\boldsymbol{w}_{\mathcal{N}_a^{\text{out}}\{q_a\}a}^{\text{outT}}(k)\right]^{\text{T}} = \boldsymbol{K}_a^{\text{out}}\boldsymbol{x}_a(k),\end{aligned} \tag{4.5}$$

where $\mathcal{N}_a^{\text{in}}$ is the set of agents connected to agent a by an interconnecting input and $\mathcal{N}_a^{\text{in}}\{i\}$ denotes the i^{th} element of this set of agents, e.g., if agent j is the first agent in $\mathcal{N}_a^{\text{in}}$ connected to agent a, then $\boldsymbol{w}_{\mathcal{N}_a^{\text{in}}\{1\}a}^{\text{in}}=\boldsymbol{w}_{ja}^{\text{in}}$. Similarly, $\mathcal{N}_a^{\text{out}}$ is the set of agents connected to agent a by an interconnecting output and $\mathcal{N}_a^{\text{out}}\{i\}$ denotes the i^{th} element of this set of agents. There are m_a agents connected to agent a by an interconnecting input, q_a agents are connected to agent a by an interconnecting output, and $\boldsymbol{K}_a^{\text{out}}$ is a matrix of zeros, with entries of 1 used in the positions that pick out the states in $\boldsymbol{x}_a(k)$ that connect agent a to other subnetworks.

Using (4.3) and (4.4) incremental forms of each subsystem's model can be constructed. These incremental models can then be used to perform state-space based predictions using the same methodology used in the previous chapter. For the calculation of each agent's local optimal control inputs, agents must formulate ways of reaching consensus on the values of their interconnecting variables over the full predictions horizon, as the values of these interconnecting variables are dependent on the dynamics of other connected subsystems. The distributed MPC algorithm is implemented by reformulating the centralised MPC problem as the sum of the minimisations of the local cost functions with equality constraints placed on the interconnecting variables between subsystems over the full prediction horizon. Incremental subsystem models are used for predictions. For a system of

n subsystems, this centralised MPC problem can be stated as follows:

$$\boldsymbol{\vartheta}(k) = \arg\min_{\boldsymbol{\vartheta}} \sum_{a=1}^{n} J_a^{\text{local}}(k), \tag{4.6}$$

subject to the following equality constraints over the prediction horizon,

$$\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k) = \tilde{\boldsymbol{w}}_{aj}^{\text{out}}(k), \text{ for } j \in \mathcal{N}_a, \text{ and } a = 1, \ldots, n. \tag{4.7}$$

Here, $\boldsymbol{\vartheta}(k)$=[$\Delta\tilde{\boldsymbol{u}}^{\text{T}}(k)$,$\Delta\tilde{\boldsymbol{w}}^{\text{inT}}(k)$]$^{\text{T}}$, where $\Delta\tilde{\boldsymbol{w}}^{\text{inT}}(k)$= [$\Delta\tilde{\boldsymbol{w}}_1^{\text{inT}}(k)$,...,$\Delta\tilde{\boldsymbol{w}}_n^{\text{inT}}(k)$]. The local control goals for subsystem a are given by $J_a^{\text{local}}(k,l) = \sum_{p=1}^{N} J_a^{\text{stage}}(k,l,p)$. The stage cost, $J_a^{\text{stage}}(k,l,p)$, of agent a, is given by:

$$J_a^{\text{stage}}(k,l,p) = \boldsymbol{e}_a^{\text{T}}(k+p+1,l)\boldsymbol{Q}_a\boldsymbol{e}_a(k+p+1,l) + \Delta\boldsymbol{u}_a^{\text{T}}(k+p,l)\boldsymbol{R}_a\Delta\boldsymbol{u}_a(k+p,l), \tag{4.8}$$

where $\boldsymbol{Q}_a$ and $\boldsymbol{R}_a$ are the agents a's local MPC weights, and $\boldsymbol{e}_a(k+p,l)$ is the vector of errors in the MPC problem at the p^{th} stage of the prediction horizon, at iteration l of the distributed MPC cycle during sample time k. The error, $\boldsymbol{e}_a(k+p,l) = \boldsymbol{y}_a(k+p,l) - \boldsymbol{r}_a(k+p)$, where $\boldsymbol{y}_a(k+p,l)$ are subsystem a's outputs and $\boldsymbol{r}_a(k+p)$ is a vector of subsystem a's setpoints for sample step $k+p$.

The equality constraint seeks to ensure that all interconnecting variables are made equal to each other over the prediction horizon according to the dynamics of each subsystem, as given in (4.3). This centralised MPC directly manipulates vectors $\Delta\tilde{\boldsymbol{w}}_1^{\text{in}}(k)$,...,$\Delta\tilde{\boldsymbol{w}}_n^{\text{in}}(k)$ in order to satisfy the equality constraints. For convenience, in this thesis reference is made to $\Delta\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k)$ being manipulated when optimising interconnecting variables. However, the first element of $\Delta\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k)$, $\Delta\boldsymbol{w}_{ja}^{\text{in}}(k)$, is the measurement of the interconnecting input that agent a receives from agent j at sample time k. Therefore, when optimising for $\Delta\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k)$, it is elements $\Delta\boldsymbol{w}_{ja}^{\text{in}}(k+1|k)\ldots\Delta\boldsymbol{w}_{ja}^{\text{in}}(k+N-1|k)$ that are optimised for, where $x(k+i|k)$ denotes the predicted value of x at step i of the prediction horizon at sample step k. Variables $\tilde{\boldsymbol{w}}_{aj}^{\text{out}}(k)$ for $j \in \mathcal{N}_a$ and for $a = 1, \ldots, n$ rely on the manipulation of both the inputs and interconnecting inputs.

Augmented Lagrangians can be used to incorporate equality and inequality constraints into a cost function. This results in an unconstrained cost function, which is then solved over a number of iterations in order to solve the original constrained optimisation problem. At each iteration l of the augmented Lagrangian optimisation, Lagrange multipliers $\boldsymbol{\lambda}$ are updated and these denote to what degree the constraints are being satisfied. The

augmented Lagrangian iterations terminate when Lagrange multipliers between 2 iterations agree to within a small tolerance ϵ. The algorithm is described in full in Appendix A. An augmented Lagrangian formulation can be developed from (4.6) to incorporate the equality constraints (4.7) into the cost function. At each sample time k, a number of augmented Lagrangian iterations are run to convergence in order to solve the MPC optimisation problem. The problem solved at each augmented Lagrangian iteration l, during sample time k, is given as follows:

$$\boldsymbol{\vartheta}(k,l) = \arg\min_{\boldsymbol{\vartheta}} \sum_{a=1}^{n} \Bigg(J_a^{\text{local}}(k,l) + \sum_{j \in \mathcal{N}_a} \Big(\tilde{\boldsymbol{\lambda}}_{ja}(k,l)(\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k,l) - \tilde{\boldsymbol{w}}_{aj}^{\text{out}}(k,l)) + \frac{c}{2} ||\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k,l) - \tilde{\boldsymbol{w}}_{aj}^{\text{out}}(k,l)||_2^2 \Big) \Bigg), \tag{4.9}$$

where $\tilde{\boldsymbol{\lambda}}_{ja}(k,l)$ is the vector of Lagrange multipliers associated with the equality constraints placed on the variables connecting agents a and j, $\boldsymbol{\vartheta}(k,l)$ is the vector of inputs and interconnecting inputs to be optimised, $\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k,l)$ are the interconnecting inputs to agent a from agent j, $\tilde{\boldsymbol{w}}_{aj}^{\text{out}}(k,l)$ are the interconnecting outputs from agent a to agent j, and c is a positive constant.

The augmented Lagrangian combines a quadratic penalty on equality constraint violations with successive estimates of the Lagrange multipliers. In other quadratic penalty methods, where these Lagrange multiplier estimates are not used, the value of c is increased over a number of iterations to ensure the satisfaction of constraints. However, this can cause ill-conditioning of the problem. It has been shown that, with the use of Augmented Lagrangian formulation, c can be kept constant, and so avoids this ill-conditioning problem (Nocedal and Wright, 2006). The algorithm, and other penalty methods for solving optimisation problems with equality and inequality constraints, are described in greater detail in Appendix A.

The quadratic terms of the augmented Lagrangian formulation are distributed across the agents using the Alternating Direction Method of Multipliers (Tosserams, 2008; Bertsekas and Tsitsikilis, 1989). This method will now be introduced before showing how it is used to distribute the centralised MPC problem in (4.9) amongst the agents.

4.2.1 Alternating Direction Method of Multipliers (ADMOM)

The ADMOM is derived from the application of Block Coordinate Descent (BCD) (also called the nonlinear Gauss-Seidel method) to the augmented Lagrangian formulation of a problem. BCD is a serial decomposition method for optimisation problems. Using BCD the problem:

$$\boldsymbol{x} = [\boldsymbol{x}_1^{\mathrm{T}}, \ldots, \boldsymbol{x}_n^{\mathrm{T}}]^{\mathrm{T}} = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \begin{cases} \kappa_i(\boldsymbol{x}) \geq 0, & \forall\ i \in \mathcal{I} \\ \kappa_j(\boldsymbol{x}) = 0, & \forall\ j \in \mathcal{E} \end{cases} \tag{4.10}$$

can be decomposed into a number of smaller optimisation problems, based on sub-vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, which are solved sequentially in an iterative fashion. The optimisation for sub-vector $\boldsymbol{x}_i(p)$ at iteration p of the overall optimisation procedure becomes

$$\begin{aligned} &\boldsymbol{x}_i(p) = \arg\min_{\boldsymbol{x}_i} f([\boldsymbol{x}_1^{\mathrm{T}}(p-1), \ldots, \boldsymbol{x}_i^{\mathrm{T}}(p), \ldots, \boldsymbol{x}_n^{\mathrm{T}}(p-1)]^{\mathrm{T}}) \\ &\text{subject to} \begin{cases} \kappa_i([\boldsymbol{x}_1^{\mathrm{T}}(p-1), \ldots, \boldsymbol{x}_i^{\mathrm{T}}(p), \ldots, \boldsymbol{x}_n^{\mathrm{T}}(p-1)]^{\mathrm{T}}) \geq 0, & \forall\ i \in \mathcal{I}, \\ \kappa_j([\boldsymbol{x}_1^{\mathrm{T}}(p-1), \ldots, \boldsymbol{x}_i^{\mathrm{T}}(p), \ldots, \boldsymbol{x}_n^{\mathrm{T}}(p-1)]^{\mathrm{T}}) = 0, & \forall\ j \in \mathcal{E}. \end{cases} \end{aligned} \tag{4.11}$$

It is shown in (Bertsekas and Tsitsikilis, 1989) that if $f(\boldsymbol{x})$, $\kappa_i(\boldsymbol{x})\ \forall\ i \in \mathcal{I}$, and $\kappa_j(\boldsymbol{x})\ \forall\ j \in \mathcal{E}$ are all convex functions, that over a finite number of iterations the answer given by iteratively solving for each $\boldsymbol{x}_i$ as in (4.11), for $i = 1, \ldots, n$, converges to the solution given by (4.10).

It is possible to apply BCD to augmented Lagrangian problems. This is done by having an inner and outer loop of the augmented Lagrangian problem. The outer loop is the Lagrange multiplier update loop. This Lagrange multiplier is kept constant for the inner loop. The inner loop then applies BCD to the unconstrained Lagrangian function. This is achieved by dividing the main optimisation vector $\boldsymbol{x}$ into sub-vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and solving the Lagrangian function as shown above in (4.11). However, these inner loop BCD iterations can be computationally costly so inexact formulations of the problem can be used, where inner iterations are terminated before convergence. The most extreme form of this is the ADMOM which uses only one BCD iteration before updating the Lagrange multipliers in the outer loop. This is shown to converge where the cost function being optimised is convex, additively separable, and subject to linear coupling constraints between variables (Tosserams et al., 2008; Bertsekas and Tsitsikilis, 1989). Given that this is the nature of the problem described in (4.6) it is possible therefore to solve this problem using the ADMOM. The following example provides a very simple illustration of how the ADMOM

works. Take the optimisation problem:

$$\begin{aligned} \boldsymbol{x} = [\boldsymbol{x}_1^{\mathrm{T}},\boldsymbol{x}_2^{\mathrm{T}}]^{\mathrm{T}} = \arg\min_{\boldsymbol{x}_1,\boldsymbol{x}_2} \Big(f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_2)\Big) \\ \text{such that } \boldsymbol{x}_1 = \boldsymbol{x}_2. \end{aligned} \tag{4.12}$$

The unconstrained augmented Lagrangian form of this equation at iteration l of the augmented Lagrangian optimisation is given as:

$$\begin{aligned} \boldsymbol{x}(l) = [\boldsymbol{x}_1^{\mathrm{T}}(l), \boldsymbol{x}_2^{\mathrm{T}}(l)]^{\mathrm{T}} = \arg\min_{\boldsymbol{x}_1,\boldsymbol{x}_2} & f_1(\boldsymbol{x}_1(l)) + f_2(\boldsymbol{x}_2(l)) \\ & + \boldsymbol{\lambda}(l)(\boldsymbol{x}_1(l) - \boldsymbol{x}_2(l)) + \frac{c}{2}||\boldsymbol{x}_1(l) - \boldsymbol{x}_2(l)||_2^2, \end{aligned} \tag{4.13}$$

where $\boldsymbol{\lambda}(l)$ are the Lagrange multipliers at iteration l of the augmented Lagrangian optimisation. Using the ADMOM (4.13) is solved in a serial fashion for iteration l of the augmented Lagrangian optimisation as follows:

$$\begin{aligned} \boldsymbol{x}_1(l) &= \arg\min_{\boldsymbol{x}_1(l)} f_1(\boldsymbol{x}_1(l)) + \boldsymbol{\lambda}(l)\boldsymbol{x}_1(l) + \frac{c}{2}||\boldsymbol{x}_1(l) - \boldsymbol{x}_2(l-1)||_2^2 \\ \boldsymbol{x}_2(l) &= \arg\min_{\boldsymbol{x}_2(l)} f_2(\boldsymbol{x}_2(l)) - \boldsymbol{\lambda}(l)\boldsymbol{x}_2(l) + \frac{c}{2}||\boldsymbol{x}_1(l) - \boldsymbol{x}_2(l)||_2^2. \end{aligned} \tag{4.14}$$

At each iteration of the augmented Lagrangian optimisation problem (4.14) is solved for subvectors $\boldsymbol{x}_1(l)$ and $\boldsymbol{x}_2(l)$. The optimisation for subvector $\boldsymbol{x}_1(l)$ is carried out first using the most recent update of subvector $\boldsymbol{x}_2(l)$, which is $\boldsymbol{x}_2(l-1)$. Then the optimisation for subvector $\boldsymbol{x}_2(l)$ uses the most recent update of subvector $\boldsymbol{x}_1(l)$, which is $\boldsymbol{x}_1(l)$ (as $\boldsymbol{x}_1(l)$'s optimisation has already been carried out in iteration l). Lagrange multiplier updates followed by evaluations of (4.14) are then carried out in an iterative fashion until the Lagrange multipliers converge.

4.2.2 The distributed MPC algorithm

The use of ADMOM on (4.6) leads to the iterative distributed MPC algorithm. Each distributed MPC cycle consists of an outer loop where Lagrange multiplier updates are carried out and an inner loop where agents perform one optimisation each. It may take a number of outer loop Lagrange multiplier iterations in order for the distributed MPC algorithm to terminate in a given sample step. A distributed MPC iteration consists of an outer loop multiplier update and the internal loop optimisations carried out by each agent. The number of the distributed MPC iterations that have occurred during a sample step k will be denoted by l (which is also the number of Lagrangian multiplier updates

that have occured). In this approach, one agent at a time optimises values for its inputs, $\Delta\tilde{\boldsymbol{u}}_a(k,l)$, and its desired interconnecting input variables $\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k,l)$, for each $j \in \mathcal{N}_a$, in order to reach consensus on values for the interconnecting variables over the full prediction horizon.

The optimisation problem of agent a, for iteration l of the distributed MPC cycle, at sample step k is:

$$\boldsymbol{\vartheta}_a(k,l) = \arg\min_{\boldsymbol{\vartheta}_a} \left(J_a^{\text{local}}(k) + J_a^{\text{inter}}(k,l) \right), \tag{4.15}$$

where $\boldsymbol{\vartheta}_a(k,l)=[\Delta\tilde{\boldsymbol{u}}_a^{\text{T}}(k,l), \Delta\tilde{\boldsymbol{w}}_a^{\text{inT}}(k,l)]^{\text{T}}$, and $J_a^{\text{inter}}(k,l)$ is the interconnection cost for agent a, given by:

$$J_a^{\text{inter}}(k,l) = \sum_{j \in \mathcal{N}_a} J_{ja}^{\text{inter}}(k,l), \tag{4.16}$$

and $J_{ja}^{\text{inter}}(k,l)$ is the cost associated with the inter-agent coordination with agent j given by:

$$\begin{aligned} J_{ja}^{\text{inter}}(k,l) = &\begin{bmatrix} \tilde{\boldsymbol{\lambda}}_{ja}^{\text{in}}(k,l) \\ -\tilde{\boldsymbol{\lambda}}_{aj}^{\text{in}}(k,l) \end{bmatrix}^{\text{T}} \begin{bmatrix} \tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k,l) \\ \tilde{\boldsymbol{w}}_{ja}^{\text{out}}(k,l) \end{bmatrix} \\ &+ \frac{c}{2} \left\| \begin{bmatrix} \tilde{\boldsymbol{w}}_{aj,\text{prev}}^{\text{in}}(k,l) - \tilde{\boldsymbol{w}}_{ja}^{\text{out}}(k,l) \\ \tilde{\boldsymbol{w}}_{aj,\text{prev}}^{\text{out}}(k,l) - \tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k,l) \end{bmatrix} \right\|_2^2, \end{aligned} \tag{4.17}$$

where $\tilde{\boldsymbol{\lambda}}_{ja}^{\text{in}}(k,l)$ are the Lagrange multipliers associated with the interconnecting constraints $\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k,l) = \tilde{\boldsymbol{w}}_{aj}^{\text{out}}(k,l)$ at iteration l, and sample step k.

Each agent optimises this cost in a serial fashion, communicating the interconnecting variables with its neighbours. The values $\tilde{\boldsymbol{w}}_{aj,\text{prev}}^{\text{out}}(k,l)$ and $\tilde{\boldsymbol{w}}_{aj,\text{prev}}^{\text{in}}(k,l)$ are taken as the most recently updated values of $\tilde{\boldsymbol{w}}_{aj}^{\text{out}}(k,l)$ and $\tilde{\boldsymbol{w}}_{aj}^{\text{in}}(k,l)$, respectively. When each agent has performed one optimisation in the inner loop of the distributed MPC cycle the Lagrange multipliers are updated as follows in the outer loop:

$$\tilde{\boldsymbol{\lambda}}_{ja}^{\text{in}}(k,l+1) = \tilde{\boldsymbol{\lambda}}_{ja}^{\text{in}}(k,l) + c\left(\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k,l) - \tilde{\boldsymbol{w}}_{aj}^{\text{out}}(k,l)\right), \tag{4.18}$$

The distributed MPC iterations terminate when:

$$\begin{aligned} ||\tilde{\boldsymbol{\lambda}}_{ja}^{\text{in}}(k,l+1) - \tilde{\boldsymbol{\lambda}}_{ja}^{\text{in}}(k,l)||_\infty \leq \epsilon \\ \text{for } a = 1,\ldots,n \text{ and for all } j \in \mathcal{N}_a, \end{aligned} \tag{4.19}$$

where ϵ is a specified tolerance and $\|.\|_\infty$ denotes the infinity norm. A flow chart representation of the distributed MPC algorithm at sample time k can be seen in Fig. 4.2.

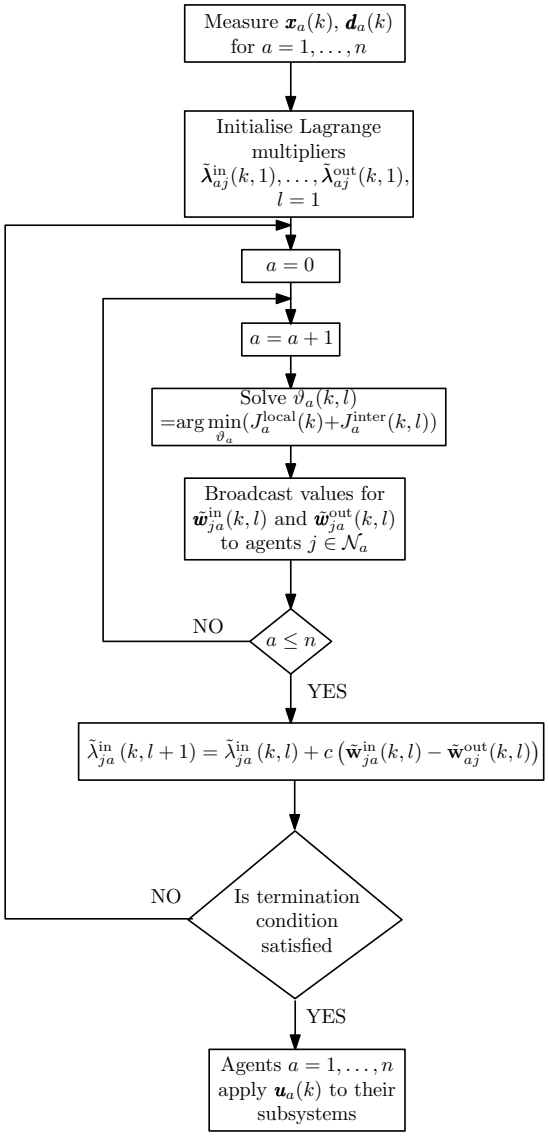


Figure 4.2: The distributed MPC algorithm, at the k^{th} sample step.

The c and ϵ parameters determine the importance that each agent gives to achieving consensus with other connected agents versus fulfilling their local cost function objectives. The tuning of the c and ϵ parameters of the distributed MPC, and the $\boldsymbol{Q}_a$ and $\boldsymbol{R}_a$ parameters associated with each agent a's local cost function, given by (4.8), significantly impacts on the closed-loop performance of the system and on the amount of communication used by the distributed MPC scheme to achieve this control. It is worth noting that this algorithm only requires local communication between agents, which makes the algorithm scalable, and hence suitable for use with large scale systems such as power networks. However, as has been discussed previously, this results in the reaching of Nash rather than Pareto equilibria. Particularly in cases where systems are highly interconnected, these Nash equi-

librium seeking distributed MPCs can result in instability, whereas Pareto equilibrium distributed MPC algorithms would remain stable (Venkat, 2006). It is therefore useful to be able to determine a-priori whether the distributed MPC will give stable control. Here a novel stability proof is developed for the unconstrained case of the distributed MPC algorithm.

4.3 Convergence and stability of unconstrained linear distributed MPC

As was stated in the previous section the ADMOM converges to the centralised MPC problem (4.9), as it is constructed from the addition of convex functions and is only subject to linear equality constraints. Therefore, the stability of the solutions given by (4.9) implies the stability of the distributed MPC. The unconstrained augmented Lagrangian centralised MPC problem that the distributed MPC solves in an iterative fashion is:

$$\begin{aligned}\boldsymbol{\vartheta}(k,l) = \arg\min_{\boldsymbol{\vartheta}} \sum_{a=1}^{n} \Bigg(J_a^{\text{local}}(k,l) + \sum_{j\in\mathcal{N}_a} \Bigg(\tilde{\boldsymbol{\lambda}}_{ja}^{\text{T}}(k,l)(\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k,l) - \tilde{\boldsymbol{w}}_{aj}^{\text{out}}(k,l)) \\ + \frac{c}{2}||\tilde{\boldsymbol{w}}_{ja}^{\text{in}}(k,l) - \tilde{\boldsymbol{w}}_{aj}^{\text{out}}(k,l)||_2^2 \Bigg) \Bigg) \\ = \arg\min_{\boldsymbol{\vartheta}} J^{\text{local}}(k,l) + \tilde{\boldsymbol{\lambda}}^{\text{T}}(k,l)(\tilde{\boldsymbol{w}}^{\text{in}}(k,l) - \tilde{\boldsymbol{w}}^{\text{out}}(k,l)) + \frac{c}{2}||\tilde{\boldsymbol{w}}^{\text{in}}(k,l) - \tilde{\boldsymbol{w}}^{\text{out}}(k,l)||_2^2.\end{aligned} \tag{4.20}$$

where $\boldsymbol{\vartheta}(k,l)=[\Delta\boldsymbol{u}^{\text{T}}(k,l),\Delta\tilde{\boldsymbol{w}}^{\text{inT}}(k,l)]^{\text{T}}$. The interconnecting inputs and outputs and the Lagrange multipliers are grouped into the terms $\tilde{\boldsymbol{w}}^{\text{in}}(k,l)$, $\tilde{\boldsymbol{w}}^{\text{out}}(k,l)$, and $\tilde{\boldsymbol{\lambda}}(k,l)$, respectively and are given as follows:

$$\tilde{\boldsymbol{w}}^{\text{in}}(k,l)=\begin{bmatrix}\tilde{\boldsymbol{w}}^{\text{in}}_{\mathcal{N}_1^{\text{in}}\{1\}1}(k,l)\\ \tilde{\boldsymbol{w}}^{\text{in}}_{\mathcal{N}_1^{\text{in}}\{2\}1}(k,l)\\ \vdots\\ \tilde{\boldsymbol{w}}^{\text{in}}_{\mathcal{N}_1^{\text{in}}\{m_1\}1}(k,l)\\ \cdots\cdots\\ \vdots\\ \cdots\cdots\\ \tilde{\boldsymbol{w}}^{\text{in}}_{\mathcal{N}_n^{\text{in}}\{1\}n}(k,l)\\ \tilde{\boldsymbol{w}}^{\text{in}}_{\mathcal{N}_n^{\text{in}}\{2\}n}(k,l)\\ \vdots\\ \tilde{\boldsymbol{w}}^{\text{in}}_{\mathcal{N}_n^{\text{in}}\{m_n\}n}(k,l)\end{bmatrix}, \tilde{\boldsymbol{w}}^{\text{out}}(k,l)=\begin{bmatrix}\tilde{\boldsymbol{w}}^{\text{out}}_{\mathcal{N}_1^{\text{out}}\{1\}1}(k,l)\\ \tilde{\boldsymbol{w}}^{\text{out}}_{\mathcal{N}_1^{\text{out}}\{2\}1}(k,l)\\ \vdots\\ \tilde{\boldsymbol{w}}^{\text{out}}_{\mathcal{N}_1^{\text{out}}\{q_1\}1}(k,l)\\ \cdots\cdots\\ \vdots\\ \cdots\cdots\\ \tilde{\boldsymbol{w}}^{\text{out}}_{\mathcal{N}_n^{\text{out}}\{1\}n}(k,l)\\ \tilde{\boldsymbol{w}}^{\text{out}}_{\mathcal{N}_n^{\text{out}}\{2\}n}(k,l)\\ \vdots\\ \tilde{\boldsymbol{w}}^{\text{out}}_{\mathcal{N}_n^{\text{out}}\{q_n\}n}(k,l)\end{bmatrix}, \tilde{\boldsymbol{\lambda}}(k,l)=\begin{bmatrix}\tilde{\boldsymbol{\lambda}}^{\text{in}}_{\mathcal{N}_1^{\text{in}}\{1\}1}(k,l)\\ \tilde{\boldsymbol{\lambda}}^{\text{in}}_{\mathcal{N}_1^{\text{in}}\{1\}2}(k,l)\\ \vdots\\ \tilde{\boldsymbol{\lambda}}^{\text{out}}_{\mathcal{N}_1^{\text{in}}\{m_1\}1}(k,l)\\ \cdots\cdots\\ \vdots\\ \cdots\cdots\\ \tilde{\boldsymbol{\lambda}}^{\text{in}}_{\mathcal{N}_n^{\text{in}}\{1\}n}(k,l)\\ \tilde{\boldsymbol{\lambda}}^{\text{in}}_{\mathcal{N}_n^{\text{in}}\{2\}n}(k,l)\\ \vdots\\ \tilde{\boldsymbol{\lambda}}^{\text{in}}_{\mathcal{N}_n^{\text{in}}\{m_n\}n}(k,l)\end{bmatrix},$$

where agent a has m_a interconnecting inputs and q_a interconnecting outputs.

The term $J^{\text{local}}(k)=\sum_{a=1}^{n}J_a^{\text{local}}(k,l)$. Using the incremental form of the model given in (4.3) for predictions, $J^{\text{local}}(k,l)$ is formed as follows:

$$J^{\text{local}}(k,l)=\left(\tilde{\boldsymbol{y}}(k+1,l)-\tilde{\boldsymbol{r}}(k+1)\right)^{\text{T}}\boldsymbol{Q}\left(\tilde{\boldsymbol{y}}(k+1,l)-\tilde{\boldsymbol{r}}(k+1)\right)+\Delta\tilde{\boldsymbol{u}}^{\text{T}}(k,l)\boldsymbol{R}\Delta\tilde{\boldsymbol{u}}(k,l) \tag{4.21}$$

where $\tilde{\boldsymbol{r}}(k)=[\tilde{\boldsymbol{r}}_1^{\text{T}}(k),\ldots,\tilde{\boldsymbol{r}}_n^{\text{T}}(k)]^{\text{T}}$, $\boldsymbol{Q}=\text{diag}(\boldsymbol{Q}_1,\ldots,\boldsymbol{Q}_n)$, $\boldsymbol{R}=\text{diag}(\boldsymbol{R}_1,\ \ldots,\ \boldsymbol{R}_n)$, and $\tilde{\boldsymbol{y}}(k+1,l)=\boldsymbol{K}_c\tilde{\boldsymbol{x}}^{\text{aug}}(k+1,l)=\boldsymbol{K}_c(\hat{\boldsymbol{A}}^{\text{f}}\boldsymbol{x}^{\text{aug}}(k)+\hat{\boldsymbol{B}}^{\text{f}}\Delta\tilde{\boldsymbol{u}}(k,l)+\hat{\boldsymbol{V}}^{\text{f}}[\Delta\boldsymbol{v}^{\text{T}}(k),\Delta\tilde{\boldsymbol{w}}^{\text{inT}}(k,l)]^{\text{T}})$ (the disturbance model is omitted here but this can easily be included in exactly the same way as the other matrices). Here, $\boldsymbol{x}^{\text{aug}}(k)=[\Delta\boldsymbol{x}(k)^{\text{T}},\boldsymbol{x}(k)^{\text{T}}]^{\text{T}}$ is the augmented state space matrix, and $\hat{\boldsymbol{A}}^{\text{f}}$, $\hat{\boldsymbol{B}}^{\text{f}}$, and $\hat{\boldsymbol{V}}^{\text{f}}$ are the prediction matrices associated with $\boldsymbol{x}^{\text{aug}}(k)$, $\Delta\tilde{\boldsymbol{u}}(k,l)$, and $\Delta\tilde{\boldsymbol{w}}^{\text{in}}(k,l)$, respectively. The state of the entire system $\boldsymbol{x}^{\text{aug}}(k)=[\boldsymbol{x}_1^{\text{augT}}(k),\ldots,\boldsymbol{x}_n^{\text{augT}}(k)]^{\text{T}}$, the system's predicted incremental inputs $\Delta\tilde{\boldsymbol{u}}(k,l)=[\Delta\tilde{\boldsymbol{u}}_1^{\text{T}}(k,l),\ldots,\Delta\tilde{\boldsymbol{u}}_n^{\text{T}}(k,l)]^{\text{T}}$, the system's incremental interconnecting inputs $\Delta\boldsymbol{v}(k)=[\Delta\boldsymbol{v}_1^{\text{T}}(k),\ldots,\Delta\boldsymbol{v}_n^{\text{T}}(k)]^{\text{T}}$, and the system's predicted incremental interconnecting inputs $\Delta\tilde{\boldsymbol{w}}^{\text{in}}(k,l)=[\Delta\tilde{\boldsymbol{w}}_1^{\text{inT}}(k,l),\ldots,$ $\Delta\tilde{\boldsymbol{w}}_n^{\text{inT}}(k,l)]^{\text{T}}$, and their associated predictive matrices $\hat{\boldsymbol{A}}^{\text{f}}=\text{diag}(\hat{\boldsymbol{A}}_1^{\text{f}},\ldots,\hat{\boldsymbol{A}}_n^{\text{f}})$, $\hat{\boldsymbol{B}}^{\text{f}}=$ $\text{diag}(\hat{\boldsymbol{B}}_1^{\text{f}},\ldots,\hat{\boldsymbol{B}}_n^{\text{f}})$, and $\hat{\boldsymbol{V}}^{\text{f}}=\text{diag}(\hat{\boldsymbol{V}}_1^{\text{f}},\ldots,\hat{\boldsymbol{V}}_n^{\text{f}})$. The way in which matrices $\hat{\boldsymbol{A}}_i^{\text{f}}$, $\hat{\boldsymbol{B}}_i^{\text{f}}$, and $\hat{\boldsymbol{V}}_i^{\text{f}}$, for $i=1,\ldots,n$, are derived for each subsystem is similar to the way in which the augmented state-space prediction matrices were formed in in Section 3.1.2, previously. Here, $\boldsymbol{K}_c$ is a matrix of zeroes, with entries of 1 in the positions that pick the outputs $\tilde{\boldsymbol{y}}(k+1,l)$ from the augmented state prediction vector $\tilde{\boldsymbol{x}}^{\text{aug}}(k,l)$.

Variables in (4.20) are grouped together to enable the problem to be posed in terms of $\boldsymbol{\vartheta}(k,l)$ as follows:

- Let $\tilde{\boldsymbol{y}}(k+1) = \boldsymbol{K}_c\boldsymbol{D}\boldsymbol{s}(k) + \boldsymbol{K}_c\boldsymbol{Z}\boldsymbol{\vartheta}(k,l)$. The matrix $\hat{\boldsymbol{V}}^{\mathrm{f}} = [\hat{\boldsymbol{V}}^{\mathrm{f}}_{\boldsymbol{v}}, \hat{\boldsymbol{V}}^{\mathrm{f}}_{\boldsymbol{w}^{\mathrm{in}}}]$. Here, $\hat{\boldsymbol{V}}^{\mathrm{f}}_{\boldsymbol{v}}$ determines the effect of $\Delta\boldsymbol{v}(k)$ on $\tilde{\boldsymbol{x}}^{\mathrm{aug}}(k+1,l)$. The matrix $\hat{\boldsymbol{V}}^{\mathrm{f}}_{\boldsymbol{w}^{\mathrm{in}}}$ similarly determines the effect of $\Delta\tilde{\boldsymbol{w}}^{\mathrm{in}}(k,l)$ on $\tilde{\boldsymbol{x}}^{\mathrm{aug}}(k+1,l)$. The matrix $\boldsymbol{D} = [\hat{\boldsymbol{A}}^{\mathrm{f}}, \hat{\boldsymbol{V}}^{\mathrm{f}}_{\boldsymbol{v}}]$ and vector $\boldsymbol{s}(k) = [\boldsymbol{x}^{\mathrm{aug}\mathrm{T}}(k), \Delta\boldsymbol{v}^{\mathrm{T}}(k)]^{\mathrm{T}}$. Note that $\boldsymbol{K}_c\boldsymbol{D}\boldsymbol{s}(k)$ is fixed during the MPC iterations at each sample step. The latter group of terms $\boldsymbol{K}_c\boldsymbol{Z}\boldsymbol{\vartheta}(k,l)$ varies, via the manipulation of $\boldsymbol{\vartheta}(k,l)$, where $\boldsymbol{Z} = [\hat{\boldsymbol{B}}^{\mathrm{f}}, \hat{\boldsymbol{V}}^{\mathrm{f}}_{\boldsymbol{w}^{\mathrm{in}}}]$.

- The interconnecting inputs are given by $\tilde{\boldsymbol{w}}^{\mathrm{in}}(k,l) = \boldsymbol{K}_v\boldsymbol{s}(k) + \boldsymbol{K}_w\boldsymbol{\vartheta}(k,l)$ where $\boldsymbol{K}_v$ is used to pick out the relevant interconnecting variables from $\boldsymbol{s}(k)$. $\boldsymbol{K}_w$ is then used to sum the incremental values of the interconnecting variables to the current value of the interconnecting variable so as to give the value of the interconnecting variable over the prediction horizon. Take for example, an interconnecting variable v. This will be contained in the full state space vector $\boldsymbol{x}^{\mathrm{aug}}(k)$ which is contained in $\boldsymbol{s}(k)$. In order to pick out this variable, the matrix $\boldsymbol{K}_v$ will have an entry of 1 in the relevant row and column, and the rest of that row will have entries of 0. Then incremental values Δv are picked from $\boldsymbol{\vartheta}(k,l)$ using the $\boldsymbol{K}_w$ matrix. For instance, to compare the value of $v(k+2|k)$ with its corresponding interconnecting output the values v, $\Delta v(k+1|k)$, and $\Delta v(k+2|k)$ would be added together, using $\boldsymbol{K}_v$ to pick out v from $\boldsymbol{s}(k)$ and $\boldsymbol{K}_w$ to pick out $\Delta v(k+1|k)$, and $\Delta v(k+2|k)$ from $\boldsymbol{\vartheta}(k,l)$, where $v(k+i|k)$ is the predicted value of v i steps into the prediction horizon at sample step k.

- The interconnecting outputs are given by $\tilde{\boldsymbol{w}}^{\mathrm{out}}(k,l) = \boldsymbol{K}_I(\boldsymbol{D}\boldsymbol{s}(k) + \boldsymbol{Z}\boldsymbol{\vartheta}(k,l))$. Here, $\boldsymbol{K}_I$ is a matrix of zeroes with entries of one in the positions that pick the interconnecting outputs which correspond to the interconnecting inputs given previously.

- The term $\tilde{\boldsymbol{w}}^{\mathrm{in}}(k,l) - \tilde{\boldsymbol{w}}^{\mathrm{out}}(k,l)$ in the cost function can then be written as:

$$\begin{aligned}\tilde{\boldsymbol{w}}^{\mathrm{in}}(k,l) - \tilde{\boldsymbol{w}}^{\mathrm{out}}(k,l) &= \boldsymbol{K}_v\boldsymbol{s}(k) + \boldsymbol{K}_w\boldsymbol{\vartheta}(k,l) - \boldsymbol{K}_I(\boldsymbol{D}\boldsymbol{s}(k) + \boldsymbol{Z}\boldsymbol{\vartheta}(k,l)) \\ &= (\boldsymbol{K}_v - \boldsymbol{K}_I\boldsymbol{D})\boldsymbol{s}(k) + (\boldsymbol{K}_w - \boldsymbol{K}_I\boldsymbol{Z})\boldsymbol{\vartheta}(k,l) \\ &= \boldsymbol{K}_s\boldsymbol{s}(k) + \boldsymbol{K}_\vartheta\boldsymbol{\vartheta}(k,l)\end{aligned}$$

  where $\boldsymbol{K}_s = \boldsymbol{K}_v - \boldsymbol{K}_I\boldsymbol{D}$ and $\boldsymbol{K}_w = \boldsymbol{K}_w - \boldsymbol{K}_I\boldsymbol{Z}$.

Problem (4.20) can now be stated as:

$$\begin{aligned}\boldsymbol{\vartheta}(k,l) = \arg\min_{\boldsymbol{\vartheta}} &\left(\boldsymbol{P}\boldsymbol{s}\left(k\right) + \boldsymbol{G}\boldsymbol{\vartheta}\left(k,l\right) - \tilde{\boldsymbol{r}}(k+1)\right)^{\mathrm{T}} \boldsymbol{Q}\left(\boldsymbol{P}\boldsymbol{s}\left(k\right) + \boldsymbol{G}\boldsymbol{\vartheta}\left(k,l\right) - \tilde{\boldsymbol{r}}(k+1)\right) \\ &+ \boldsymbol{\vartheta}^{\mathrm{T}}\left(k,l\right)\boldsymbol{R}_{\vartheta}\boldsymbol{\vartheta}\left(k,l\right) + \tilde{\boldsymbol{\lambda}}^{\mathrm{T}}\left(k,l\right)\left(\boldsymbol{K}_{s}\boldsymbol{s}\left(k\right) + \boldsymbol{K}_{\vartheta}\boldsymbol{\vartheta}\left(k,l\right)\right) \\ &+ \frac{c}{2}\left(\boldsymbol{K}_{s}\boldsymbol{s}\left(k\right) + \boldsymbol{K}_{\vartheta}\boldsymbol{\vartheta}\left(k,l\right)\right)^{\mathrm{T}}\left(\boldsymbol{K}_{s}\boldsymbol{s}\left(k\right) + \boldsymbol{K}_{\vartheta}\boldsymbol{\vartheta}\left(k,l\right)\right)\end{aligned} \tag{4.22}$$

where $\boldsymbol{P} = \boldsymbol{K}_{c}\boldsymbol{D}$, $\boldsymbol{G} = \boldsymbol{K}_{c}\boldsymbol{Z}$, and $\boldsymbol{R}_{\vartheta}$=diag$(\boldsymbol{R}, \boldsymbol{0}_{(N-1)n_{v}\times(N-1)n_{v}})$ where n_{v} is the size of $\boldsymbol{v}$. A worked example will be given in Section 4.4 which will show how all the above matrices are constructed for an example system.

Taking (4.22) and rearranging the matrices, it is possible to represent (4.22) in the following quadratic form:

$$\begin{aligned}\boldsymbol{\vartheta}(k,l) &= \arg\min_{\boldsymbol{\vartheta}} J^{\mathrm{quad}}(k,l) \\ &= \arg\min_{\boldsymbol{\vartheta}} \boldsymbol{\vartheta}^{\mathrm{T}}(k,l)\boldsymbol{H}\boldsymbol{\vartheta}(k,l) + \boldsymbol{\vartheta}^{\mathrm{T}}(k,l)\boldsymbol{f}(k,l) + \theta\end{aligned} \tag{4.23}$$

where $J^{\mathrm{quad}}(k,l)$ is the quadratic cost function at sample step k and distributed MPC iteration l. Here, $\boldsymbol{H} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{G} + \boldsymbol{R}_{\vartheta} + \frac{c}{2}\boldsymbol{K}_{\vartheta}^{\mathrm{T}}\boldsymbol{K}_{\vartheta}$, $\boldsymbol{f}(k,l) = 2\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{P}\boldsymbol{s}(k) - 2\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}\tilde{\boldsymbol{r}}(k+1) + \boldsymbol{K}_{\vartheta}^{\mathrm{T}}\tilde{\boldsymbol{\lambda}}(k,l) + c\boldsymbol{K}_{\vartheta}^{\mathrm{T}}\boldsymbol{K}_{s}\boldsymbol{s}(k)$, and finally $\theta = \boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{P}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{P}\boldsymbol{s}(k) - 2\tilde{\boldsymbol{r}}^{\mathrm{T}}(k+1)\boldsymbol{Q}\boldsymbol{P}\boldsymbol{s}(k) + \tilde{\boldsymbol{\lambda}}^{\mathrm{T}}\boldsymbol{K}_{s}\boldsymbol{s}(k) + \frac{c}{2}\boldsymbol{s}^{\mathrm{T}}(k)\boldsymbol{K}_{s}^{\mathrm{T}}\boldsymbol{K}_{s}\boldsymbol{s}(k) + \tilde{\boldsymbol{r}}^{\mathrm{T}}(k+1)\boldsymbol{Q}\tilde{\boldsymbol{r}}(k+1)$, which does not depend on ϑ.

The optimal value of $\boldsymbol{\vartheta}(k,l)$ at sample step k and distributed MPC iteration l, $\boldsymbol{\vartheta}^{*}(k,l)$, is found by setting $\frac{\partial}{\partial\boldsymbol{\vartheta}(k,l)}J^{\mathrm{quad}}(k,l) = 0$, which yields,

$$\begin{aligned}2\boldsymbol{H}\boldsymbol{\vartheta}^{*}(k,l) + \boldsymbol{f}(k,l) &= 0, \\ \boldsymbol{\vartheta}^{*}(k,l) &= -\frac{1}{2}\boldsymbol{H}^{-1}\boldsymbol{f}(k,l).\end{aligned} \tag{4.24}$$

After $\boldsymbol{\vartheta}^{*}(k,l)$ is found, the Lagrange multipliers are calculated as follows:

$$\begin{aligned}\tilde{\boldsymbol{\lambda}}(k,l+1) &= \tilde{\boldsymbol{\lambda}}(k,l) + c(\tilde{\boldsymbol{w}}^{\mathrm{in}}(k,l) - \tilde{\boldsymbol{w}}^{\mathrm{out}}(k,l)) \\ &= \tilde{\boldsymbol{\lambda}}(k,l) + c(\boldsymbol{K}_{s}\boldsymbol{s}(k) + \boldsymbol{K}_{\vartheta}\boldsymbol{\vartheta}^{*}(k,l)) \\ &= \tilde{\boldsymbol{\lambda}}(k,l) + c\boldsymbol{K}_{s}\boldsymbol{s}(k) - \frac{c}{2}\boldsymbol{K}_{\vartheta}\boldsymbol{H}^{-1}(2\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{P}\boldsymbol{s}(k) \\ &\quad - 2\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}\tilde{\boldsymbol{r}}(k+1) + \boldsymbol{K}_{\vartheta}^{\mathrm{T}}\tilde{\boldsymbol{\lambda}}(k,l) + c\boldsymbol{K}_{\vartheta}^{\mathrm{T}}\boldsymbol{K}_{s}\boldsymbol{s}(k)) \\ &= (\mathbf{I}_{n_{\lambda}\times n_{\lambda}} - \frac{c}{2}\boldsymbol{K}_{\vartheta}\boldsymbol{H}^{-1}\boldsymbol{K}_{\vartheta}^{\mathrm{T}})\tilde{\boldsymbol{\lambda}}(k,l) + \boldsymbol{C}_{\lambda}(k)\end{aligned} \tag{4.25}$$

where n_{λ} is the length of vector $\tilde{\boldsymbol{\lambda}}(k,l)$, and $\boldsymbol{C}_{\lambda}(k) = c\boldsymbol{K}_{s}\boldsymbol{s}(k) - \frac{1}{2}c\boldsymbol{K}_{\vartheta}\boldsymbol{H}^{-1}(2\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{P}\boldsymbol{s}(k) - \tilde{\boldsymbol{r}}(k+1)) + c\boldsymbol{K}_{\vartheta}^{\mathrm{T}}\boldsymbol{K}_{s}\boldsymbol{s}(k))$ is a constant over the course of the optimisation at sample step

k. The distributed MPC algorithm will then converge if:

$$|\text{eig}(\mathbf{I}_{n_\lambda \times n_\lambda} - \frac{c}{2}\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{K}_\vartheta^{\mathrm{T}})|_\infty < 1, \tag{4.26}$$

i.e. the moduli of the eigenvalues of the coefficient of $\tilde{\boldsymbol{\lambda}}(k,l)$ in (4.25) are less than unity (Li et al., 2005; Bertsekas and Tsitsikilis, 1989).

Upon convergence of the augmented Lagrangian iterations $\tilde{\boldsymbol{\lambda}}(k,l+1) \approx \tilde{\boldsymbol{\lambda}}(k,l) = \tilde{\boldsymbol{\lambda}}^*(k)$, where $\tilde{\boldsymbol{\lambda}}^*(k)$ is the optimal values of the Lagrange multipliers at sample step k. Equation (4.25) then gives:

$$\begin{aligned} \tilde{\boldsymbol{\lambda}}^*(k) &= (\mathbf{I}_{n_\lambda \times n_\lambda} - \frac{c}{2}\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{K}_\vartheta^{\mathrm{T}})\tilde{\boldsymbol{\lambda}}^*(k) + \boldsymbol{C}_\lambda(k) \\ \tilde{\boldsymbol{\lambda}}^*(k) &= \frac{2}{c}(\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{K}_\vartheta^{\mathrm{T}})^{-1}\boldsymbol{C}_\lambda(k) \end{aligned} \tag{4.27}$$

Substituting $\tilde{\boldsymbol{\lambda}}^*(k)$ back into the (4.24) and rearranging the matrices gives:

$$\begin{aligned} \boldsymbol{\vartheta}^*(k) &= -\frac{1}{2}\boldsymbol{H}^{-1}\boldsymbol{f}(k,l) \\ &= \boldsymbol{F}\tilde{\boldsymbol{r}}(k+1) - \boldsymbol{W}\boldsymbol{s}(k) \end{aligned} \tag{4.28}$$

where $\boldsymbol{F} = -\boldsymbol{H}^{-1}(\boldsymbol{K}_\vartheta^{\mathrm{T}}(\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{K}_\vartheta^{\mathrm{T}})^{-1}\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q} - \boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q})$ and $\boldsymbol{W} = -\boldsymbol{H}^{-1}(\boldsymbol{K}_\vartheta^{\mathrm{T}}(\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{K}_\vartheta^{\mathrm{T}})^{-1}\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{P} - \boldsymbol{G}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{P} - \frac{c}{2}\boldsymbol{K}_\vartheta^{\mathrm{T}}\boldsymbol{K}_s - \boldsymbol{K}_\vartheta^{\mathrm{T}}(\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{K}_\vartheta^{\mathrm{T}})^{-1}\boldsymbol{K}_s + \frac{c}{2}\boldsymbol{K}_\vartheta^{\mathrm{T}}(\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{K}_\vartheta^{\mathrm{T}})^{-1}\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{K}_\vartheta^{\mathrm{T}}\boldsymbol{K}_s)$.

The inputs can now be applied to the discrete time system. The optimal input applied to the system is:

$$\Delta \boldsymbol{u}^*(k) = \boldsymbol{S}\boldsymbol{\vartheta}^*(k) \tag{4.29}$$

where $\boldsymbol{S}$ picks out the optimal system inputs at sample k from $\boldsymbol{\vartheta}^*(k)$. It is also observed that

$$\boldsymbol{s}(k) = \begin{bmatrix} \boldsymbol{x}^{\mathrm{aug}}(k) \\ \Delta \boldsymbol{v}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n_x \times n_x} \\ \boldsymbol{L} \end{bmatrix} \boldsymbol{x}^{\mathrm{aug}}(k) = \boldsymbol{\Upsilon}\boldsymbol{x}^{\mathrm{aug}}(k) \tag{4.30}$$

where the matrix $\boldsymbol{L}$ picks out $\Delta \boldsymbol{v}(k)$ from $\boldsymbol{x}^{\mathrm{aug}}(k)$.

The discrete time state-space equation for the whole system is then:

$$\begin{aligned} \boldsymbol{x}^{\mathrm{aug}}(k+1) &= \boldsymbol{D}\boldsymbol{s}(k) + \boldsymbol{B}\Delta \boldsymbol{u}^*(k) \\ &= \boldsymbol{D}\boldsymbol{\Upsilon}\boldsymbol{x}^{\mathrm{aug}}(k) + \boldsymbol{B}\boldsymbol{S}(\boldsymbol{F}\tilde{\boldsymbol{r}}(k+1) - \boldsymbol{W}\boldsymbol{s}(k)) \\ &= (\boldsymbol{D}\boldsymbol{\Upsilon} - \boldsymbol{Z}\boldsymbol{S}\boldsymbol{W}\boldsymbol{\Upsilon})\boldsymbol{x}^{\mathrm{aug}}(k) + \boldsymbol{B}\boldsymbol{S}\boldsymbol{F}\tilde{\boldsymbol{r}}(k+1) \end{aligned} \tag{4.31}$$

Therefore the system will then be stable, as in the convergence proof, if:

$$|\mathrm{eig}(\boldsymbol{D\Upsilon} - \boldsymbol{ZSW\Upsilon})|_\infty < 1. \tag{4.32}$$

An example system will now be used to demonstrate the ability of this algorithm to determine the point of instability of a system controlled using the distributed MPC system.

4.4 Example Control System-Two Area Discrete-Time Load Frequency Control

The stability proof is now applied to a simplified discrete-time 2 area LFC problem shown in Fig. 4.3. The controller utilizes an exact model of the system being controlled and the disturbance is used to create a deviation for one sample and then removed again so that there is no model mismatch. It is then possible to determine the accuracy of the stability formula (4.32).

4.4.1 Dynamic system model and control

The continuous-time dynamics of subsystem 1 are described by the following second-order system (Negenborn et al., 2008):

$$\begin{gathered}\frac{\mathrm{d}}{\mathrm{dt}}\delta_1(t) = 2\pi f_1(t), \\ \frac{\mathrm{d}}{\mathrm{dt}}f_1(t) = -\frac{1}{T_{\mathrm{p}_1}}f_1(t) + \frac{K_{\mathrm{p}_1}}{T_{\mathrm{p}_1}}\left(P_{\mathrm{G}1}(t) - P_{\mathrm{d}1}(t) + \frac{K_{\mathrm{S}_{12}}}{2\pi}\left(\delta_2(t) - \delta_1(t)\right)\right), \\ \boldsymbol{y}_1(t) = \begin{bmatrix}\delta_1(t) \\ f_1(t)\end{bmatrix}.\end{gathered} \tag{4.33}$$

The continuous-time dynamics of subsystem 2 are described similarly as follows:

$$\begin{gathered}\frac{\mathrm{d}}{\mathrm{dt}}\delta_2(t) = 2\pi f_2(t), \\ \frac{\mathrm{d}}{\mathrm{dt}}f_2(t) = -\frac{1}{T_{\mathrm{p}_2}}f_2(t) + \frac{K_{\mathrm{p}_2}}{T_{\mathrm{p}_2}}\left(P_{\mathrm{G}2}(t) - P_{\mathrm{d}2}(t) + \frac{K_{\mathrm{S}_{12}}}{2\pi}\left(\delta_1(t) - \delta_2(t)\right)\right), \\ \boldsymbol{y}_2(t) = \begin{bmatrix}\delta_2(t) \\ f_2(t)\end{bmatrix}.\end{gathered} \tag{4.34}$$

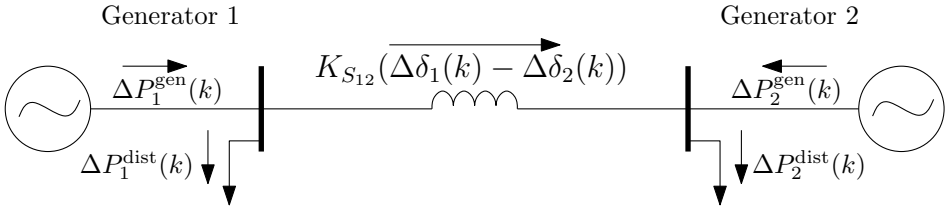


Figure 4.3: The 2 area discrete time LFC.

a	1	2
$T_{\text{P}a}$	20	25
$K_{\text{P}a}$	120	112.5
Q_a	1	1
R_a	0.01	0.01

Table 4.1: 2 area LFC problem's physical and control parameters.

where $f_a(t)$ is the frequency of generator a, and $\boldsymbol{y}_a(t)$ represents the measured output states at time t. The gain $K_{\text{P}a}$, and the time constant $T_{\text{P}a}$, are all constants for the a^{th} subsystem. For the purposes of these simulations output measurements, the output measurements $\boldsymbol{y}_a(t)$ are assumed to be noise free.

In discrete time, subsystem a's local control input is $u_a(k)$=$P_{\text{G}a}(k)$, the local disturbance is $d_a(k) = P_{\text{d}a}(k)$, and the local state is $\boldsymbol{x}_a(k) = [\delta_a(k), f_a(k)]^{\text{T}}$. The external input to subnetwork 1 is $v_1(k) = \delta_2(k)$ and the external input to subnetwork 2 is $v_2(k) = \delta_1(k)$. Discretizing the continuous-time model using an Euler approximation (with sample size τ), the model can be written as in (4.3) with:

$$\begin{aligned} A_a &= \begin{bmatrix} 1 & \tau 2\pi \\ \tau\frac{-K_{\text{P}a}K_{\text{S}12}}{2\pi T_{\text{P}a}} & 1-\frac{\tau}{T_{\text{P}a}} \end{bmatrix}, B_a = \begin{bmatrix} 0 \\ \tau\frac{K_{\text{P}a}}{T_{\text{P}a}} \end{bmatrix}, \\ D_a &= \begin{bmatrix} 0 \\ -\tau\frac{K_{\text{P}a}}{T_{\text{P}a}} \end{bmatrix}, V_a = \begin{bmatrix} 0 \\ \tau\frac{K_{\text{P}a}K_{\text{S}12}}{2\pi T_{\text{P}a}} \end{bmatrix}. \end{aligned} \tag{4.35}$$

The above model is used to run the discrete time simulation with a sample time of τ=0.5s. The two subnetworks' parameters are given in Table 4.1. The states and inputs are also unconstrained in the example.

Typically, problems with stability arise in distributed MPC due to strong interactions

between areas (Venkat, 2006). In this example the interconnection term determines the degree of interaction between the two areas. In this work $K_{\mathrm{S}_{12}}$ was increased until the simulated system under distributed MPC becomes unstable. This was compared with the $K_{\mathrm{S}_{12}}$ predicted by theory for the limit of stability.

An incremental state space model was used for control and a prediction horizon of $N = 2$ was used to adequately take into account each subnetwork's dynamic response. The following stage quadratic cost function is used for the distributed MPC:

$$J_a^{\mathrm{stage}}(k,p) = Q_a f_a^2(k+p+1) + R_a \Delta u_a^2(k+p) \tag{4.36}$$

where Q_a and R_a are the scalar weights in the cost functions for the variables $f_a(k+p)$ and $\Delta u_a(k+p)$ respectively. Q_a and R_a maintain the same value for all stages of the prediction horizon. Using this stage cost, $J_a^{\mathrm{local}}(k)$ is formed as in (4.21). Values for Q_a and R_a are given in Table 4.1.

The interconnecting inputs of the a^{th} agent are $\delta_j(k)$, where agent j is the agent connected to agent a, and the interconnecting output is $\delta_a(\mathrm{k})$. The following gives the interconnection cost agent a experiences due to its connection to agent j:

$$\begin{aligned} J_{ja}^{\mathrm{inter}}(k,l) = &\begin{bmatrix} \tilde{\boldsymbol{\lambda}}_{ja}^{\mathrm{in},\boldsymbol{\delta}_j}(k,l) \\ -\tilde{\boldsymbol{\lambda}}_{aj}^{\mathrm{in},\boldsymbol{\delta}_a}(k,l) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \tilde{\boldsymbol{w}}_{ja}^{\mathrm{in},\boldsymbol{\delta}_j}(k,l) \\ \tilde{\boldsymbol{w}}_{ja}^{\mathrm{out},\boldsymbol{\delta}_a}(k,l) \end{bmatrix} \\ &+ \frac{c}{2} \left\| \begin{bmatrix} \tilde{\boldsymbol{w}}_{aj,\mathrm{prev}}^{\mathrm{in},\boldsymbol{\delta}_j}(k,l) - \tilde{\boldsymbol{w}}_{ja}^{\mathrm{out},\boldsymbol{\delta}_a}(k,l) \\ \tilde{\boldsymbol{w}}_{aj,\mathrm{prev}}^{\mathrm{out},\boldsymbol{\delta}_a}(k,l) - \tilde{\boldsymbol{w}}_{ja}^{\mathrm{in},\boldsymbol{\delta}_a}(k,l) \end{bmatrix} \right\|_2^2, \end{aligned} \tag{4.37}$$

where the extra superscript on the Lagrange multiplier and interconnecting input and output variables denotes the specific state to which each of these variables is related, i.e., $\tilde{\boldsymbol{\lambda}}_{ja}^{\mathrm{in},\boldsymbol{\delta}_j}(k,l)$ is associated with the interconnecting input $\boldsymbol{\delta}_j$. Agent a directly optimises for $\tilde{\boldsymbol{w}}_{ja}^{\mathrm{in},\boldsymbol{\delta}_j}(k,l)$ and sends these variables to the other agent to let it know what values of $\tilde{\boldsymbol{\delta}}_j$ it would like to receive. Through optimisation of $\tilde{\boldsymbol{u}}_a(k,l)$ and $\tilde{\boldsymbol{w}}_{ja}^{\mathrm{in},\boldsymbol{\delta}_j}(k,l)$ agent a manipulates $\tilde{\boldsymbol{w}}_{ja}^{\mathrm{out},\boldsymbol{\delta}_a}(k,l)$, and sends this to the agent j so it knows what values of $\tilde{\boldsymbol{\delta}}_a$ agent a plans to send to it over the prediction horizon. The overall cost function for each agent is formed using (4.36) and (4.37), as in (4.15).

4.4.2 Formation of matrices for stability proof

Here $\hat{\boldsymbol{A}}^{\mathrm{f}}$=diag($\hat{\boldsymbol{A}}_1^{\mathrm{f}}$,$\hat{\boldsymbol{A}}_2^{\mathrm{f}}$), $\hat{\boldsymbol{B}}^{\mathrm{f}}$=diag($\hat{\boldsymbol{B}}_1^{\mathrm{f}}$,$\hat{\boldsymbol{B}}_2^{\mathrm{f}}$), $\hat{\boldsymbol{V}}^{\mathrm{f}}$=diag($\hat{\boldsymbol{V}}_1^{\mathrm{f}}$,$\hat{\boldsymbol{V}}_2^{\mathrm{f}}$), $\boldsymbol{Q}$=diag($\boldsymbol{Q}_1$,$\boldsymbol{Q}_2$), $\boldsymbol{R}$ =diag($\boldsymbol{R}_1, \boldsymbol{R}_2$), and the augmented state $\boldsymbol{x}^{\mathrm{aug}}(k) = [\boldsymbol{x}_1^{\mathrm{augT}}(k), \boldsymbol{x}_2^{\mathrm{augT}}(k)]^{\mathrm{T}}$. The inputs are $\Delta\tilde{\boldsymbol{u}}(k,l) = [\Delta\tilde{\boldsymbol{u}}_1^{\mathrm{T}}(k,l),\Delta\tilde{\boldsymbol{u}}_2^{\mathrm{T}}(k,l)]^{\mathrm{T}}$, interconnecting inputs $\Delta\tilde{\boldsymbol{w}}^{\mathrm{in}}(k,l) = [\Delta\tilde{\boldsymbol{w}}_1^{\mathrm{inT}}(k,l)$, $\Delta\tilde{\boldsymbol{w}}_2^{\mathrm{inT}}(k,l)]^{\mathrm{T}}$, and the setpoints $\tilde{\boldsymbol{r}}(k)$=$[\tilde{\boldsymbol{r}}_1^{\mathrm{T}}(k), \tilde{\boldsymbol{r}}_2^{\mathrm{T}}(k)]^{\mathrm{T}}$ are arranged as shown. From these, $\boldsymbol{s}(k)$, $\boldsymbol{D}$, $\boldsymbol{Z}$, and $\boldsymbol{R}_\vartheta$ can be constructed as explained in the Section 4.3. The matrix $\boldsymbol{K}_c$ is arranged as follows so as to extract $[\tilde{\boldsymbol{f}}_1^{\mathrm{T}}(k), \tilde{\boldsymbol{f}}_2^{\mathrm{T}}(k)]^{\mathrm{T}}$, from the state predictions:

$$\boldsymbol{K}_c = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The matrix $\boldsymbol{K}_v$ is constructed as follows to extract the matrix $[v_1(k), v_2(k)]^{\mathrm{T}}$=$[\delta_2(k), \delta_1(k)]^{\mathrm{T}}$:

$$\boldsymbol{K}_v = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrix $\boldsymbol{K}_w$ is constructed in order to extract the vector $[\tilde{\boldsymbol{w}}_1^{\mathrm{inT}}(k,l), \tilde{\boldsymbol{w}}_2^{\mathrm{inT}}(k,l)]^{\mathrm{T}}$ from $\boldsymbol{K}_v\boldsymbol{s}(k) + \boldsymbol{K}_w\boldsymbol{\vartheta}(k,l)$ as follows:

$$\boldsymbol{K}_v = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$\boldsymbol{K}_I$ is then constructed to pick out the interconnecting outputs that match the interconnecting inputs $[\tilde{\boldsymbol{w}}_1^{\mathrm{inT}}(k,l), \tilde{\boldsymbol{w}}_2^{\mathrm{inT}}(k,l)]^{\mathrm{T}}$ as follows:

$$\boldsymbol{K}_I = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Given these matrices, it is possible to proceed to construct the other matrices necessary to determine if the system is stable, as outlined in the previous section.

4.4.3 Results

Disturbances $P_{\mathrm{d1}} = 0.3$ pu and $P_{\mathrm{d2}} = -0.3$ pu were applied to the system at the first sample and then removed in order to perturb the system. Simulations were run for 100

c	$\lvert\text{eig}(\mathbf{I}_{n_\lambda \times n_\lambda} - \frac{c}{2}\boldsymbol{K}_\vartheta \boldsymbol{H}^{-1}\boldsymbol{K}_\vartheta^{\mathrm{T}})\rvert_\infty$	$\lvert\text{eig}(\boldsymbol{D\Upsilon} - \boldsymbol{ZSW\Upsilon})\rvert_\infty$
0.1	0.997	0.9985
1	0.971	0.9985
10	0.77	0.9985
100	0.251	0.9985

Table 4.2: The effect of varying c, with fixed values of $\boldsymbol{Q}_a$ and $\boldsymbol{R}_a$, for $a = 1, 2$, on the maximum eigenvalues associated with the convergence of the Lagrange multipliers and the system stability, when $K_{\mathrm{S}_{12}} = 258$, $\epsilon = 10^{-3}$.

samples. After the disturbances were removed there was no model mismatch so one would expect an accurate prediction of when the system goes unstable, from (4.32). First, the interconnection coefficient $K_{\mathrm{S}_{12}}$ was increased in value to the point where it causes the modulus of one of the eigenvalues in (4.32) to be just greater than 1, with c=1 and $\epsilon = 10^{-3}$. The theoretically predicted value of $K_{\mathrm{S}_{12}}$ at which instability occurred under distributed MPC control was then compared to the value of $K_{\mathrm{S}_{12}}$ at which the simulated system went unstable. The predicted value of $K_{\mathrm{S}_{12}}$ at which the system goes unstable is 259, which closely matches the simulated point at which the system goes unstable which is $K_{\mathrm{S}_{12}} = 260$.

Using (4.26) and (4.32) it is possible to examine the effects of c upon both the rate of convergence of the Lagrange multipliers (the second column) and the stability of the system (the third column), as can be seen in Table 4.2 ($\boldsymbol{Q}_a$ and $\boldsymbol{R}_a$ also affect the stability but these are kept constant here). It can be seen that for this system the maximum eigenvalue related to the stability of the system do not change with the value of c. When the system is simulated, it is found that this observation is true as instability only occurs when $K_{\mathrm{S}_{12}} = 260$ for a wide choice of c. However, as c increases the maximum moduli of the eigenvalues related to convergence decrease, which indicates a reduction in the number of iterations taken to achieve convergence. This can be observed in Figs. 4.4 and 4.5 for increasing values of c. Intuitively, this would be expected to happen, as increasing c places greater emphasis on the satisfaction of the equality constraints which should ensure that the Lagrange multipliers converge sooner.

The effect of varying ϵ when c=1 was also examined. Instability is predicted to happen at $K_{\mathrm{S}_{12}} = 259$, using (4.32). For $\epsilon = 10^{-3}, 2 \times 10^{-3}, 3 \times 10^{-3}$, the simulation stays stable up to $K_{\mathrm{S}_{12}} = 260, 261, 264$, respectively. The smaller ϵ was, the closer the predicted and

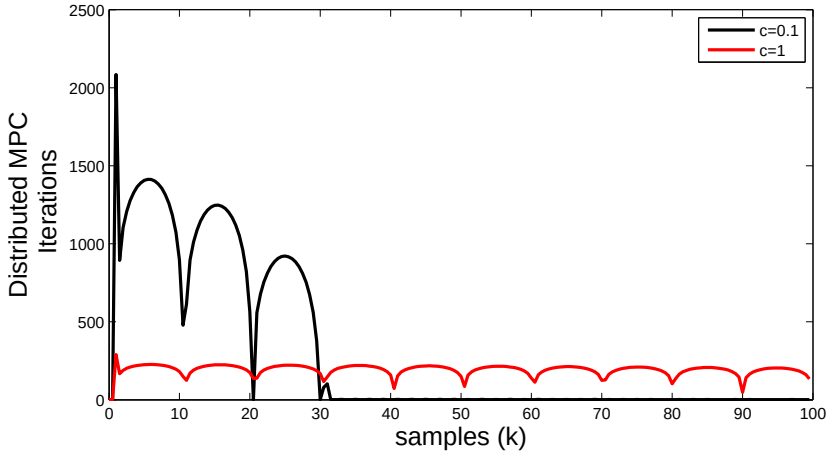


Figure 4.4: Distributed MPC iterations at each sample with small values of c when $K_{S_{12}} = 258$, $\epsilon = 10^{-3}$.

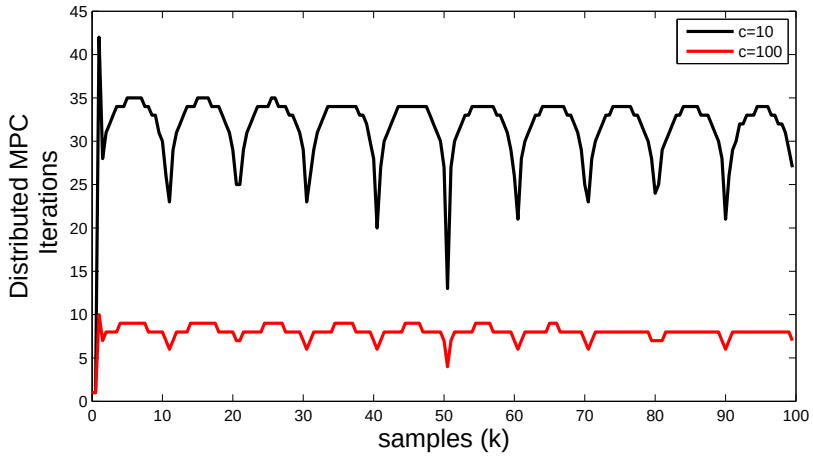


Figure 4.5: Distributed MPC iterations at each sample with large values of c when $K_{S_{12}} = 258$, $\epsilon = 10^{-3}$.

simulated points of system instability were to each other. For small increments in ϵ it is possible to increase the region in which stable control is possible in the simulated system. As the values of ϵ become larger, large offsets begin to be introduced into the steady state as can be seen in Fig. 4.6, where $K_{\mathrm{S}_{12}} = 260$, c=1, and $\epsilon = 10^{-2}$. As was stated above when $K_{\mathrm{S}_{12}} = 260$ and $\epsilon = 2\times10^{-3}$ stable system operation was given without these offsets.

These results indicate that when ϵ is small that the convergence and stability of the control can accurately be predicted for a given system. A further comparison was also

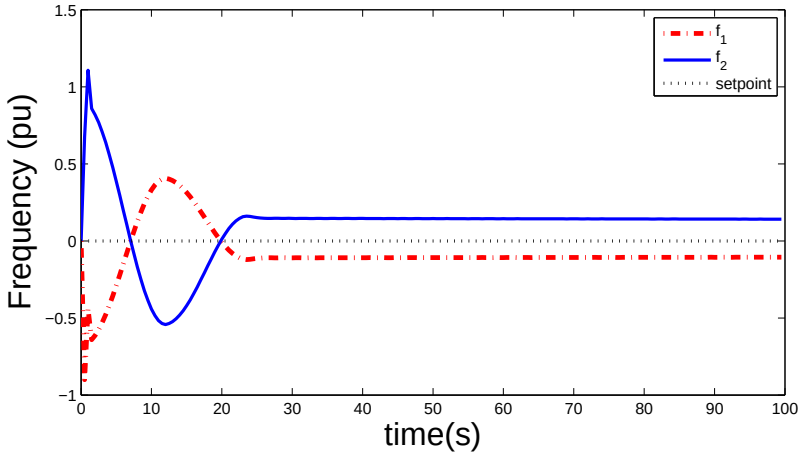


Figure 4.6: Response when $K_{S_{12}} = 260$, c=1, and $\epsilon = 10^{-2}$.

made on the point at which instability occurs with a decentralised MPC approach, where independent MPCs are carried out by each subsystem not taking into account the effect of interconnecting inputs on the subsystems. Likewise, the stability limit for $K_{S_{12}}$ was also determined for a centralised MPC that reaches a Pareto equilibrium. The decentralised MPC approach was found to go unstable when $K_{S_{12}} = 0.77$. This demonstrates the potentially large disparities that may occur between the level of interconnection distributed MPCs can handle versus decentralised MPCs. As a final comparison, the limit of stability for the centralised MPC was $K_{S_{12}} = 605$. This means that centralised MPC could handle over twice the level of interconnection that distributed MPC could. While this may be the case, the problems inherent in the implementation of centralised MPCs for large systems have already been outlined.

4.5 Summary

In this chapter distributed MPC based on the Alternating Direction Method of Multipliers (ADMOM) was introduced, and a novel proof for the convergence and stability conditions of this algorithm were derived for the linear unconstrained case. An example was outlined showing how the convergence and stability of the control algorithm were accurately determined a priori for a 2 area LFC system. It was also noted that distributed MPC could not stabilise the example system when there was a very large interconnection coefficient, despite it being controllable using centralised MPC. However, distributed MPC was shown

to be capable of handling large values of the interconnection coefficient in comparison to a decentralised MPC approach. It could be seen that the trade off for this performance is a large communication overhead. For highly interconnected systems with short sample times, stabilisation with distributed MPC may not be guaranteed. However, typical power systems do not have the extremely high interconnection costs noted in this chapter.

While stability conditions for the constrained distributed MPC algorithm have not yet been given, (Negenborn, 2007) has shown how a number of power systems were adequately controlled when constraints were added to the formulation of the control problem. The derivation of stability for the constrained case would be quite useful and the author would cite this as potential future work. Ways of limiting the amount of communication needed for the convergence of the distributed MPC would also be quite useful, especially in a Smart Grid scenario. In the next chapter, the application of distributed MPC to a challenging power systems testbed with constraints is investigated.

References

D. P. Bertsekas and J. N. Tsitsikilis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, New Jersey, USA, 1 edition, 1989.

E. Camponogara, D. Jia, B.H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22(1):44–52, Feb 2002.

R.M. Hermans, M. Lazar, and A. Jokic. Almost decentralized lyapunov-based nonlinear model predictive control. In *American Control Conference*, pages 3932–3938, July 2010.

S. Li, Y. Zhang, and Q. Zhu. Nash-optimization enhanced distributed model predictive control applied to the shell benchmark problem. *Information Sciences*, 170:329–349, 2005.

J. Liu, X. Chen, D. Muñoz de la Peña, and P. D. Christofides. Sequential and iterative architectures for distributed model predictive control of nonlinear process systems. *AIChE Journal*, 56(8):2137–2149, 2010.

R. R. Negenborn. *Multi-Agent Model Predictive Control with Application to Power Networks*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 2007.

R. R. Negenborn, B. De Schutter, and J. Hellendoorn. Multi-agent model predictive control for transportation networks: Serial versus parallel schemes. *Engineering Applications of Artificial Intelligence*, 21(3):353–366, April 2008.

J. Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operation Research and Financial Engineering. Springer, New York, USA, New York, USA, 2006.

C. Ocampo-Martinez, S. Bovo, and V. Puig. Partitioning approach oriented to the decentralised predictive control of large-scale systems. *Journal of Process Control*, 21(5):775 – 786, 2011.

S. Tosserams. *Distributed Optimization for Systems Design: An Augmented Lagrangian Coordination Method*. PhD thesis, Technical University Eindhoven, Eindhoven, Netherlands, 2008.

S. Tosserams, L. F. P. Etman, and J. E. Rooda. Augmented Lagrangian coordination for distributed optimal design in MDO. *International Journal for Numerical Methods in Engineering*, 73(13):1885–1910, 2008.

A. Venkat. *Distributed Model Predictive Control: Theory and Applications*. PhD thesis, University of Wisconsin-Madison, Wisconsin, 2006.

Chapter 5

Optimal Coordination of a Multiple HVDC Link System Using Centralised and Distributed control

5.1 Introduction

In previous chapters, the systems that the MPC techniques have been applied to have been linearised power systems. In this chapter distributed, centralised, and decentralised MPC are applied to a non-linear power system testbed and the performance of the MPC systems are compared with that of an optimised multi-loop PID control scheme. The use of PID controllers for power systems control is widespread. Distributed MPC is a scalable MPC architecture which could potentially replace PID controllers for the control of power systems. Thus, it is interesting to compare the performance of the optimised PID controller based scheme with a distributed MPC controller on a challenging non-linear power systems testbed. As part of this analysis it is also necessary to take into account the computational and communication overhead associated with the distributed MPC to get a better idea of the trade-offs involved when using distributed MPC.

One suitable challenging testbed for making such comparisons was presented in (Erikkson, 2008). It is a multiple HVDC link system based on part of the Nordic power grid. This is a non-linear, MIMO dynamic system. The power generated in the system is kept constant and the modulation of the HVDC links alone is then used to restabilise the system after line faults. Thus, there is a high level of interconnectivity between subsystems in this model. So far, centralised control techniques that were not based on optimisation, were used to control this system (Erikkson, 2008).

Typically distributed MPC systems will try to reach consensus on variables that flow between agents. However in the multiple link HVDC system all 4 agents share the same 2 control inputs. Thus, it is shown in this chapter how the distributed MPC scheme discussed in the previous chapter can naturally be used to handle situations in which agents share inputs. Then, three optimisation-based control techniques are proposed for the control of the Nordic multiple HVDC link system. The controllers used are an off-line PSO-optimised PID controller, a centralised MPC controller, and the distributed MPC controller of the previous chapter. It is also shown that decentralised MPC, where there is no inter-agent communication, is highly unsuitable for the control of this system.

First, the multiple HVDC link system will be introduced. Then an introduction to stochastic optimisation, in particular Particle Swarm Optimisation (PSO) will be given and it will be shown how this can be used in order to optimise PID controller performance. Then a comparison of the 3 controllers will be given in the final section of this chapter.

5.2 The Multiple HVDC Link System

The continuous-time dynamics of the multiple HVDC link system under study here are described in this section. The system, which is based on the multiple HVDC link system between Denmark, Norway, and Sweden, is depicted in Fig. 5.1 (Erikkson, 2008). It consists of 4 buses with their own generation and loads. Both Alternating Current (AC) and HVDC links connect the buses. The HVDC links are of the Line Commutated Converter type (Pai et al., 1981). Generation capacities and loads are kept constant in this chapter. Large amounts of power are transferred from bus 2, which has the largest generation capacity, to bus 4, which has the largest power load.

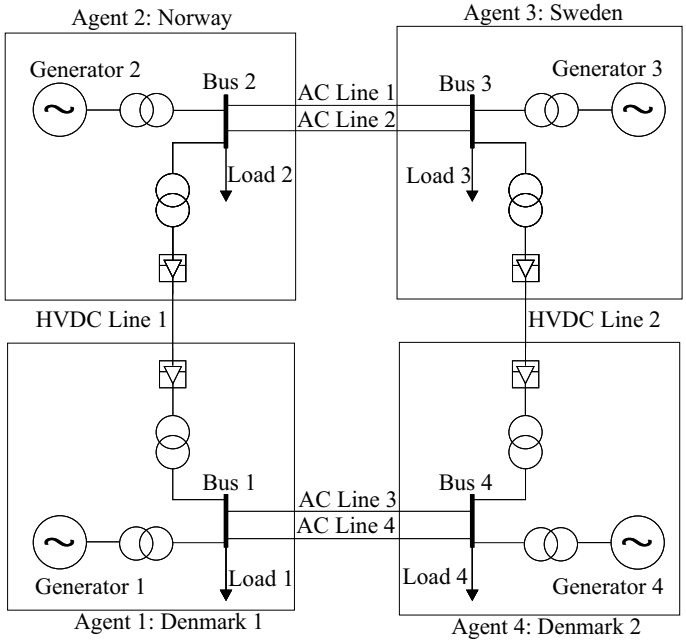


Figure 5.1: The multiple HVDC link system with areas controlled by agents (Erikkson, 2008).

The objective of any control scheme here is to maintain the rotor frequencies as close as possible to 1 pu at all times for the multiple link HVDC system. The total generated power equals the total consumed power in the network, and so the modulation of the HVDC link powers alone should be sufficient to restabilise the generator frequencies at each bus, and return them to their desired setpoints after frequency deviations are incurred due to line fault disturbances. It is desirable that the rotor frequencies return to this setpoint in the minimum possible time after disturbances, and that the magnitude of any deviations from the setpoint are also minimised. Also a multi-agent, distributed approach is preferred for controlling this system since it spans a vast geographical area and crosses international borders, which means that several separate agents are responsible for the control of the different subsystems.

5.2.1 Generator Model

The classical swing equations for a generator a are (Kundur, 1994):

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta_{\mathrm{r}_a}(t) = \omega_0 \Delta\omega_{\mathrm{r}_a}(t) \tag{5.1}$$

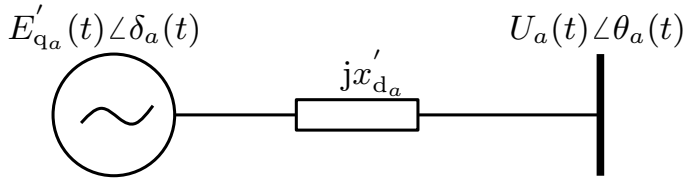


Figure 5.2: Generator a connected to a bus.

$$\frac{\mathrm{d}}{\mathrm{d}t}\omega_{r_a}(t) = \frac{1}{2H_a}(P_{m_a}(t) - P_{G_a}(t) - D_a\Delta\omega_{r_a}(t)), \tag{5.2}$$

where $\delta_{r_a}(t)$ is the rotor angle (rad), H_a is the inertial constant (s), $\omega_{r_a}(t)$ is the rotor speed (per unit), $\Delta\omega_{r_a}(t)$= $\omega_{r_a}(t)-1$ is the rotor speed deviation (per unit), ω_0 is the base rotor speed (rad/s), $P_{m_a}(t)$ and $P_{G_a}(t)$ are the mechanical and generated power (per unit), respectively, and D_a is the damping factor (per unit).

The current injected by generator a, $\overrightarrow{I}_{g_a}(t)$, is given by:

$$\overrightarrow{I}_{g_a}(t) = \frac{\overrightarrow{E}'_{q_a}(t) - \overrightarrow{U}_a(t)}{jx'_{d_a}}, \tag{5.3}$$

where $\overrightarrow{E}'_{q_a}(t) = E'_{q_a}(t)\angle\delta_a(t)$ is the internal voltage (per unit) with magnitude $E'_{q_a}(t)$ and angle $\delta_a(t)$, $\overrightarrow{U}_a(t) = U_a(t)\angle\theta_a(t)$ is the voltage (per unit) at the bus to which the generator is connected with magnitude $U_a(t)$ and angle $\theta_a(t)$, and x'_{d_a} is the d-axis transient reactance, as seen in Fig. 5.2. All variables are defined as in the standard reference work (Kundur, 1994). The generated power is then given by:

$$P_{G_a}(t) = \mathbb{R}[\overrightarrow{E}'_{q_a}(t)\,\overrightarrow{I}^*_{g_a}(t)]. \tag{5.4}$$

Equations (5.1) and (5.2) of the classical model of a synchronous generator assume that $E'_{q_a}(t)$ is constant (Kundur, 1994). These classical equations are suitable for analysis of power oscillations and transient stability studies.

5.2.2 Power System Model

A π-model representation (Kundur, 1994) of the AC links is used, as in Fig. 5.3, where X_{L_i} is the line reactance, X_{S_i} is the shunt reactance, X_{G_i} is a ground reactance used here

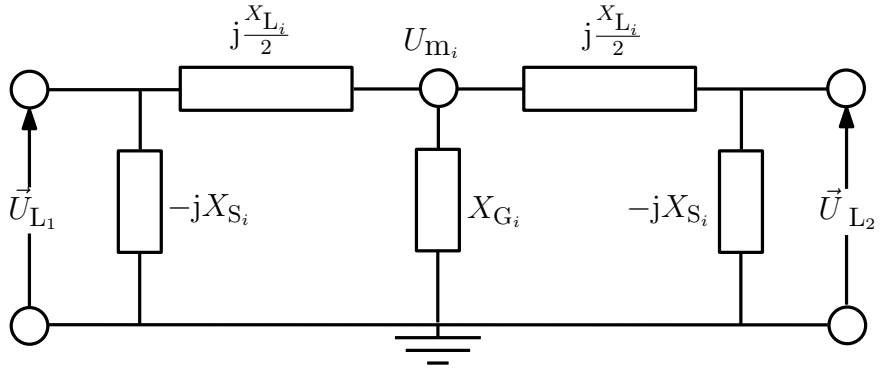


Figure 5.3: π-Model of lines.

to simulate 3-phase to ground faults in the middle of line i. The line voltages at either side of line i are U_{L_1} and U_{L_2}, respectively, and U_{m_i} is the voltage at the middle of the line connected to ground through X_{G_i}. The value of X_{G_i} decreases from ∞ in the non-fault state to 0 in the case of the 3-phase fault. As the fault occurs in the middle of the line, the line reactance is divided in half on either side of the ground fault "line".

A number of simplifications are made to the power system model, subsequently reducing the complexity of the MPC problem proposed in the next section, and making it faster, without neglecting the system dynamics most relevant to system stability.

- Loads are modelled as constant impedances.
- HVDC power transmission is considered instantaneous.
- The HVDC reactive power is taken as proportional to HVDC active power.

The details of these simplifications are given in the following paragraphs.

Loads are modelled as constant impedances in the admittance matrix, i.e. with $\overrightarrow{S}_a = P_a^{\mathrm{LL}} + jQ_a^{\mathrm{LL}}$, the complex load power in VA, the load impedance $\overrightarrow{Z}_a^{\mathrm{LL}}$ is given by

$$\overrightarrow{Z}_a^{\mathrm{LL}} = \frac{\overrightarrow{U}_a \overrightarrow{U}_a^*}{\overrightarrow{S}_a^*}. \tag{5.5}$$

The HVDC link model in (Erikkson, 2008) is used to simplify the representation of the system dynamics. This idealised version of the HVDC link assumes instantaneous, lossless

power delivery and power factors that are equal on both the inverter and rectifier sides. Thus, when active power injections from a HVDC link j are applied to a bus in this model, the bus 'sees' $+P_j^{\mathrm{DC}}$, where P_j^{DC} is the active power flow in the HVDC link. Equally, the bus from which the HVDC active power is being sent sees a -P_j^{DC}. This model is further simplified by assuming that $Q_j^{\mathrm{DC}} = q_{\mathrm{r}_j} P_j^{\mathrm{DC}}$, where q_{r_j} is a constant, and Q_j^{DC} is the reactive HVDC power in HVDC link j (Pai et al., 1981). A simplification adopted in this chapter is to directly calculate and apply the HVDC powers. However in a real system, currents are injected that are calculated from these powers (Kundur, 1994).

5.2.3 Modelling of the Swing Equation Using the Internal Node Representation

The internal node representation is used to model the generator swing equation from (5.2) as it allows the power system to then be represented using a system of first-order differential equations (Erikkson, 2008; Pai et al., 1981; Sauer and Pai, 1998). This is done by rearranging the admittance matrix of the power network so as to find its generator currents in terms of the network voltages and HVDC currents, and then substituting the values for these currents into (5.2). This is demonstrated in the following paragraphs. To do this it is assumed that P_{m_a} is constant and that the loads are modelled as constant impedances.

An admittance matrix gives the relationship between the voltage nodes and currents in a power system. Lines and loads are then represented by admittances in this matrix. Consider the a^{th} bus, with n voltage nodes and m HVDC links connected to it. Then using Kirchhoff's current law, which states that the sum of the currents entering a node equals the sum of those leaving the node, the following is given:

$$\sum_{l=1}^{n} \vec{Y}_{a,l} \vec{U}_l - \vec{I}_{\mathrm{g}_a} - \sum_{j=1}^{m} \vec{I}_{\mathrm{DC}_{a,\,j}} = 0, \tag{5.6}$$

where $\vec{Y}_{a,l} = \frac{1}{\vec{Z}_{a,l}}$, where $\vec{Z}_{a,l}$ is the impedance between bus a and voltage node l and $\vec{I}_{\mathrm{DC}_{a,\,j}}$ is the current injected into bus a from HVDC link j. This is repeated for all voltage nodes in the network.

Using (5.6), an admittance matrix can be constructed, based on models for the AC lines,

the generators, and the loads, as described above:

$$\begin{pmatrix} \boldsymbol{I}_{\mathrm{g}} \\ \boldsymbol{I}_{\mathrm{DC}} \\ 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{Y}_{\mathrm{A}} & \boldsymbol{Y}_{\mathrm{B}} & 0 \\ \boldsymbol{Y}_{\mathrm{C}} & \boldsymbol{Y}_{\mathrm{D}} & \boldsymbol{Y}_{\mathrm{E}} \\ 0 & \boldsymbol{Y}_{\mathrm{F}} & \boldsymbol{Y}_{\mathrm{G}} \end{pmatrix} \begin{pmatrix} \boldsymbol{E} \\ \boldsymbol{U} \\ \boldsymbol{U}_{\mathrm{m}} \end{pmatrix}, \tag{5.7}$$

where $\boldsymbol{I}_{\mathrm{g}} = [\vec{I}_{\mathrm{g}_1}, \ldots, \vec{I}_{\mathrm{g}_n}]^{\mathrm{T}}$, $\boldsymbol{E} = [\vec{E}'_{\mathrm{q}_1}, \ldots, \vec{E}'_{\mathrm{q}_n}]^{\mathrm{T}}$, $\boldsymbol{U} = [\vec{U}_1, \ldots, \vec{U}_n]^{\mathrm{T}}$ and $\boldsymbol{U}_{\mathrm{m}} = [\vec{U}_{\mathrm{m}_1}, \ldots, \vec{U}_{\mathrm{m}_b}]^{\mathrm{T}}$ where b is the number of AC lines in the system and $\boldsymbol{I}_{\mathrm{DC}} = [\vec{I}_{\mathrm{DC}_{1,\,1}}, \ldots, \vec{I}_{\mathrm{DC}_{n,\,m}}]^{\mathrm{T}}$.

By finding an expression for the generator currents in terms of the system voltages and HVDC currents it is possible to express $P_{\mathrm{G}a}(t)$ in terms of these voltages and HVDC currents using (5.4). From (5.7) the following can be found for $\boldsymbol{I}_{\mathrm{g}}$ in terms of $\boldsymbol{I}_{\mathrm{DC}}$ and $\boldsymbol{E}$:

$$\begin{aligned} \boldsymbol{I}_{\mathrm{g}} &= (\boldsymbol{Y}_{\mathrm{A}} - \boldsymbol{Y}_{\mathrm{B}}(\boldsymbol{Y}_{\mathrm{D}} - \boldsymbol{Y}_{\mathrm{E}}\boldsymbol{Y}_{\mathrm{G}}^{-1}\boldsymbol{Y}_{\mathrm{F}}))\boldsymbol{E} + \boldsymbol{Y}_{\mathrm{B}}(\boldsymbol{Y}_{\mathrm{D}} - \boldsymbol{Y}_{\mathrm{E}}\boldsymbol{Y}_{\mathrm{G}}^{-1}\boldsymbol{Y}_{\mathrm{F}})^{-1}\boldsymbol{I}_{\mathrm{DC}} \\ &= (\boldsymbol{G} + j\boldsymbol{B})\boldsymbol{E} + \boldsymbol{Y}_{\mathrm{DC}}\boldsymbol{I}_{\mathrm{DC}} \end{aligned} \tag{5.8}$$

where $\boldsymbol{G}{=}\mathbb{R}[\boldsymbol{Y}_{\mathrm{A}}{-}\boldsymbol{Y}_{\mathrm{B}}(\boldsymbol{Y}_{\mathrm{D}}{-}\boldsymbol{Y}_{\mathrm{E}}\boldsymbol{Y}_{\mathrm{G}}^{-1}\boldsymbol{Y}_{\mathrm{F}})]$, $\boldsymbol{B}{=}\mathbb{I}[\boldsymbol{Y}_{\mathrm{A}}{-}\boldsymbol{Y}_{\mathrm{B}}(\boldsymbol{Y}_{\mathrm{D}}{-}\boldsymbol{Y}_{\mathrm{E}}\boldsymbol{Y}_{\mathrm{G}}^{-1}\boldsymbol{Y}_{\mathrm{F}})]$, and $\boldsymbol{Y}_{\mathrm{DC}}{=}\boldsymbol{Y}_{\mathrm{D}}{-}\boldsymbol{Y}_{\mathrm{E}}\boldsymbol{Y}_{\mathrm{G}}^{-1}\boldsymbol{Y}_{\mathrm{F}}$.

The following swing equation for generator a can be derived using (5.2), (5.4), and (5.8):

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\omega_{\mathrm{r}_a}(t) =& \frac{1}{2H_a}\Big(P_{\mathrm{m}_a} - G_{a,a}E'^{2}_{\mathrm{q}_a} - \sum_{\substack{l=1 \\ l\neq a}}^{n} E'_{\mathrm{q}_a}E'_{\mathrm{q}_l}(G_{a,l}\cos(\delta_{\mathrm{r}_a}(t) - \delta_{\mathrm{r}_l}(t)) \\ &+ B_{a,l}\sin(\delta_{\mathrm{r}_a}(t) - \delta_{\mathrm{r}_l}(t))) + g_{a,1}P_1^{\mathrm{DC}}(t) + \ldots + g_{a,m}P_m^{\mathrm{DC}}(t) - D_a\Delta\omega_{\mathrm{r}_a}(t)\Big), \end{aligned} \tag{5.9}$$

where $g_{a,j}$ is the coefficient of the contribution of the power injections from HVDC link j at bus a. An illustrative example will now be given as to how these parameters are derived for generator 1. The same process can be used in the derivation of these parameters for each of the other generator equations.

Taking equation (5.9) for generator 1 gives:

$$\begin{aligned} \dot{\omega}_{\mathrm{r}_1}(t) = \frac{1}{2H_1}\Big(&P_{\mathrm{m}_1} - G_{1,1}E'^{2}_{\mathrm{q}_1} - E'_{\mathrm{q}_1}E'_{\mathrm{q}_4}(G_{1,4}\cos(\delta_{\mathrm{r}_1}(t) - \delta_{\mathrm{r}_4}(t)) + B_{1,4}\sin(\delta_{\mathrm{r}_1}(t) - \delta_{\mathrm{r}_4}(t))) \\ &+ g_{1,1}P_1^{\mathrm{DC}}(t) + g_{1,2}P_2^{\mathrm{DC}}(t) - D_1\Delta\omega_{r_1}(t)\Big) \end{aligned} \tag{5.10}$$

The $g_{1,1}$ and $g_{1,2}$ parameters are derived as follows: Assume that power flows from bus 2 to 1 in HVDC link 1 and from bus 3 to 4 in HVDC link 2:

$$\boldsymbol{I}_{\mathrm{DC}} = \begin{pmatrix} \vec{I}^{\mathrm{DC}}_{1,1} \\ \vec{I}^{\mathrm{DC}}_{2,1} \\ \vec{I}^{\mathrm{DC}}_{3,2} \\ \vec{I}^{\mathrm{DC}}_{4,2} \end{pmatrix} = \begin{pmatrix} (\frac{P^{\mathrm{DC}}_1 - \mathrm{j}Q^{\mathrm{DC}}_1}{\vec{U}_1})^* \\ (\frac{-P^{\mathrm{DC}}_1 - \mathrm{j}Q^{\mathrm{DC}}_1}{\vec{U}_2})^* \\ (\frac{P^{\mathrm{DC}}_2 - \mathrm{j}Q^{\mathrm{DC}}_2}{\vec{U}_3})^* \\ (\frac{-P^{\mathrm{DC}}_2 - \mathrm{j}Q^{\mathrm{DC}}_2}{\vec{U}_4})^* \end{pmatrix} \tag{5.11}$$

The generator power

$$P_{\mathrm{G}_1} = \mathbb{R}(\vec{E}'_{\mathrm{q}_1}\vec{I}^*_{\mathrm{g}_1}) = \mathbb{R}\left(\vec{E}'_{\mathrm{q}_1}\boldsymbol{T}_{\vec{I}_{\mathrm{q}_1}}\Big((\boldsymbol{G}+\mathrm{j}\boldsymbol{B})\boldsymbol{E} + \boldsymbol{Y}_{\mathrm{DC}}\boldsymbol{I}_{\mathrm{DC}}\Big)^*\right) \tag{5.12}$$

where $\boldsymbol{T}_{\vec{I}_{\mathrm{G}_1}}$ is a matrix of ones that picks out $\vec{I}_{\mathrm{G}_1}$. It is the $\boldsymbol{Y}_{\mathrm{DC}}\boldsymbol{I}_{\mathrm{DC}}$ part of this term that gives the $g_{1,1}$ and $g_{1,2}$ parameters:

$$\begin{aligned} g_{1,1}P^{\mathrm{DC}}_1 + g_{1,2}P^{\mathrm{DC}}_2 &= \mathbb{R}\Big(-\vec{E}'_{\mathrm{q}_1}T_{\vec{I_{\mathrm{G}_1}}}(\boldsymbol{Y}_{\mathrm{DC}}\boldsymbol{I}_{\mathrm{DC}})^*\Big) \\ &= \mathbb{R}\Big(-\frac{\vec{E}'_{\mathrm{q}_1}}{\vec{U}_1}(d^{\mathrm{R}}_{1,1}+\mathrm{j}d^{\mathrm{I}}_{1,1})^*(P^{\mathrm{DC}}_1(1-\mathrm{j}q_{\mathrm{r}_1})) \\ &\qquad -\frac{\vec{E}'_{\mathrm{q}_1}}{\vec{U}_4}(d^{\mathrm{R}}_{1,4}+\mathrm{j}d^{\mathrm{I}}_{1,4})^*(P^{\mathrm{DC}}_2(1-\mathrm{j}q_{\mathrm{r}_2}))\Big) \\ &= \frac{-1}{U_1U_4}\Big(d^{\mathrm{R}}_{1,1}\vec{E}'_{\mathrm{q}_1}U_4\cos(\delta_{\mathrm{r}_1,0}-\theta_{1,0}) - d^{\mathrm{I}}_{1,1}\vec{E}'_{\mathrm{q}_1}q_{\mathrm{r}_1}U_4\cos(\delta_{\mathrm{r}_1,0}-\theta_{1,0}) \\ &+ d^{\mathrm{I}}_{1,1}\vec{E}'_{\mathrm{q}_1}U_4\sin(\delta_{\mathrm{r}_1,0}-\theta_{1,0}) + d^{\mathrm{R}}_{1,1}\vec{E}'_{\mathrm{q}_1}q_{\mathrm{r}_1}U_4\sin(\delta_{\mathrm{r}_1,0}-\theta_{1,0})\Big)P^{\mathrm{DC}}_1 \\ &- \frac{1}{U_1U_4}\Big(d^{\mathrm{R}}_{1,4}\vec{E}'_{\mathrm{q}_1}U_1\cos(\delta_{\mathrm{r}_1,0}-\theta_{4,0}) - d^{\mathrm{I}}_{1,4}\vec{E}'_{\mathrm{q}_1}q_{\mathrm{r}_2}U_1\cos(\delta_{\mathrm{r}_1,0}-\theta_{4,0}) \\ &+ d^{\mathrm{I}}_{1,4}\vec{E}'_{\mathrm{q}_1}U_1\sin(\delta_{\mathrm{r}_1,0}-\theta_{4,0}) + d^{\mathrm{R}}_{1,4}\vec{E}'_{\mathrm{q}_1}q_{\mathrm{r}_2}U_1\sin(\delta_{\mathrm{r}_1,0}-\theta_{4,0})\Big)P^{\mathrm{DC}}_2 \end{aligned} \tag{5.13}$$

where the elements of $\boldsymbol{Y}_{\mathrm{DC}}$ are given by $\vec{d}_{i,j} = d^{\mathrm{R}}_{i,j} + \mathrm{j}d^{\mathrm{I}}_{i,j}$, q_{r_1} and q_{r_2} are the ratios of reactive to active power in HVDC links 1 and 2, respectively, and $\delta_{\mathrm{r}_a,0}$ and θ_{j_0} denote the initial conditions of the rotor angle of generator a and the bus angle at bus j, respectively. This gives $g_{1,1} = -\frac{1}{U_1U_4}(d^{\mathrm{R}}_{1,1}\vec{E}'_{\mathrm{q}_1}U_4\cos(\delta_{\mathrm{r}_1,0}-\theta_{1,0}) - d^{\mathrm{I}}_{1,1}\vec{E}'_{\mathrm{q}_1}q_{\mathrm{r}_1}U_4\cos(\delta_{\mathrm{r}_1,0}-\theta_{1,0}) + d^{\mathrm{I}}_{1,1}\vec{E}'_{\mathrm{q}_1}U_4\sin(\delta_{\mathrm{r}_1,0}-\theta_{1,0}) + d^{\mathrm{R}}_{1,1}\vec{E}'_{\mathrm{q}_1}q_{\mathrm{r}_1}U_4\sin(\delta_{\mathrm{r}_1,0}-\theta_{1,0}))$ and $g_{1,2} = -\frac{1}{U_1U_4}(d^{\mathrm{R}}_{1,4}\vec{E}'_{\mathrm{q}_1}U_1\cos(\delta_{\mathrm{r}_1,0}-\theta_{4,0}) - d^{\mathrm{I}}_{1,4}\vec{E}'_{\mathrm{q}_1}q_{\mathrm{r}_2}U_1\cos(\delta_{\mathrm{r}_1,0}-\theta_{4,0}) + d^{\mathrm{I}}_{1,4}\vec{E}'_{\mathrm{q}_1}U_1\sin(\delta_{\mathrm{r}_1,0}-\theta_{4,0}) + d^{\mathrm{R}}_{1,4}\vec{E}'_{\mathrm{q}_1}q_{\mathrm{r}_2}U_1\sin(\delta_{\mathrm{r}_1,0}-\theta_{4,0}))$. It should also be noted that the complex $\frac{\vec{E}}{\vec{U}}$ ratios are taken as constant in order to simplify the equations. This process is then repeated for each generator. The parameters that are used in the simulations of the multiple HVDC link system in this chapter are given in Table 5.1 with base terms $S_{\mathrm{base}} = 100 \times 10^6$ VA, $U_{\mathrm{base}} = 100 \times 10^3$ V, $f_{\mathrm{base}} = 50$ Hz, $w_0 = 2\pi f_{\mathrm{base}}$ rad/s.

Line	1	2	3	4
X_{L} pu	0.6	0.6	0.1	0.1
X_{S} pu	0.1	0.1	0.1	0.1
Generator	1	2	3	4
$x_{\mathrm{d}}^{'}$ pu	0.09	0.06	0.12	0.12
H (s)	2	4	2	2
D pu	1	1	1	1
$P_{\mathrm{G}} = P_{\mathrm{m}}$ pu	0.1	0.6	0.1	0.1
δ_{r_0} rad	5.9874	0.2871	5.585	5.03
$E_{\mathrm{q}}^{'}$ pu	0.4454	0.513	0.6807	1.0622
Bus	1	2	3	4
Load pu	0.1+0.05i	0.1+0.05i	0.1+0.05i	0.6+0.2759i
U pu	0.1097	0.2426	0.256	0.2219
θ rad	-0.4809	6.2768	5.5161	-1.3042

HVDC link a=	1	2
$P_{a,0}^{\mathrm{DC}}$ pu	0.3573	0.1427
q_{r_a}	0.8952	0.9037

Table 5.1: Multiple HVDC link system parameters.

5.3 Off-line Stochastic Optimisation of a Centralised PID Control Scheme Using PSO

An extension of the control strategy proposed in (Erikkson, 2008) for the control of the multiple HVDC link system can be seen in Fig. 5.4. The generator speed deviation between the AC connected generators is passed to a bandpass filter with the following transfer function:

$$G(s) = \frac{s/\omega_{\mathrm{c}}^2}{s^2 + Bs + \omega_{\mathrm{c}}^2} \tag{5.14}$$

where B is the -3dB bandwidth in rad/s and ω_{c} is the center frequency of the filter in rad/s. In the diagram $P_{1,0}^{\mathrm{DC}}$ and $P_{2,0}^{\mathrm{DC}}$ are the initial operating points for P_1^{DC} and P_2^{DC}, respectively. The bandpass filters are centered on the major oscillatory modes between buses 1 and 4 (ω_{14} rad/s) and buses 2 and 3 (ω_{32} rad/s). PID controllers are then used to modulate the HVDC links in order to dampen the oscillations in the AC line speed

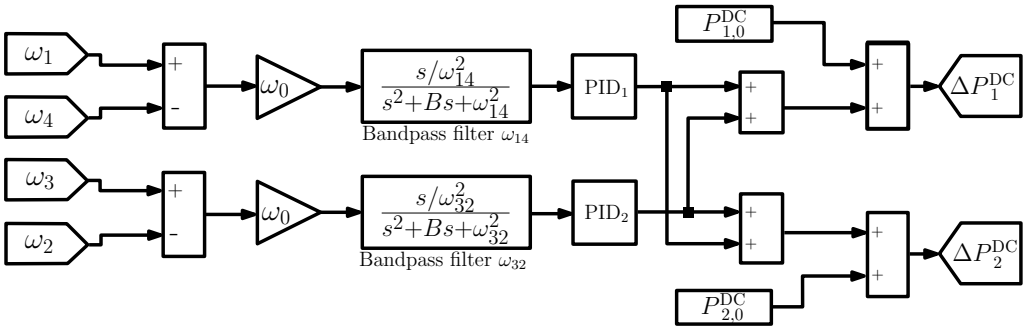


Figure 5.4: An extension of the control strategy used in (Erikkson, 2008) to control the multiple link HVDC system, which uses PID rather than P controllers

deviations at the major oscillatory mode.

Proportional controllers were initially proposed in (Erikkson, 2008) for the control of the multiple HVDC link system. The proportional gains and filter bandwidths were chosen based on trial and error, observing which combinations of values gave a good damping response. However there is the potential for improved control performance by using Proportional, Integral and Derivative (PID) controllers and then optimising both the controller gains and the filter bandwidths, based on simulation runs of the power network, using a suitable tuning criterion.

As simulations of the power network, which is represented by a non-linear model, are used to provide the input to the tuning criterion, the derivation of the relationship between the PID gains and the tuning criterion is non-trivial and so a derivative-free optimisation technique is proposed here for optimisation of the gains. Due to the non-convex nature of these tuning problems, it is desirable to use an optimisation technique that does not get trapped in local minima.

5.3.1 Stochastic Optimisation for Off-line Controller Tuning

Stochastic optimisation techniques use randomness as part of their optimisation procedures in order to search for optima. While they may not be as efficient as deterministic techniques, they are generally quite flexible in terms of the type of surface on which they can be used, they are robust to noise, and unlike most deterministic optimisation methods, stochastic optimsation methods generally do not require access to functional derivatives

(Weise, 2009). It is because they are not as efficient as deterministic techniques that these algorithms would typically be used for a-priori, or off-line tuning, of a set of system parameters, as opposed to the on-line tuning of parameters with deterministic techniques, as is performed on the system inputs when using MPC.

Many popular stochastic optimisation methods are derived from stochastic hill climbing (Weise, 2009). This very simple algorithm uses its current solution to produce a new solution randomly and chooses to update its position to the new position if it results in an improvement in the solution. Tabu search methods improve on this by keeping a list of previously visited areas and rejecting a move to this area if it has already been visited (Glover and Laguna, 1997). An issue with Hill Climbing and Tabu search methods is that they can get caught in local minima. One way of overcoming this is to allow random position restarts after a number of iterations in order to search the function space.

In many cases, such stochastic optimisation algorithms have drawn inspiration from nature. Simulated Annealing (SA) (van Laarhoven, 1987) emulates how crystalline structures found in nature cool to form solids. It is based on a process where at an initial high 'temperature' position updates are quite unrestricted in terms of searching the function being optimised. This is analogous to atoms in the annealing process being at a very high initial energy state. Then as the procedure progresses this temperature reduces, constricting the distance travelled between position update iterations. This is analagous to the decrease in the energy of the atoms as the crystal structure solidifies. Eventually at very low temperatures the optimisation process searches for a local minumum. Many other stochastic optimisation algorithms are similar to SA in that there is an initial exploration phase of the search, where updates can occur over a large area of the search space, followed by an exploitation phase where updated positions converge on a local minimum. These algorithms are often used to find global minima and so are often called Global Optimisation (GO) algorithms. Two other broad categories of GO algorithms derived from natural optimisation processes are Evolutionary Algorithms and Swarm Algorithms.

Most Evolutionary Algorithms (EAs) are based on biological phenomena found in genes. Crossover and mutations are used to update the genes and those genes with the best solutions to the problem survive to the next iteration. It is this general process that is used to evolve solutions in Genetic Algorithms (GAs) (Haupt and Haupt, 2004). A number of positions, called genes here, are evaluated to determine their fitness, which

is usually an evaluation of the function being optimised. Then the genes with the best fitnesses are allowed to continue to the next iteration and are encoded in a certain way to allow a crossover or mutation operation to be applied to them. For example, in binary encoding, positions are converted to binary bits and it is on these bits that the crossover and mutation operations are carried out. In crossover operations, two different candidate solutions exchange some of these bit positions. Mutation introduces random changes into bit positions and is the source of the stochastic element in this procedure. This process is then carried out in an iterative fashion and over a number of iterations an optimal solution evolves. This is the underlying processes of most EAs (Weise, 2009).

Swarm based algorithms are abstractions of processes that groups of animals use in nature for foraging or flocking. The two most common algorithms are Ant Colony Optimisation (ACO) and Particle Swarm Optimisation (PSO). ACO is based on the way in which ants forage for food. In ACO, candidate solutions, called ants, set out from a starting position walking in random positions. They leave behind them a trace, called a pheromone, which decays over time. Ants then choose to either follow paths with higher pheromone levels or pick a random direction based on a probability distribution. Therefore, over time the optimal solution is found based on the trail with the highest level of pheromone. Typically ACO is used for finding the shortest path over a graph (Dorigo and Blum, 2005). PSO is based on the flocking behaviour of swarm of birds foraging for food in nature. Candidate solutions, called particles, keep a record of their own previous best fitness and the best fitness found in the entire group. Their updated positions are then biased towards these positions. It is quite useful for general optimisation problems (Clerc and Kennedy, 2002) and will be examined in greater detail in the next section.

Stochastic search techniques such as SA, PSO, and GAs, have previously been quite successful in finding optimal controller gains (Kwok and Sheng, 1994; Gaing, 2004; Jones and De Moura Oliveira, 1995). While the use of GAs for the optimisation of controller gains is widespread, it has been noted that GAs' efficiency degrade when they are is used for the optimisation of highly epistatic functions, i.e., functions in which there is a high level of correlation between the parameters being optimised. PSO has been shown to outperform GAs when used for the optimisation of controller gains, in terms of both the quality of the solutions found and the time needed to find them (Gaing, 2004). Hence, PSO is used in this thesis for the optimisation of controller gains.

5.3.2 Particle Swarm Optimisation

As was stated previously PSO is a stochastic optimisation technique based on the social behaviour of swarms of flocking animals (Kennedy and Eberhart, 1995). It is suitable for the optimisation of convex, non-convex, continuous and discontinuous surfaces.

In PSO a population of P particles, each of dimension d, are initially distributed across the parameter space. The q^{th} particle at the i^{th} iteration of the PSO algorithm, has a position $\boldsymbol{x}_q(i)$, and associated cost $x_q^{\text{c}}(i)$. Each of these particles has a memory of its own previous best position $\boldsymbol{p}_q^{\text{b}}(i)$ and an associated cost $p_q^{\text{c,b}}(i)$. Here $p^{\text{g}}(i)$, the global best position, is the particle position associated with the best cost $p^{\text{c,g}}(i)$ that has been found previously across the population of particles. The q^{th} particle position is then updated, biased towards both the global best position and its previous best position. The PSO algorithm (in the case of the minimisation of a cost function) is as follows:

1. Initialise a population of P particles in d dimensions, within upper and lower bounds in each dimension, in the cost function space.
2. Evaluate the cost function values at each of the P particles' positions.
3. If for particle q, $x_q^{\text{c}}(i)<p_q^{\text{c,b}}(i-1)$, then let $\boldsymbol{p}_q^{\text{b}}(i)=\boldsymbol{x}_q(i)$.

   If $x_q^{\text{c}}(i)<p^{\text{c,g}}(i-1)$, let $\boldsymbol{p}^{\text{g}}(i)=\boldsymbol{x}_q(i)$.

   If $x_q^{\text{c}}(i)\geq p_q^{\text{c,b}}(i-1)\geq p^{\text{c,g}}(i-1)$, then $\boldsymbol{p}_q^{\text{b}}(i)$ and $\boldsymbol{p}^{\text{g}}(i)$ remain at the same positions as in iteration $i-1$.
4. The velocity, $\boldsymbol{v}_{\text{q}}(i)$, and position $\boldsymbol{x}_q(i)$, of particle q at the i^{th} iteration of the PSO algorithm are updated for the next iteration as follows:

$$\begin{aligned}\boldsymbol{v}_q(i+1) = \omega\boldsymbol{v}_q(i) &+ c_1\boldsymbol{r}_1(i)\circ\left(\boldsymbol{p}_q^{\text{b}}(i)-\boldsymbol{x}_q(i)\right)\\ &+ c_2\boldsymbol{r}_2(i)\circ\left(\boldsymbol{p}^{\text{g}}(i)-\boldsymbol{x}_q(i)\right)\end{aligned} \tag{5.15}$$

$$\boldsymbol{x}_q(i+1) = \boldsymbol{x}_q(i) + \boldsymbol{v}_{q_{\text{app}}}(i+1) \tag{5.16}$$

   where $\circ$ denotes the Schur product, $\boldsymbol{r}_1(i)$ and $\boldsymbol{r}_2(i)$ are random vectors with entries uniformly distributed in the interval [0,1], the positive scalar ω is the inertial weight which controls the exploration and exploitation in the search space, c_1 and c_2 are

acceleration constants called the cognition and social components, respectively, and $\boldsymbol{v}_{q_{\text{app}}}$ is the applied particle velocity.

Applied particle velocities are bounded by $\boldsymbol{v}_{q_{\text{app}}} \geq \boldsymbol{v}_{q_{\min}}$, $\boldsymbol{v}_{q_{\text{app}}} \leq \boldsymbol{v}_{q_{\max}}$ where $\boldsymbol{v}_{q_{\min}}$ and $\boldsymbol{v}_{q_{\max}}$ are the lower and upper bounds on particle velocities, respectively. If updated velocities exceed the bounds, the applied velocity, $\boldsymbol{v}_{q_{\text{app}}}$, is taken at the upper or lower bound, i.e., if $\boldsymbol{v}_q < \boldsymbol{v}_{q_{\min}}$, let $\boldsymbol{v}_{q_{\text{app}}} = \boldsymbol{v}_{q_{\min}}$; if $\boldsymbol{v}_q > \boldsymbol{v}_{q_{\max}}$, let $\boldsymbol{v}_{q_{\text{app}}} = \boldsymbol{v}_{q_{\max}}$; else let $\boldsymbol{v}_{q_{\text{app}}} = \boldsymbol{v}_q$.

5. Repeat steps (2)-(4) until certain termination criteria are performed, e.g., a maximum amount of iterations are met, $\boldsymbol{p}^{\text{g}}(i)$ has not changed for a given number of iterations, etc.

In (Trelea, 2003) it was shown that good convergence properties could be obtained for the PSO, using the following parameter selection; $\omega = 0.6$, $c_1 = c_2 = 1.7$. These parameter values were selected in this thesis so as to give a good convergence performance on the final set of values chosen by the PSO.

5.3.3 PSO Optimisation of Multiple Link HVDC Control System Parameters

PSO is used to optimise the PID gains and filter bandwidths. A PSO particle is made to correspond to the 8 dimensional vector of the controller parameters. These parameters are the 2 sets of PID gains and the 2 bandpass filter bandwidths. Each particle runs a simulation of the multiple link HVDC system with the PID controller, using its current position to determine the controller gains and bandwidths. The particle cost is obtained by recording the generator frequencies over the course of the simulation and passing these to an evaluation criterion. The PSO is then allowed to run as described previously.

The criterion used here to evaluate the cost for a simulation run is the integral of the square of the error (ISE) (Gaing, 2004) given by:

$$\text{ISE} = \sum_{a=1}^{n} \sum_{k=1}^{k_{\text{f}}} (\omega_{\text{r}_a}(k) - 1)^2, \tag{5.17}$$

where there are n agents, and k_{f} is the number of samples in one system simulation run.

The ISE is used to reward behaviours that minimise errors over the simulation, with penalties increasing quadratically as the errors increase in magnitude. This criterion is used as it is desirable to minimise the maximum size of frequency deviations in power systems.

The PID parameters and bandwidths used in the controller in Fig. 5.4 were optimised using the PSO Toolbox (Birge, 2003). The parameters used in the toolbox are given in Table 5.2. Before optimisation of the controller parameters, the major oscillatory modes were found at ω_{14}=6.2118 rad/s and ω_{32}=4.0285 rad/s using eigenvalue analysis. These were then used as the centre frequencies for the bandpass filters.

Parameter	Description	
p	number of particles	50
mvden	max. velocity divisor	2
errgrad	error gradient tolerance	1e-5
epoch	maximum number of iterations	2000
errgraditer	number of epochs without errgrad change before termination	9

Table 5.2: Parameters used in PSO toolbox for controller gain optimisations.

50 particles were initially placed randomly across the 8-dimensional plane being optimised. Particle fitness was based on the sum of the ISE for two different faults scenarios: a 100 ms fault is applied followed by a 100 ms line break applied to lines 1 and 3 separately (the lines return to the non-fault state after the line break is finished). This ensured that the optimisation considers equally faults that happen on both sides of the HVDC links. Simulation runs lasted for 20 seconds in each case.

The optimal PID and bandwidth values found using PSO, with initial position [B_{14}, B_{23}, K_{14}, K_{23}, I_{14}, I_{23}, D_{14}, D_{23}]=[20 20 20 20 0 0 0 0], were B_{14}=2.7 rad/s, B_{23}=17.4 rad/s, K_{14}=18.9, K_{23}=460, I_{14}=3702, I_{23}=1190, D_{14}=56.2, D_{23}=0, where B_a is the filter bandwidth, K_a is the proportional gain, I_a is the integral gain, and D_a is the derivative gain used in the controller for area a.

5.4 Design of the Centralised and Distributed MPC

The state-space model based used with the centralised and distributed MPC is based on a linearisation of (5.1) and (5.9). At each sample the state equations for each generator are linearised about the current operating point as follows:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\delta_{\mathrm{r}_a}(t)\\ \omega_{\mathrm{r}_a}(t)\end{bmatrix} &= \begin{bmatrix}0 & \omega_0\\ \frac{\partial f_{\mathrm{r}_a}}{\partial \delta_{\mathrm{r}_a}}|_{\mathrm{op}} & \frac{\partial f_{\mathrm{r}_a}}{\partial \omega_{\mathrm{r}_a}}|_{\mathrm{op}}\end{bmatrix}\begin{bmatrix}\delta_{\mathrm{r}_a}(t)\\ \omega_{\mathrm{r}_a}(t)\end{bmatrix}\\ &+ \begin{bmatrix}0 & 0\\ \frac{\partial f_{\mathrm{r}_a}}{\partial P_1^{\mathrm{DC}}}|_{\mathrm{op}} & \frac{\partial f_{\mathrm{r}_a}}{\partial P_2^{\mathrm{DC}}}|_{\mathrm{op}}\end{bmatrix}\begin{bmatrix}P_1^{\mathrm{DC}}(t)\\ P_2^{\mathrm{DC}}(t)\end{bmatrix} + \begin{bmatrix}0\\ \frac{\partial f_{\mathrm{r}_a}}{\partial \delta_{\mathrm{r}_l}}|_{\mathrm{op}}\end{bmatrix}\delta_{\mathrm{r}_l}(t)\end{aligned} \tag{5.18}$$

where in the above equation $f_{\mathrm{r}_a}(\delta_{\mathrm{r}_a}(t), \omega_{\mathrm{r}_a}(t), P_1^{\mathrm{DC}}(t), P_2^{\mathrm{DC}}(t), \delta_{\mathrm{r}_l}(t)) = \frac{\mathrm{d}}{\mathrm{d}t}\omega_{\mathrm{r}_a}(t)$, as defined in (5.9), and op indicates the linearisation of the relevant variable, vector, or function about the current operating point.

For centralised MPC the states are taken as $\boldsymbol{x}$=[δ_{r_1}, $\omega_{\mathrm{r}_1}, \ldots, \delta_{\mathrm{r}_4}$, ω_{r_4},]$^{\mathrm{T}}$, and the inputs as $\boldsymbol{u}$=[P_1^{DC} P_2^{DC}]$^{\mathrm{T}}$. For distributed MPC the states of agent a are taken as $\boldsymbol{x}_a$=[δ_{r_a} ω_{r_a}]$^{\mathrm{T}}$, the inputs $\boldsymbol{u}_a$=[P_1^{DC} P_2^{DC}]$^{\mathrm{T}}$, and the interconnecting input $\boldsymbol{v}_a$=δ_{r_l}. The full system model for the centralised case is discretised using a zero-order hold with a sample time τ = 0.01s, providing the discrete-time state space equations for the centralised and distributed MPC systems. Predictions in both centralised and distributed cases are formed using incremental state space models so as to ensure integral action, i.e., the augmented state $\boldsymbol{x}_{\mathrm{aug}} = [\Delta\boldsymbol{x}^{\mathrm{T}}, \boldsymbol{x}^{\mathrm{T}}]^{\mathrm{T}}$, incremental inputs $\Delta\boldsymbol{u}$ and $\Delta\boldsymbol{u}_a$, and incremental interconnecting inputs $\Delta\boldsymbol{v}_a$ and their associated state space models are used for predictions and optimisations. A prediction horizon of N=50 is used so as to accurately represent the system dynamics in the optimisation.

Each subsystem a's stage cost function (in the multi-agent case there is one agent for each generator, so for convenience the subscript a is used to index both), $J_a^{\mathrm{stage}}(k,p)$, for the p^{th} prediction step at sample step k, is given as follows:

$$J_a^{\mathrm{stage}}(k,p) = Q_a(\omega_{\mathrm{r}_a}(k+p) - 1)^2, \tag{5.19}$$

where Q_a is the weight corresponding to ω_{r_a} in the cost function. This cost function penalises deviations of the frequency from the base frequency. The centralised MPC

optimisation problem is then given by:

$$J(k) = \sum_{p=1}^{N} \sum_{a=1}^{n} J_a^{\text{stage}}(k,p). \tag{5.20}$$

The weights $[Q_1, \ldots, Q_4]$=[10,30,10,10] are used for both MPC cases.

5.4.1 Application of Distributed MPC with Shared Inputs Between Agents

In typical control applications, agents have their own local control inputs which are not shared between agents. However, in the application in this chapter, all 4 agents must determine actions for the 2 control inputs, $P_1^{\text{DC}}(k)$ and $P_2^{\text{DC}}(k)$. In other circumstances different agents' local inputs may be coupled, for example, via the objective function or through the system dynamics.

The distributed MPC algorithm, described in the previous chapter, naturally extends to such cases in the following way. Agents create a duplicate variable vector, $\tilde{\boldsymbol{w}}_a^{\text{u}}(k)$, for agent a, of the control inputs, $\tilde{\boldsymbol{u}}(k)$, and then try to form consensus on these duplicate variables. These duplicate variables are then treated as local control inputs by each of the agents. Equality constraints are then placed on the duplicate variables as follows $\tilde{\boldsymbol{w}}_1^{\text{u}}(k)$=$\tilde{\boldsymbol{w}}_2^{\text{u}}(k)$, $\tilde{\boldsymbol{w}}_2^{\text{u}}(k)$=$\tilde{\boldsymbol{w}}_3^{\text{u}}(k)$,..., $\tilde{\boldsymbol{w}}_{n-1}^{\text{u}}(k)$=$\tilde{\boldsymbol{w}}_n^{\text{u}}(k)$, such that $\tilde{\boldsymbol{w}}_1^{\text{u}}(k)$=...=$\tilde{\boldsymbol{w}}_n^{\text{u}}(k)$ for a system of n subsystems. When the problem is distributed amongst agents, then each agent will optimise to find the local duplicate inputs. Agents then compare their local duplicate inputs to the values calculated previously by connected agents in order to achieve consensus, in the same way that agents compare other interconnecting variables. Each agent's final $\tilde{\boldsymbol{w}}_a^{\text{u}}(k)$ value will differ slightly from that of the other agents, depending on the values of c and ϵ, as these determine to what extent agents will form a consensus on variables. The control engineer must decide at the design stage which agent will ultimately decide on the value of the input to be applied to the real system being controlled, from the inputs calculated separately by each agent.

The interconnection cost for the distributed MPC case at sample step k and iteration l of the control cycle, $J_a^{\text{inter}}(k,l)$, is formed from a hypothetical centralised augmented

Lagrangian MPC formulation which is given as follows:

$$\begin{aligned}\min_{\Delta\tilde{\boldsymbol{u}}_1,\ldots,\Delta\tilde{\boldsymbol{u}}_4} \sum_{a=1}^{4}\left(J_a^{\text{local}}(k,l)\right) \quad &+ \begin{bmatrix}\tilde{\boldsymbol{\lambda}}_{41}^{\text{in},\boldsymbol{x}_4}(k,l)\\ \tilde{\boldsymbol{\lambda}}_{32}^{\text{in},\boldsymbol{x}_3}(k,l)\\ \tilde{\boldsymbol{\lambda}}_{23}^{\text{in},\boldsymbol{x}_2}(k,l)\\ \tilde{\boldsymbol{\lambda}}_{14}^{\text{in},\boldsymbol{x}_1}(k,l)\\ \tilde{\boldsymbol{\lambda}}_{41}^{\text{u}}(k,l)\\ \tilde{\boldsymbol{\lambda}}_{12}^{\text{u}}(k,l)\\ \tilde{\boldsymbol{\lambda}}_{23}^{\text{u}}(k,l)\\ \tilde{\boldsymbol{\lambda}}_{34}^{\text{u}}(k,l)\end{bmatrix}^{\text{T}}\begin{bmatrix}\tilde{\boldsymbol{w}}_{41}^{\text{in},\boldsymbol{x}_4}(k,l)-\tilde{\boldsymbol{w}}_{14}^{\text{out},\boldsymbol{x}_4}(k,l)\\ \tilde{\boldsymbol{w}}_{32}^{\text{in},\boldsymbol{x}_3}(k,l)-\tilde{\boldsymbol{w}}_{23}^{\text{out},\boldsymbol{x}_3}(k,l)\\ \tilde{\boldsymbol{w}}_{23}^{\text{in},\boldsymbol{x}_2}(k,l)-\tilde{\boldsymbol{w}}_{32}^{\text{out},\boldsymbol{x}_2}(k,l)\\ \tilde{\boldsymbol{w}}_{14}^{\text{in},\boldsymbol{x}_1}(k,l)-\tilde{\boldsymbol{w}}_{41}^{\text{out},\boldsymbol{x}_1}(k,l)\\ \tilde{\boldsymbol{w}}_1^{\text{u}}(k,l)-\tilde{\boldsymbol{w}}_4^{\text{u}}(k,l)\\ \tilde{\boldsymbol{w}}_2^{\text{u}}(k,l)-\tilde{\boldsymbol{w}}_1^{\text{u}}(k,l)\\ \tilde{\boldsymbol{w}}_3^{\text{u}}(k,l)-\tilde{\boldsymbol{w}}_2^{\text{u}}(k,l)\\ \tilde{\boldsymbol{w}}_4^{\text{u}}(k,l)-\tilde{\boldsymbol{w}}_3^{\text{u}}(k,l)\end{bmatrix}\\ &+\frac{c}{2}\left\|\begin{matrix}\tilde{\boldsymbol{w}}_{41}^{\text{in},\boldsymbol{x}_4}(k,l)-\tilde{\boldsymbol{w}}_{14}^{\text{out},\boldsymbol{x}_4}(k,l)\\ \tilde{\boldsymbol{w}}_{32}^{\text{in},\boldsymbol{x}_3}(k,l)-\tilde{\boldsymbol{w}}_{23}^{\text{out},\boldsymbol{x}_3}(k,l)\\ \tilde{\boldsymbol{w}}_{23}^{\text{in},\boldsymbol{x}_2}(k,l)-\tilde{\boldsymbol{w}}_{32}^{\text{out},\boldsymbol{x}_2}(k,l)\\ \tilde{\boldsymbol{w}}_{14}^{\text{in},\boldsymbol{x}_1}(k,l)-\tilde{\boldsymbol{w}}_{41}^{\text{out},\boldsymbol{x}_1}(k,l)\\ \tilde{\boldsymbol{w}}_1^{\text{u}}(k,l)-\tilde{\boldsymbol{w}}_4^{\text{u}}(k,l)\\ \tilde{\boldsymbol{w}}_2^{\text{u}}(k,l)-\tilde{\boldsymbol{w}}_1^{\text{u}}(k,l)\\ \tilde{\boldsymbol{w}}_3^{\text{u}}(k,l)-\tilde{\boldsymbol{w}}_2^{\text{u}}(k,l)\\ \tilde{\boldsymbol{w}}_4^{\text{u}}(k,l)-\tilde{\boldsymbol{w}}_3^{\text{u}}(k,l)\end{matrix}\right\|_2^2,\end{aligned} \tag{5.21}$$

where the local cost for subsystem a, $J_a^{\text{local}}(k)$, is given by:

$$J_a^{\text{local}}(k) = \sum_{p=1}^{N} J_a^{\text{stage}}(k,p). \tag{5.22}$$

This formulation enables the distribution of the problem so that agents can reach agreement on the control inputs, i.e., the HVDC powers. Each agent a has a duplicate vector of the control inputs $\tilde{\boldsymbol{w}}_a^{\text{u}}(k)$ where $\boldsymbol{w}_a^{\text{u}}(k) = [P_1^{\text{DC}}(k)\ P_2^{\text{DC}}(k)]^{\text{T}}$. The order in which agents optimise for the distributed MPC cycles starts with agent 1 and ends with 4, as can be seen in Fig. 5.5. Therefore in the hypothetical centralised augmented Lagrangian case, the equality constraint $\tilde{\boldsymbol{w}}_a^{\text{u}}(k) = \tilde{\boldsymbol{w}}_{a,\text{last}}^{\text{u}}(k)$ is applied for each agent ($w_{a,\text{last}}^{\text{u}}$ denotes the last agent to optimise) in order to reach consensus on the duplicate input values. Interconnecting constraints between interconnecting state variables are also applied.

When (5.21) is distributed amongst the agents, $J_a^{\text{inter}}(k,l)$ takes the following distributed

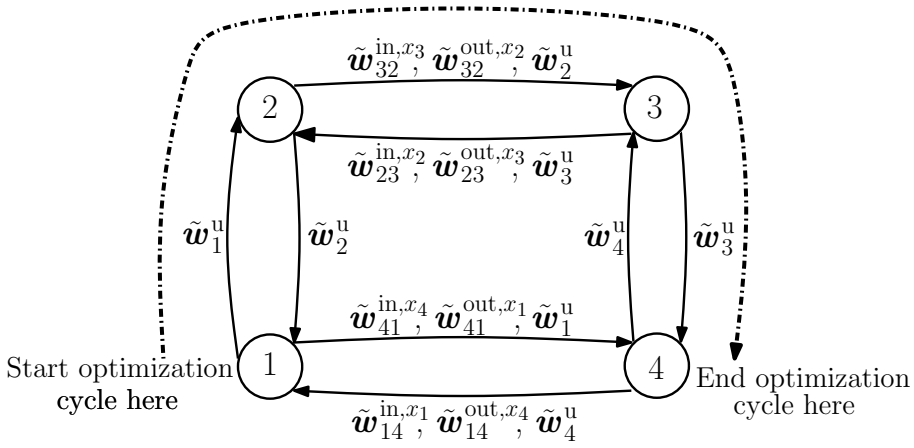


Figure 5.5: Order of serial distributed MPC optimizations and variables communicated between agents

form for agent a, where bus j is AC-connected to bus a:

$$J_a^{\text{inter}}(k,l) = \begin{bmatrix} \tilde{\boldsymbol{\lambda}}_{ja}^{\text{in},\boldsymbol{x}_j}(k,l) \\ -\tilde{\boldsymbol{\lambda}}_{aj}^{\text{in},\boldsymbol{x}_a}(k,l) \\ \tilde{\boldsymbol{\lambda}}_{a}^{\text{u}}(k,l) \\ -\tilde{\boldsymbol{\lambda}}_{a,\text{next}}^{\text{u}}(k,l) \end{bmatrix}^{\text{T}} \begin{bmatrix} \tilde{\boldsymbol{w}}_{ja}^{\text{in},\boldsymbol{x}_j}(k,l) \\ \tilde{\boldsymbol{w}}_{ja}^{\text{out},\boldsymbol{x}_a}(k,l) \\ \tilde{\boldsymbol{w}}_{a}^{\text{u}}(k,l) \\ \tilde{\boldsymbol{w}}_{a}^{\text{u}}(k,l) \end{bmatrix} + \frac{c}{2} \left\| \begin{matrix} \tilde{\boldsymbol{w}}_{aj,\text{prev}}^{\text{out},\delta_{\text{r}_j}}(k,l) - \tilde{\boldsymbol{w}}_{ja}^{\text{in},\boldsymbol{x}_j}(k,l) \\ \tilde{\boldsymbol{w}}_{aj,\text{prev}}^{\text{in},\boldsymbol{x}_a}(k,l) - \tilde{\boldsymbol{w}}_{ja}^{\text{out},\boldsymbol{x}_a}(k,l) \\ \tilde{\boldsymbol{w}}_{\text{last,prev}}^{\text{u}}(k,l) - \tilde{\boldsymbol{w}}_{a}^{\text{u}}(k,l) \\ \tilde{\boldsymbol{w}}_{\text{next,prev}}^{\text{u}}(k,l) - \tilde{\boldsymbol{w}}_{a}^{\text{u}}(k,l) \end{matrix} \right\|_2^2 \tag{5.23}$$

where $w_{a,\text{next}}^{\text{u}}$ denotes the next agent to optimise. The distributed MPC parameters related to communication are given as follows: $c = 0.1$, $\epsilon = 10^{-2}$.

After agent a has completed its optimisation, it sends the relevant updated values of the variables to the agents that are connected to it, for use in their distributed MPC optimisations. The final optimisation problem for agent a is given by:

$$\boldsymbol{\vartheta}_a(k,l) = \arg\min_{\boldsymbol{\vartheta}_a} \left(J_a^{\text{local}}(k) + J_a^{\text{inter}}(k,l) \right), \tag{5.24}$$

This can be put into quadratic form using simple matrix manipulation, where the optimisation vector is $\boldsymbol{\vartheta}_a(k,l) = [\Delta\tilde{\boldsymbol{u}}^{\text{T}}(k), \Delta\tilde{\boldsymbol{w}}_{\text{in}}^{\text{T}}(k)]^{\text{T}}$.

In the centralised MPC case, the optimal values calculated for $P_1^{\text{DC}}(k)$ and $P_2^{\text{DC}}(k)$ are applied to the system. However, unlike the centralised case the 4 agents in the distributed MPC system calculate slightly different values for the HVDC powers to each other, as these powers only have to match to a degree, determined by the distributed MPC parameters c and ϵ. Therefore, one agent per HVDC link a is assigned to apply its calculated $P_a^{\text{DC}}(k)$ value to the system at sample k. Here the values for $P_1^{\text{DC}}(k)$ and $P_2^{\text{DC}}(k)$, calculated by

agents 2 and 3 respectively, are the control inputs that are applied (these were chosen as the vast majority of power transfer is from agents 2 and 3 to agents 1 and 4, and so it is assumed these agents insist on having the final say on what power is allowed to be transferred to agents 1 and 4).

5.5 Results

The PSO-optimised PID-based control scheme of Section 5.3, and the centralised and distributed MPC controllers of Section 5.4, were used to control the multiple HVDC link system. Simulink was used to simulate the non-linear, continuous-time power system, using the Dormand-Prince (ode45 in Matlab) continuous-time algorithm, with a maximum step size of 2ms and a relative tolerance of 0.001. The non-linear equations used to simulate the system were based on equations (5.1) and (5.9) derived in Section 5.2. The inputs were calculated and applied at fixed time steps of 10ms using Matlab and the calculated inputs were passed to the continuous-time Simulink simulation. All MPC optimisations were performed using the Matlab function **quadprog**.

For the simulations involving centralised controllers (the centralised MPC and PSO-optimized PID), a single control agent was used to control the whole system. However, as was stated in Section 5.2, it may not be possible here to use a centralised controller as each country may have its own controller and within countries with deregulated power markets, sections of the grid may have different control agents that are responsible for the control of different sections of the grid.

For the decentralised and distributed control systems it is assumed here that both agents 1 and 4 in Denmark are run by two separate controllers due to deregulation of that market, with one controller each for Norway and Sweden (or indeed those particular sections of the grid in those countries). Therefore, 4 different agents are responsible for these 4 different areas. In the decentralised approach the agents take a greedy approach and do not try to obtain consensus on the shared inputs between the different areas, whereas in the distributed approach the adjacent agents communicate with each other as in Fig. 5.5.

In the multi-agent case one agent was assigned per generator to control its frequency.

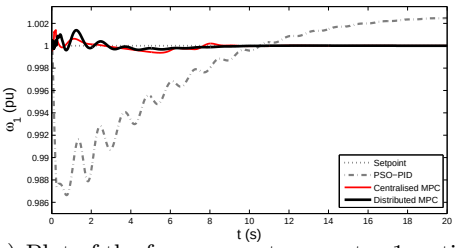
Each agent had access to its relevant state space model, the constraints on its variables, and could communicate with agents to which it was connected by an AC or HVDC link. The HVDC link ranges are $-2 \leq P^{\mathrm{DC}}(k) \leq 2$ pu and the frequency range at all buses is $0.984 \leq \omega(k) \leq 1.016$ pu. These constraints were applied over the full prediction horizon in the centralised and distributed MPC simulations.

Two simulations were run in which an AC line was subjected to a 100ms line fault, followed by 100ms line break, and then returned to its original non-fault state. In the first simulation this fault scenario was applied to line 1, and it was applied to line 3 in the second simulation. The results of the simulations can be seen in Figs. 5.6 and 5.7, which show the frequencies of each generator, the applied HVDC powers, and the number of distributed MPC iterations needed at each sample step, plotted against time for each control system, for the first and second simulations, respectively.

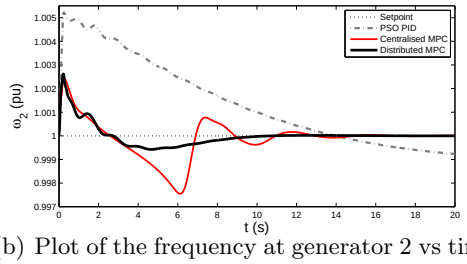
First of all it can be seen that the decentralised control gives by far the worst performance (simulations are terminated due to excessive constraint violations) and that at least some level of communication is necessary between agents to control this system. This becomes apparent from Fig. 5.7(h). Agents 2 and 3 are responsible for applying the final control inputs to the system and so are not affected by the line fault on line 3 and hence these agents remain at 1 pu. Therefore they do not change the control inputs to help stabilise the frequencies of generators 1 and 4. However, even in Fig. 5.6(h) when they are affected by the line fault in line 1 they are not able to satisfactorily restabilise the system.

It can be seen in Table 5.3 that the distributed MPC yields the best performance in an ISE sense, followed by the centralised MPC, and finally the PSO-optimised PID performance scheme. The MPC strategies do not experience the large unacceptable deviations from the setpoint experienced by the PID controller in Fig. 5.7, which violate the constraints on the maximum allowable frequency deviations. The MPC strategies can also be seen to have improved the system damping. Of note, in the presented scenario, is the fact that due to limited horizons and discrepancies between the real world and MPC models, the distributed MPC performs outperforms the centralised one.

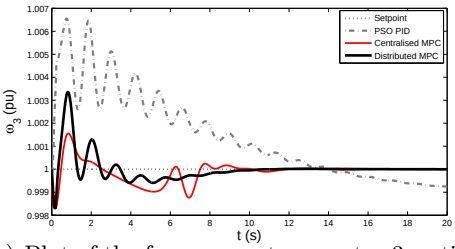
The trade-off experienced by the centralised and distributed MPC controllers for better disturbance rejection over the PID control scheme is a significant computational overhead in both cases, and a communications overhead in the case of the distributed MPC. The



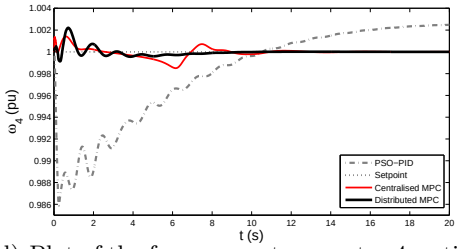
(a) Plot of the frequency at generator 1 vs time.



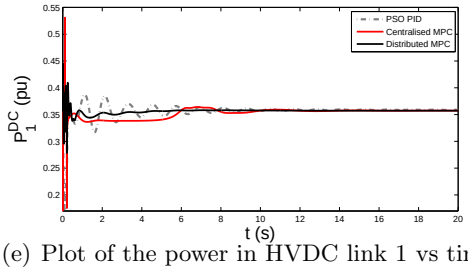
(b) Plot of the frequency at generator 2 vs time.



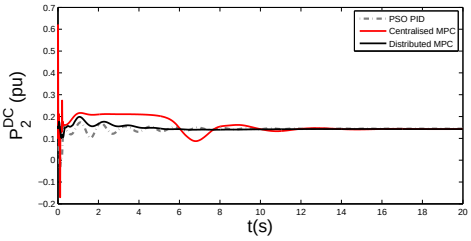
(c) Plot of the frequency at generator 3 vs time.



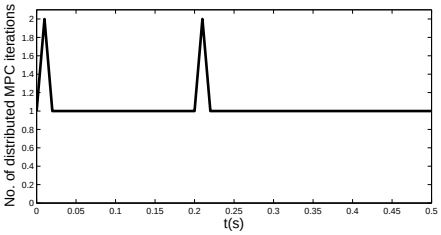
(d) Plot of the frequency at generator 4 vs time.



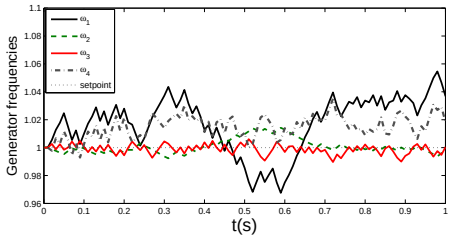
(e) Plot of the power in HVDC link 1 vs time.



(f) Plot of the power in HVDC link 2 vs time.



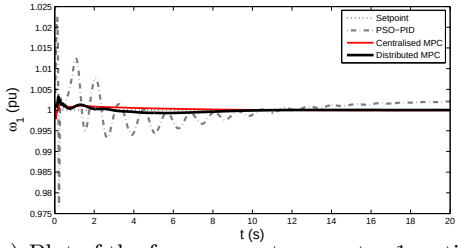
(g) Plot of the first 0.5 seconds of distributed MPC iterations vs time (iterations stay at 1 for remainder of simulation).



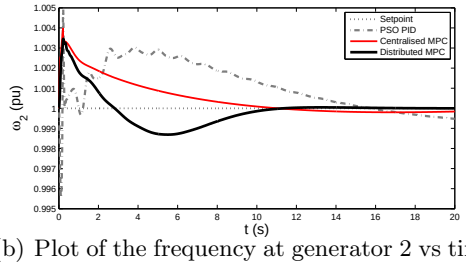
(h) Plot of the generator frequencies vs time for decentralised MPC (simulation stopped at t=1 s due to constraints violations).

Figure 5.6: Plots of pu frequency and HVDC powers vs. time for a 100ms fault followed by a 100ms line break applied to line 1.

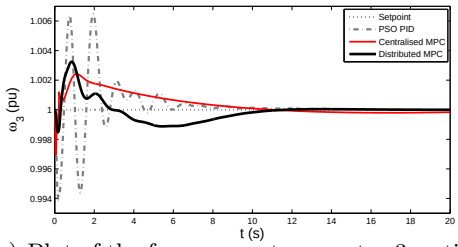
average and longest times taken to compute the control inputs for a centralised MPC cycle were 0.47 s and 1.125 s, respectively, and the average and longest times taken to converge on final solution for a distributed MPC cycle were 0.82 s and 1.8 s, respectively, on a computer with an Intel® Core™ 2 6400 operating at 2.13 GHz and with 3 GB of RAM (These times were taken as the time from the linearisation of the state space to



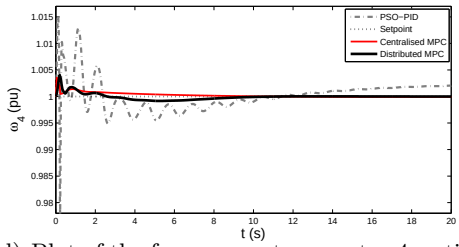
(a) Plot of the frequency at generator 1 vs time.



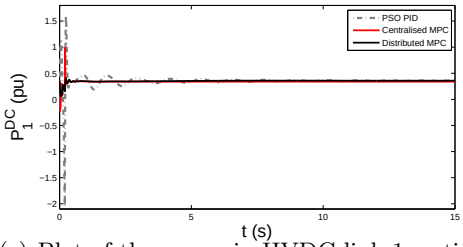
(b) Plot of the frequency at generator 2 vs time.



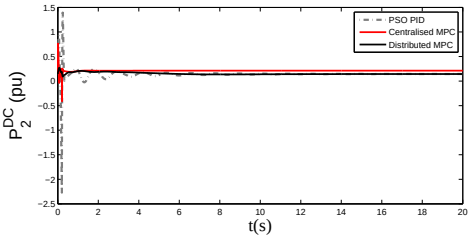
(c) Plot of the frequency at generator 3 vs time.



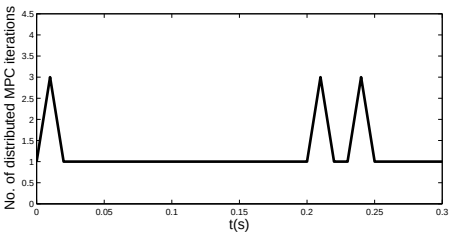
(d) Plot of the frequency at generator 4 vs time.



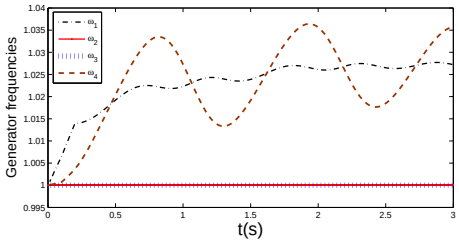
(e) Plot of the power in HVDC link 1 vs time.



(f) Plot of the power in HVDC link 2 vs time.



(g) Plot of the first 0.3 seconds of distributed MPC iterations vs time (iterations stay at 1 for remainder of simulation).



(h) Plot of the generator frequencies vs time for decentralised MPC (simulation stopped at t=3 s due to constraints violations).

Figure 5.7: Plots of pu frequency and HVDC powers vs. time for a 100ms fault followed by a 100ms line break applied to line 3.

the application of the control inputs, measured using the cputime command in Matlab. The actual time for a multi-core processor is roughly equal to the cputime divided by the number of cores). It should be noted that the distributed MPC simulation was also performed on an single PC, rather than several different PCs as would be the case for a real implementation.

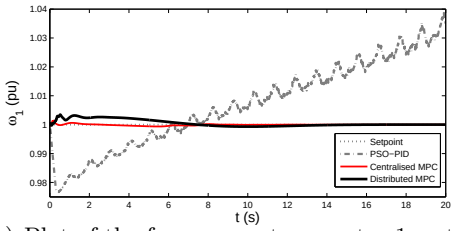
Fault on Line	PSO-PID	MPC	
		Centralised	Distributed
1	0.1161	0.0022	0.0015
3	0.0582	0.0042	0.0037
Total	0.1743	0.0066	0.0052

Table 5.3: Comparison of ISE for PSO-PID and MPC schemes

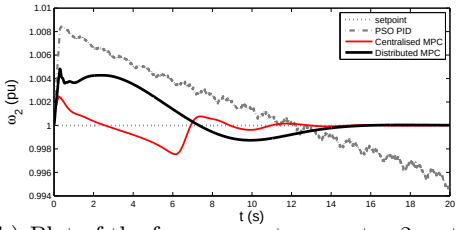
It can be seen from these results that the computational effort needed for the distributed MPC problem is larger than that of the centralised MPC, as well as having an added communications overhead. The number of distributed MPC iterations necessary to complete each optimisation cycle at each sample in each simulation, which represents the level of communication necessary, are given in Figs. 5.6(g) and 5.7(g). However, this centralised MPC problem is still relatively small. For larger power systems the centralised MPC problem would become increasingly intractable from a computational point of view, whereas the distributed MPC problems seen by each agent would stay the same size. However, for distributed MPC problems the amount of communication necessary between agents would increase with the size of the problem. Also, it should again be noted that there are situations such as those depicted in this chapter, where a multi-agent approach is desirable due to a number of separate controllers controlling different subsystems, where it may not be possible to adopt a centralised control approach.

In general, the techniques developed in (Erikkson, 2008) give acceptable performance for fault scenarios of less than 100ms in duration, and while PID control results in the exceeding of the desired upper and lower frequency bounds in Figs. 5.6 and 5.7 it still provides stabilising control. However, in Fig. 5.8 a 200ms line break follows the 100ms fault and the PID controller gains are optimised using this fault in the same way they were previously optimised for the 100ms line break fault. Here it can be seen that even when optimised for this fault that the PID controller cannot maintain stability for the system. From these plots it could be concluded that for serious faults the PID controller does not provide adequate closed loop performance, unlike the distributed MPC.

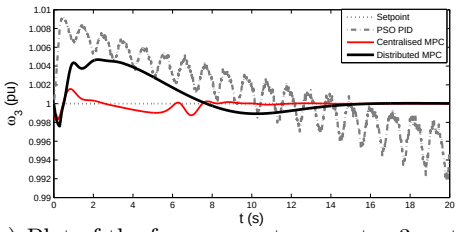
With regards to disturbance rejection and stability performance in a larger power network, one could expect that for disturbances of a similar size, that the distributed MPC would



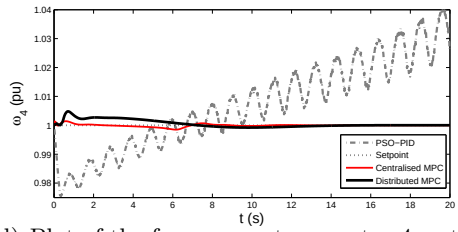
(a) Plot of the frequency at generator 1 vs time for fault scenario applied to line 1.



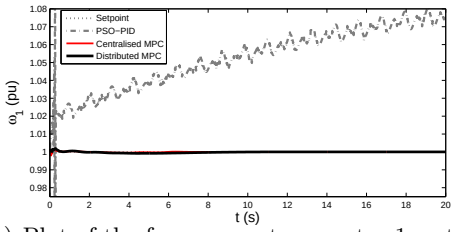
(b) Plot of the frequency at generator 2 vs time for fault scenario applied to line 1.



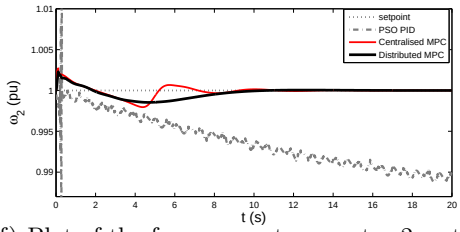
(c) Plot of the frequency at generator 3 vs time for fault scenario applied to line 1.



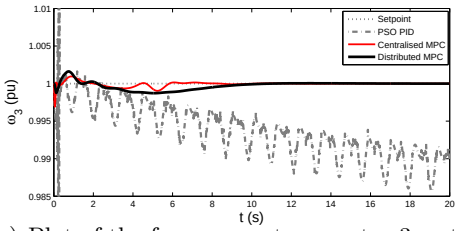
(d) Plot of the frequency at generator 4 vs time for fault scenario applied to line 1.



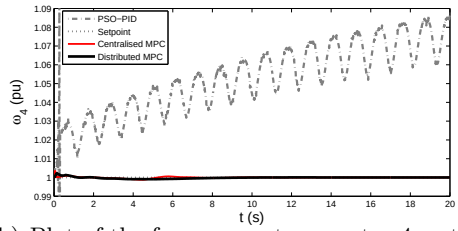
(e) Plot of the frequency at generator 1 vs time for fault scenario applied to line 3.



(f) Plot of the frequency at generator 2 vs time for fault scenario applied to line 3.



(g) Plot of the frequency at generator 3 vs time for fault scenario applied to line 3.



(h) Plot of the frequency at generator 4 vs time for fault scenario applied to line 3.

Figure 5.8: Plots of pu frequency and HVDC powers vs. time for a 100ms fault followed by a 200ms line break applied to lines 1 and 3 separately.

continue to provide satisfactory control. However, it is possible that with larger disturbances and a larger number of agents, that there would be some performance degradation. The same would be expected of centralised MPC, though.

In the example in this chapter, generation capacities are kept constant and the modulation

of the HVDC links alone is used to restabilise the system. This system is therefore a useful testbed for demonstrating the capabilities of HVDC links alone to stabilise systems. System performance could potentially be further improved by taking into account varying generator capacities.

5.6 Conclusions

Here, the application of a Particle Swarm Optimisation (PSO) optimised PID controller, and a centralised and distributed Model Predictive Control (MPC) controller to a multiple High Voltage Direct Current (HVDC) link system has been discussed. The distributed MPC gives the best result, followed by the centralised MPC, and then the PSO optimised PID controller. It has been seen that decentralised control is not capable of stabilising the system. As centralised MPC problems can get quite large for power systems, and given the improvement in performance associated with distributed MPC over the PSO PID controller, it can be seen that in large power systems distributed MPC is an attractive option for advanced control.

One issue that may arise in power systems is the level of communication that may be needed before the distributed MPC converges to its final solution. In the examples given in this chapter it can be seen that up to 4 iterations were needed before convergence was achieved. Given the small time scales involved in power systems it is desirable to minimise the number of iterations needed for convergence. In the following chapter a simple method is used via optimisation of the distributed MPC weights in order to minimise the number of iterations needed for convergence.

References

B. Birge. PSOt - a Particle Swarm Optimization toolbox for use with Matlab. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 182–186, Indianapolis, Indiana, USA, April 2003.

M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6 (1):58–73, Feb 2002. ISSN 1089-778X. doi: 10.1109/4235.985692.

M. Dorigo and C. Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(23):243 – 278, 2005.

R. Erikkson. *Security-centered Coordinated Control in AC/DC Transmission Systems*. Licentiate thesis, Royal Institute of Technology, School of Electrical Engineering, Electric Power Systems, Stockholm, Sweden, 2008.

Z. W. Gaing. A Particle Swarm Optimization approach for optimum design of PID controller in AVR system. *IEEE Transactions on Energy Conversion*, 19(2):384 – 391, June 2004.

F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Massachussetts, USA, 1997.

R.L. Haupt and S.E. Haupt. *Practical Genetic Algorithms*. John Wiley & Sons, Inc., 2 edition, 2004.

A.H. Jones and P.B. De Moura Oliveira. Genetic auto-tuning of PID controllers. In *Proceedings of the first International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 141–145, Galesia, Spain, Sept. 1995.

J. Kennedy and R. C. Eberhart. Particle Swarm Optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, University of Western Australia, Perth, Australia, Nov.-Dec. 1995.

P. Kundur. *Power System Stability and Control*. Mc-Graw Hill, New York, 1994.

D.P. Kwok and F. Sheng. Genetic algorithm and simulated annealing for optimal robot arm PID control. In *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 707–713, Orlando, FL , USA, June 1994.

M.A. Pai, K.R. Padiyar, and C. Radhakrishna. Transient stability analysis of multi-machine AC/DC power systems via energy-function method. *IEEE Transactions on Power Apparatus and Systems*, 100(12):5027–5035, Dec. 1981.

P.W. Sauer and M.A. Pai. *Power System Dynamics and Stability:*. Prentice Hall, 1998.

I.C. Trelea. The Particle Swarm Optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85:317 – 325, 2003.

P.J.M. van Laarhoven. *Simulated Annealing: Theory and Applications*. Mathematics and it's applications. Kluwer Academic Publishers, 1987.

T. Weise. *Global Optimization Algorithms-Theory and Applications*. http://www.it-weise.de/, 2009.

Chapter 6

Weight Optimisation for Distributed MPC using PSO

6.1 Introduction

Weights are used in MPC to determine the relative importance of the various goals contained in the MPC cost function during optimization. By tuning the weights appropriately, MPC system performance can be improved, given a specific performance criterion. Typically, there is not an intuitive relationship between weights and the desired system performance criteria. Thus, a number of techniques in the literature have been proposed for automatically tuning centralised MPC weights (Di Cairano and Bemporad, 2010; Lee and Yu, 1994; Rowe and Maciejowski, 2000). PSO has been used in (Suzuki et al., 2007) for this purpose, with promising results. As was the case in the optimisation of PID gains, PSO is attractive for tuning, for a number of reasons, including its capability to optimise on a wide range of surfaces which may be convex, non-convex, discontinuous or multimodal (Trelea, 2003; Liang et al., 2006). This means that it is flexible in terms of what criterion can be used to determine the fitness of given weights. Also, no special knowledge is needed about the MPC algorithm being used and so it is user friendly for industrial practitioners. PSO is preferrable in comparison with GA for weight optimisation due to the level of correlation between weights when optimising in an MPC setting, as was the

case with the PID gains in the previous chapter.

While typically in centralised MPC one is concerned with tuning weights so as to fulfill a setpoint tracking criterion, in iterative distributed MPC algorithms, such as those in (Negenborn et al., 2008; Venkat, 2006; Sanchez et al., 2011), it is typically desirable to choose weights so as to achieve a good trade off between setpoint tracking and the communication overhead. In these algorithms, agents must communicate each iteration with adjacent agents. The level of communication needed depends on the number of iterations required for agents to form a consensus on inputs and interconnecting variables. In situations where plants have slow dynamics, and therefore longer sample times (e.g., chemical plants), high levels of communication may not be important, as agents have sufficient time to communicate with each other before applying control inputs to the plant. However, in large networks with fast dynamics, such as power networks, it is necessary that the number of such iterations is small, as the short sample times constrain the time allowed for communication.

In this chapter, a novel PSO based weight optimisation algorithm is proposed for the serial iterative distributed MPC technique given in Section 4.2.2. First, the effects on both disturbance rejection and the communication overhead are illustrated when the PSO weight optimisation fitness is based only on disturbance rejection. Then, an iteration deterrent, that assigns a penalty proportional to the maximum number of iterations used in a simulation is added to the PSO fitness function. The results obtained when using the deterrent are compared to those achieved in the original optimisation that is based solely on the disturbance rejection performance (while the criteria in this chapter are based on disturbance rejection, the same technique could be applied to optimise for setpoint tracking in other scenarios).

This weight tuning technique is evaluated on two different multi-agent Load Frequency Control (LFC) situations. The first system is a discrete-time 20 area LFC problem, which has a large number of tunable parameters, thus making it difficult to tune. The second system is the smaller scale, continuous-time multiple High Voltage Direct Current (HVDC) link problem that was used in the previous chapter. While there are less tunable parameters in this problem, a large number of iterations may be needed to achieve convergence of the distributed MPC problem at each sample step. It is difficult in this problem to find a set of weights to yield a desirable trade off between disturbance rejection and the

communication overhead for the large disturbance that is considered.

The way in which PSO is used to tune the distributed MPC weights will first be discussed and then the weight optimisation will be carried out on the two systems. The effect of weight optimisation on the closed loop performance and the communication overhead will then be seen when weights are optimised based solely on disturbance rejection, and then again when the iteration deterrent is used.

6.2 PSO Weight Optimisation for Distributed MPC

PSO has previously been used to optimise the centralised MPC weights in (Suzuki et al., 2007), resulting in improved performance according to the desired criterion. Typically, when tuning controllers, practitioners are concerned with improving aspects of the set-point tracking or disturbance rejection performance of a system. However, in iterative distributed MPC algorithms practitioners are concerned with both closed-loop performance and the level of communication used to achieve this control. This would be particularly of concern in power system networks where short control sample times are needed for the control of the system, thus limiting the amount of communication allowed between agents. Therefore, a tuning algorithm for iterative distributed MPC that considers both the disturbance rejection performance of the system and the number of iterations needed for the system to converge is desirable for power systems and other systems with fast dynamics. A novel PSO based weight optimisation algorithm for agents in a distributed MPC system is developed in this section. In addition a criterion for suppressing the number of iterations needed for distributed MPC is proposed.

The vector $\mathbf{\Gamma}=[\gamma_1 \dots \gamma_{n_\gamma}]^{\mathrm{T}}$ contains n_γ tunable weights, consisting of the distributed MPC disturbance rejection related weights associated with each agent's local problem and the c and ϵ weights that are associated with achieving consensus between agents. For each of the P particles in the PSO, of dimension $d=n_{\boldsymbol{\gamma}}$, a distributed MPC simulation is carried out. In this work, this simulation is chosen as a worst case scenario that excites each of the subsystems controlled by the distributed MPC agents sufficiently, to prepare the system for the worst contingencies that might arise. The j^{th} agent's local fitness function, $f_j^{\mathrm{local}}(q,i)$ is evaluated after a simulation has been run by particle q at iteration i of the

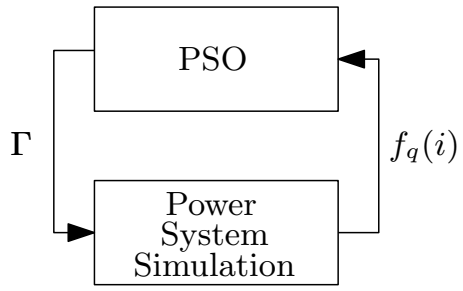


Figure 6.1: The PSO optimisation of the distributed MPC weights at iteration i of the PSO optimisation.

PSO optimisation. The sum of the local fitnesses of all n agents, $\sum_{j=1}^{n} f_j^{\text{local}}(q,i)$, then forms the overall disturbance rejection fitness for the q^{th} particle's simulation run at PSO iteration i. When a system is optimised for disturbance rejection only (henceforth referred to as the DR only case), the fitness of particle q in the swarm is given by

$$f_q(i) = \sum_{j=0}^{n} f_j^{\text{local}}(q,i), \tag{6.1}$$

where $f_q(i)$ is the fitness of particle q at iteration i of the PSO algorithm.

When it is desired to reduce the communication overhead, by suppressing the number of iterations used in a given simulation, an iteration suppressing cost $\rho(q,i)$ is used, where

$$\rho(q,i) = \max(\boldsymbol{\mu}(q,i)). \tag{6.2}$$

Here, $\boldsymbol{\mu}(q,i)$ is a vector of the number of distributed MPC iterations used at each sample step during the q^{th} particle's simulation run at PSO iteration i. In cases where it is desired to use an iteration deterrent (henceforth, referred to as the DRID case) the fitness of particle q at iteration i becomes:

$$f_q(i) = \upsilon_\rho \rho(q,i) + \sum_{j=0}^{n} f_j^{\text{local}}(q,i), \tag{6.3}$$

where υ_ρ is a positive constant used to determine the relative importance of the iteration deterrent cost to the disturbance rejection cost during optimisation.

The PSO weight optimization algorithm is as follows:

1. A random population of P particles is initialised in d dimensions, with $\boldsymbol{x}_{\min} \leq \boldsymbol{x}_q$ and $\boldsymbol{x}_q \leq \boldsymbol{x}_{\max}$, where $\boldsymbol{x}_{\min} \geq \mathbf{0}$ and $\boldsymbol{x}_{\max} \geq \mathbf{0}$ are the upper and lower bounds on particle

q's position $\boldsymbol{x}_q$. If good initial estimates are known in advance, some particles can be initialized with these values instead.

2. For each particle q, $f_j^{\text{local}}(q,i)$ for $j=1,\ldots,n$ and then $f_q(i)$ are evaluated.

3. Based on these fitnesses the PSO algorithm updates each particle's personal best position $\boldsymbol{x}_q^{\text{b}}(i)$ and fitnesses corresponding to these positions, $f_q^{\text{b}}(i)$, for q=1 ... P, and $\boldsymbol{x}^{\text{g}}(i)$, the global best position, and its associated fitness $f^{\text{g}}(i)$ and then calculates the next positions in the fitness function based on the velocity and position update equations,

$$\begin{aligned}\boldsymbol{v}_q(i+1) = \omega\boldsymbol{v}_q(i) + c_1\boldsymbol{r}_1(i) \circ \left(\boldsymbol{p}_q^{\text{b}}(i) - \boldsymbol{x}_q(i)\right) \\ + c_2\boldsymbol{r}_2(i) \circ (\boldsymbol{p}^{\text{g}}(i) - \boldsymbol{x}_q(i))\end{aligned} \tag{6.4}$$

$$\boldsymbol{x}_q(i+1) = \boldsymbol{x}_q(i) + \boldsymbol{v}_{q_{\text{app}}}(i+1). \tag{6.5}$$

Then with these P particles, the algorithm repeats from step (2).

4. The algorithm terminates when a termination criterion is satisfied; in this work this happens when $f^{\text{g}}(i)$ has not changed by more than a small specified tolerance for a given number of PSO iterations.

6.3 Simulation Experiments

In this section, PSO weight optimisation is applied to the distributed MPC control of two complex, highly interconnected power systems performing Load Frequency Control (LFC). First a discrete-time power network consisting of 20 subsystems is considered, and in the second experiment the continuous-time multiple link HVDC system, that was described in the previous chapter, consisting of 4 highly interconnected subsystems, is considered. In both cases the weights are first optimised based only on the disturbance rejection performance, and then optimised again incorporating the iteration deterrent, as in (6.3). The disturbance rejection criterion used here for the j^{th} agent in the simulation, run by particle q in the swarm, at PSO iteration i, is the Integral of the Square of Time by the Squared Error (ISTSE) (Gambier, 2007). This cost places greater emphasis on long term errors over short term errors that occur immediately after disturbances and is given by,

$$f_j^{\text{local}}(q,i) = \sum_{k=0}^{\infty} k^2 e_j^2(k), \tag{6.6}$$

Here, $e_j(k)$ is the frequency error (as compared to 1pu) in subsystem j at sample time k, and q and i are the PSO particle and iteration, respectively. As the tuning is based on simulation runs, it is not practical to run simulations for an infinite number of samples and so simulations are run for a finite time t_f that is long enough to adequately capture the systems dynamics.

The PSO routines are carried out using the PSO Toolbox for Matlab (Birge, 2003) using the parameters from (Trelea, 2003) due to their desirable convergence properties. These are $\omega = 0.6$, c_1=c_2=1.7. Other parameters used in the PSO toolbox are given in Table 6.1.

Parameter	Description	
p	number of particles	35
mvden	max. velocity divisor	2
epoch	maximum number of iterations	300

Table 6.1: Parameters used in the PSO toolbox for distributed MPC weight optimisation.

6.3.1 System 1: 20 Area Discrete-Time LFC Problem

The 20 area discrete-time LFC problem is shown in Fig. 6.2. The local control input is $u_a(k) = P_a^{\text{gen}}(k)$, the local disturbance is $d_a(k) = P_a^{\text{dist}}(k)$, and the local state is $\boldsymbol{x}_a(k) = [\delta_a(k), f_a(k)]^{\text{T}}$. The external inputs from other subnetworks are $\boldsymbol{v}_a(k) = [\delta_{\mathcal{N}_a^{\text{in}}\{1\}}(k), \ldots, \Delta\delta_{\mathcal{N}_a^{\text{in}}\{m_a\}}(k)]^{\text{T}}$, where m_a is the number of subnetworks connected to subnetwork a, and $\mathcal{N}_a^{\text{in}}\{i\}$ is the i^{th} agent connected to agent a. The discrete-time state space model for each subnetwork, with a step size τ=0.2s, are given by:

$$\begin{aligned}
\boldsymbol{A}_a &= \begin{bmatrix} 1 & \tau 2\pi \\ \sum\limits_{j\in\mathcal{N}_a} \tau \dfrac{-K_{\text{P}_a} K_{\text{S}_{aj}}}{2\pi T_{\text{P}_a}} & 1 - \frac{\tau}{T_{\text{P}_a}} \end{bmatrix}, \boldsymbol{B}_a = \begin{bmatrix} 0 \\ \tau \frac{K_{\text{P}_a}}{T_{\text{P}_a}} \end{bmatrix}, \\
\boldsymbol{D}_a &= \begin{bmatrix} 0 \\ -\tau \frac{K_{\text{P}_a}}{T_{\text{P}_a}} \end{bmatrix}, \boldsymbol{V}_a = \begin{bmatrix} 0 & \ldots & 0 \\ \tau \dfrac{K_{\text{P}_a} K_{\text{S}_{a\mathcal{N}_a^{\text{in}}\{1\}}}}{2\pi T_{\text{P}_a}} & \ldots & \tau \dfrac{K_{\text{P}_a} K_{\text{S}_{a\mathcal{N}_a^{\text{in}}\{m_a\}}}}{2\pi T_{\text{P}_a}} \end{bmatrix}.
\end{aligned} \tag{6.7}$$

This model is used to run the discrete-time simulation and the control is based on the incremental form of this model. The disturbances are not known by the control agents and so cannot be used when forming the predictive control. All subnetworks' parameters

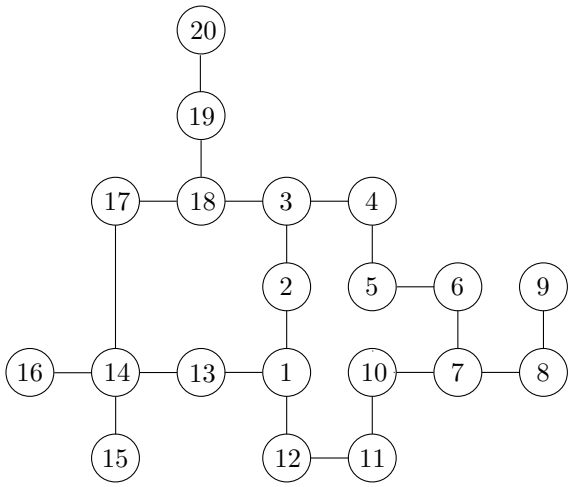


Figure 6.2: The 20 area discrete-time LFC problem.

are identical and are given as follows: $K_{\mathrm{p}_a} = 120$, $T_{\mathrm{P}_a} = 20$, $K_{\mathrm{S}_{aj}} = 0.5$. The controller developed in Section 4.4 was used again here. A prediction horizon of $N = 10$ was used to adequately take into account each subnetwork's dynamic response. Constraints on the inputs and states are as follows:

$$
\begin{aligned}
u_a(k+l) &\geq u_a^{\min} \\
u_a(k+l) &\leq u_a^{\max} \\
\boldsymbol{x}_a(k+l+1) &\geq \boldsymbol{x}_a^{min} \\
\boldsymbol{x}_a(k+l+1) &\leq \boldsymbol{x}_a^{max}
\end{aligned}
$$

for $l = 0, ..., N-1$, and $u_a^{\min}$=-0.3, $u_a^{\max}$=0.3, $\boldsymbol{x}_a^{\min} = [-10, -10]^{\mathrm{T}}$, $\boldsymbol{x}_a^{\max} = [10, 10]^{\mathrm{T}}$.

PSO Optimisation of the Distributed MPC Weights

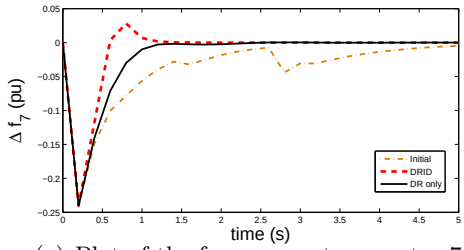
PSO is now used to optimise the weights and parameters of the distributed MPC system for the 20 area LFC system. Given the large number of agents in the system, it is non-trivial to find a combination of weights that give both good disturbance rejection performance and a low communications overhead. The weights determine the relative importance of the goals of the distributed MPC system; c is set equal to 1 and the other weights are then optimised using PSO. The vector of optimised weights here is $\boldsymbol{\Gamma}$=[Q_1... Q_{20} ϵ]$^{\mathrm{T}}$.

The R_a weights were not optimised and were given a value of 10^{-3}. However, these could be optimised if the practitioner so desired. The constraints for the variables in the PSO optimisation are as follows: $0.1 \leq Q_a \leq 100$, for agents $a=1 \ldots 20$, and $10^{-4} \leq \epsilon \leq 1$. For the PSO optimisations involving the iteration deterrent, $\upsilon=2.5$.

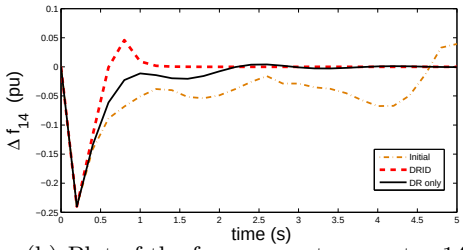
To save on the overall PSO simulation time, an upper limit of 50 distributed MPC iterations is allowed in each simulation run of the power system. If this is exceeded at any stage, a fitness of 1000 is allocated to the particle at that position and the simulation of the next particle begins. While this upper limit on distributed MPC iterations could be reduced, it allows the information from a wider range of particles to be used in the PSO optimisation. The PSO particle fitness is based on a network simulation run lasting $k_f=25$ discrete time steps, with sample time $\tau = 0.2$s, i.e., a total simulation time of 5s. This simulation involves disturbances of equal magnitude of 0.2 pu being applied at $t=0$s to all subsystems except subsystem 17, where a larger disturbance of 0.23 pu is applied. These disturbances are the largest that can be expected to occur in each area, and as such represents a worst case disturbance scenario for this system. Simulations were run on a computer with an Intel® Core™ 2 6400 operating at 2.13 GHz and with 3 GB of RAM in Matlab 7.6.0 (2008a). All distributed MPC optimisations are carried out using quadprog. The PSO terminates when f^{g} does not improve by more than 2.5×10^{-4} for 7 consecutive iterations.

Finding a set a weights to control this system for this scenario is non-trivial. The best performance that the author could achieve before optimisation, by manually tuning parameters, was with $[\ Q_1 \ldots Q_{20}\ c\ \epsilon\ ]^{\mathrm{T}} = [\ 10 \ldots 10\ 1\ 10^{-2}\ ]^{\mathrm{T}}$. Fig. 6.3 shows the results of the experiment. Unacceptable disturbance rejection is achieved, with area 14 becoming unstable towards the end of the simulation as a result of the disturbance, as can be seen in Fig. 6.3(b). The ISTSE for this performance is 131, and a large number of distributed MPC iterations were also required, incurring a significant associated communication cost.

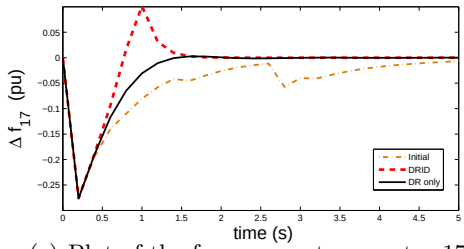
The frequencies in areas 7, 14, and 17 (used as sample illustrations of the effect of the weight optimisations) can be seen in Figs. 6.3(a), 6.3(b), and 6.3(c), respectively. These results include the initial manual tuning, the PSO tuning based only on disturbance rejection (DR only), and the PSO tuning based on disturbance rejection with an iteration deterrent (DRID). The number of distributed MPC iterations needed over time, for each of the aforementioned tunings, is given in Fig. 6.3(d), and f^{g} is plotted for each of the



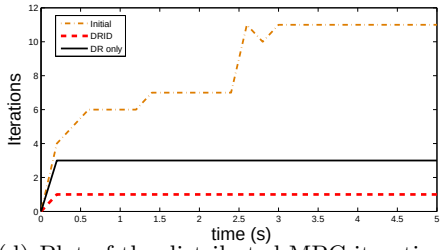
(a) Plot of the frequency at generator 7.



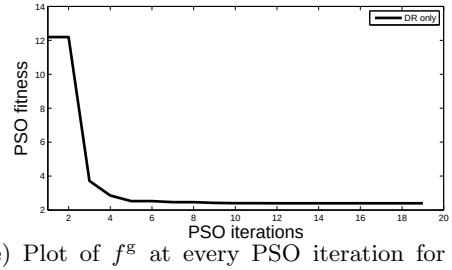
(b) Plot of the frequency at generator 14.



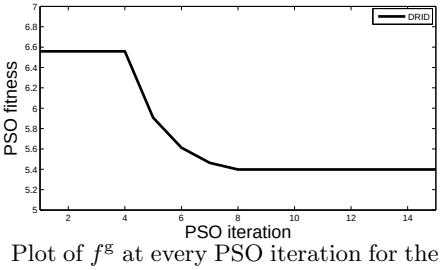
(c) Plot of the frequency at generator 17.



(d) Plot of the distributed MPC iterations.



(e) Plot of f^{g} at every PSO iteration for the disturbance rejection only PSO optimisation.



(f) Plot of f^{g} at every PSO iteration for the disturbance rejection with iteration deterrent PSO optimisation.

Figure 6.3: Plots of pu frequency and distributed MPC iterations over time, and PSO iterations for the disturbance rejection scenario applied to the 20 area discrete-time LFC network.

PSO iterations in the DR only and DRID cases in Figs. 6.3(e) and 6.3(f).

The final optimised weights for the DR only case are as follows: $[\ Q_1 ... Q_{20}\ ]^{\mathrm{T}} = [\ 1.78\ 3.11\ 94.22\ 19.98\ 63.92\ 71.42\ 0.10\ 0.10\ 83.33\ 20.93\ 97.38\ 87.52\ 0.10\ 0.10\ 42.16\ 40.24\ 95.16\ 0.10\ 19.86\ 99.55\ ]^{\mathrm{T}}$, ϵ=0.25, and the final ISTE cost of f^{g}=2.3975. The maximum number of distributed MPC iterations needed to achieve this is 3 as can be seen in Fig. 6.3(e). The final optimised weights for the DRID case are as follows: $[\ Q_1 ... Q_{20}\ ] = [\ 2.29\ 24.56\ 42.56\ 48.75\ 20.51\ 100.00\ 54.45\ 84.09\ 65.66\ 100.00\ 72.01\ 0.10\ 17.07\ 39.89\ 74.20\ 58.13\ 5.85\ 73.54\ 0.10\ 15.19\ ]^{\mathrm{T}}$, ϵ=0.37, and the final f^{g}=5.3975, consisting of an iteration cost of 2.5 and an ISTSE of 2.8975. Note, however that the maximum number of iterations needed to

achieve this control is only 1.

Looking at both PSO optimisations it can be seen that weight optimisation significantly improves not only the disturbance rejection cost of the system, but also the number of iterations needed to converge to the final solution, in both tuning cases. Comparing the ISTSEs of each of the optimisations it can be seen that the DR only case trades off an increased number of iterations for a better disturbance rejection performance, whereas the DRID case trades off a slightly worse disturbance rejection performance for a decrease in the number of iterations needed for the distributed MPC to converge. However, the disturbance rejection performance in the DRID case is still satisfactory, and a significant improvement on the disturbance rejection performance was achieved in comparison to the original tuning.

The overall PSO optimisation time for the DR only case was 5 hours and the DRID case was 5.58 hours (measured using cputime in Matlab which measures the total time in each cpu core spent over the whole simulation. The actual time taken is usually roughly equal to the cputime divided by the number of cores on the computer, i.e., on a dual core computer the real time would be roughly half the cputime). It can be seen in Figs. 6.3(e) and 6.3(f) that in both cases after 8 iterations, PSO does not significantly improve, and is quite near the final optimal value for the weights.

6.3.2 System 2: Continuous-Time Multiple HVDC Link System

The system used in this section was the multiple link HVDC system that was described in the previous chapter. Due to the high level of interconnectivity between individual agents problems this is quite a challenging distributed MPC problem, when tuning for severe fault scenarios. Finding a good combination of weights that balance both desirable closed-loop performance and a low communication overhead is non-trivial and so this problem is also a suitable testbed on which to evaluate weight tuning algorithms. The continuous-time nonlinear model of this system consisted of four nonlinear differential equations of the form given here for the a^{th} generator:

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta_{\mathrm{r}_a}(t) = \omega_0 \Delta\omega_{\mathrm{r}_a}(t) \tag{6.8}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\omega_{\mathrm{r}_a} = \frac{1}{2H_a}\Big(P_{\mathrm{m}_a} - G_{a,a}E_{\mathrm{q}_a}^{'2} - \sum_{\substack{l=1\\l\neq a}}^{n} E_{\mathrm{q}_a}^{'}E_{\mathrm{q}_l}^{'}(G_{a,l}\cos(\delta_{\mathrm{r}_a}-\delta_{\mathrm{r}_l}) + B_{a,l}\sin(\delta_{\mathrm{r}_a}-\delta_{\mathrm{r}_l})) + g_{a,1}P_1^{\mathrm{DC}} + \ldots + g_{a,m}P_m^{\mathrm{DC}} - D_a\Delta\omega_{\mathrm{r}_a}\Big), \tag{6.9}$$

where $\delta_{\mathrm{r}_a}(t)$ is the rotor angle (rad), H_a is the inertial constant (s), $\omega_{\mathrm{r}_a}(t)$ is the rotor speed (pu), $\Delta\omega_{\mathrm{r}_a}(t)$= $\omega_{\mathrm{r}_a}(t)-1$ is the rotor speed deviation (pu), ω_0 is the base rotor speed (rad/s), $P_{\mathrm{m}_a}(t)$ is the mechanical power (pu), and D_a is the damping factor (pu), E_{q_a} is the magnitude of the internal voltage of generator a, $G_{a,l}$ and $B_{a,l}$ are the coefficients of the contribution of an internal voltage E_{q_a} to generator current l in the system, and $g_{a,j}$ is the coefficient of the contribution of the power injections from HVDC link j at bus a. The power system parameters, that were given previously in Table 5.1, were used in this simulation again. The states of agent a are taken as $\boldsymbol{x}_a$=$[\delta_{\mathrm{r}_a}\ \omega_{\mathrm{r}_a}]^{\mathrm{T}}$, the inputs $\boldsymbol{u}_a$=$[P_1^{\mathrm{DC}}\ P_2^{\mathrm{DC}}]^{\mathrm{T}}$, and the interconnecting input $\boldsymbol{v}_a$=δ_{r_l}. The full system model is discretised and the distributed MPC controller described in Section 5.4 was used to control the system. The HVDC link ranges are $-2 \leq P_i^{\mathrm{DC}}(k) \leq 2$ pu, for $i \in \{1,2\}$ and the frequency range at all buses is $0.97 \leq \omega_{r_a}(k) \leq 1.03$ pu, for $a = 1\ldots4$. These constraints are applied over the full prediction horizon.

PSO Optimisation of the Distributed MPC Weights

PSO was used to optimise the weights and parameters of the distributed MPC system for the multiple HVDC link system. Due to the high level of coupling between the subsystems it can be difficult to tune the system weights to achieve good disturbance rejection in a small number of iterations. As previously stated, the weights determine the relative importance of the goals of the distributed MPC system. Therefore one weight is set equal to 1 and the rest of the weights are then optimised relative to this weight using PSO. The weight of agent 1, Q_1 is set equal to 1 and the vector of PSO weights is then $\boldsymbol{\Gamma}$=$[Q_2\ Q_3\ Q_4\ c\ \epsilon]^{\mathrm{T}}$. The R_a weights are not optimised and are simply set to a small positive constant, where R_a=diag(10^{-3},10^{-3}). This helps ensure that the optimisation problem is of full rank. However, as in the previous example, these weights could be optimised, if desired by the practitioner. The constraints for the variables in the PSO optimisation were as follows: $0.1{\leq}Q_a{\leq}100$, for $a = 2\ldots4$, $0.01{\leq}c{\leq}5$, and $10^{-4}{\leq}\epsilon{\leq}1$. The PSO terminates when f^{g} does not improve by more than 0.1 for 7 consecutive iterations.

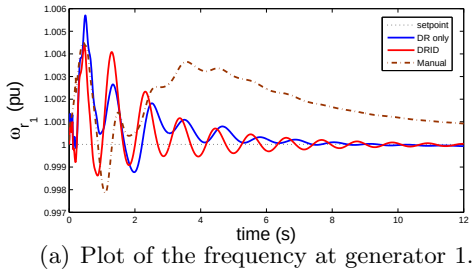
An upper limit of 120 distributed MPC iterations is allowed for each control cycle to save on simulation time in each simulation of the multiple HVDC link system that is run by the PSO particles. If this is exceeded at any stage a fitness of 1000 is allocated to the particle at that position and the simulation of the next particle is initiated. Again this could be set lower but it allows the information from a wider range of particles to be useful in the PSO optimisation.

The tuning scenario used involves three-phase to ground faults being simultaneously applied to lines 1 and 3 in the system for a duration of 200ms and then returning the system to its non-fault state. Tuning for this scenario is quite difficult and in fact a number of initial tuning attempts did not stabilise the system. When manually tuning the distributed MPC, the best performance the authors could attain was using $[Q_1\ Q_2\ Q_3\ Q_4\ c\ \epsilon]^{\mathrm{T}}$=[10 30 10 10 1 10^{-3}]$^{\mathrm{T}}$. While this guess is stabilising, there is still an offset towards the end of the simulation and up to 40 distributed MPC iterations are needed for one of the control cycles to converge. It was decided not to initialise the PSO with this guess to see how long it would take to converge on the final solution with random initial guesses. For the simulations involving the iteration deterrent, υ=100.

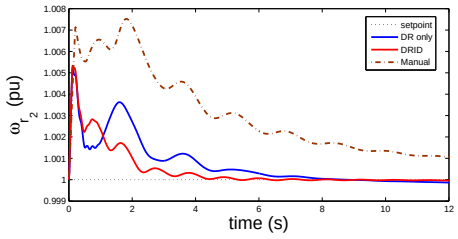
Results

Simulations were run on a 2211.412 MHz Quad-Core AMD Opteron$^{\mathrm{TM}}$ Processor 2354 with a 512 KB cache size in Matlab version 7.11.0.584 (R2010b). Simulink is used to simulate the nonlinear, continuous-time power system simulations, using the Dormand-Prince (ode45 in Matlab) continuous-time algorithm, with a maximum step size of 2ms and a relative tolerance of 0.001. Linearisations of the nonlinear equations (6.8) and (6.9), are used to derive the discrete-time state space models that are used in the distributed MPC controllers. These inputs were calculated and applied at fixed time steps of 10ms using Matlab and the calculated inputs were passed to the continuous-time Simulink simulations. All MPC optimisations were performed using TOMLAB v7.4.

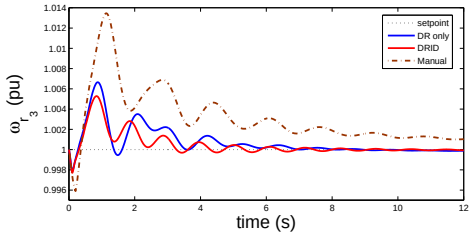
The final output of the disturbance rejection only PSO weight optimisation (DR only) set $[Q_1\ Q_2\ Q_3\ Q_4\ c\ \epsilon]^{\mathrm{T}}$=[1 22.8 100 100 2.9 0.3]$^{\mathrm{T}}$, with the final $f^{\mathrm{g}}(i)$=23.13. The final result in the PSO optimisation based on disturbance rejection with the iteration deterrent



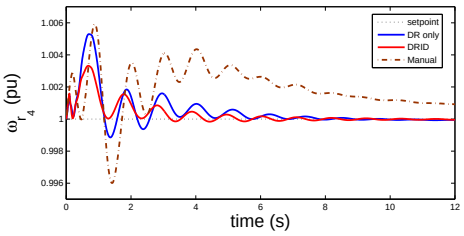
(a) Plot of the frequency at generator 1.



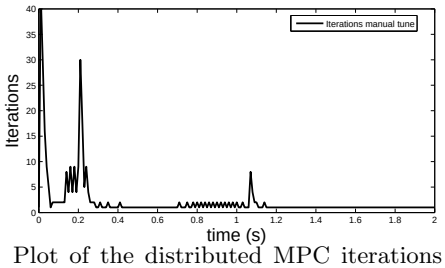
(b) Plot of the frequency at generator 2.



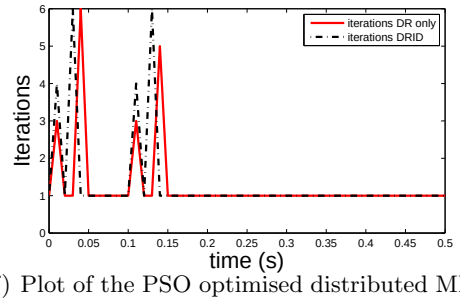
(c) Plot of the frequency at generator 3.



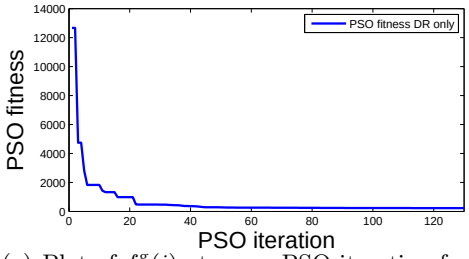
(d) Plot of the frequency at generator 4.



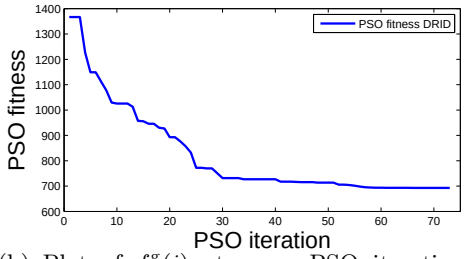
(e) Plot of the distributed MPC iterations for the original tuning (iterations stay at 1 after 2 seconds).



(f) Plot of the PSO optimised distributed MPC iterations (iterations stay at 1 after 0.5 seconds).



(g) Plot of $f^{\mathrm{g}}(i)$ at every PSO iteration for the disturbance rejection only PSO optimisation.



(h) Plot of $f^{\mathrm{g}}(i)$ at every PSO iteration for the disturbance rejection with iteration deterrent PSO optimisation.

Figure 6.4: Plots of pu frequency and iterations over time for the 200ms line fault applied to lines 1 and 3 simulataneously.

(DRID) gave $[Q_1\ Q_2\ Q_3\ Q_4\ c\ \epsilon]$=[1 59.6 100 90.1 1.06 0.13]$^{\mathrm{T}}$, with the final f^{g}=693. The f^{g} for the DRID case consisted of a disturbance rejection cost of 93 and an iteration deterrent cost of 600. The fitness at each iteration of the PSO algorithm can be seen in

Figs. 6.4(g) and 6.4(h) for the DR only and DRID cases, respectively. The other plots in Fig. 6.4 show the frequencies in each area plotted against time for the initial set of weights, the DR only weights, and the DRID weights. The plots of the distributed MPC iterations needed over the course of the simulation to achieve the control for each of the aforementioned weight scenarios are also shown.

It can be seen that in both cases the maximum number of distributed MPC iterations needed for convergence during the simulation run is significantly reduced in comparison with the original case, and that the disturbance rejection significantly improves. This illustrates that weight optimisation can simultaneously improve both the disturbance rejection and communication overhead, at least in certain situations. While the DR only case results in a smaller overall number of iterations than the DRID case, both the DR only and DRID cases converge on the same maximum number of iterations for the simulation run, but the DR only case ends up with a smaller overall communication overhead. In fact the DR only case also has the smaller disturbance rejection cost of the two. This could be simply because the PSO simply did not come across this solution while searching in the DRID case. However, it must also be considered that the discontinuities in the cost function caused by the iteration deterrent prevented this better result being found in the DRID case. From a practical point of view, though, both results are considerably better in terms of both disturbance rejection and communication performance than with the original case.

It is noted that there is a significant amount of computational overhead associated with finding these weights. DRID took approximately 4 weeks and the duration for the DR only simulation took approximately 8.5 weeks. The length of time needed for the optimisation was due to the long run-time of the multiple HVDC link simulation. On average, simulation runs of these simulations took between 10 and 20 minutes each due to the fact that the sample time needed for simulation was quite small and the fact that a very long simulation time was needed due to the long settling times in the multiple HVDC link system. Also, a prediction horizon of 50 steps is needed in this system, and so each of the distributed MPC problems are quite large. Much time was also spent in the exploitation stage, of the PSO algorithm, finding the final solution. It can be seen that even after 20 iterations, in both cases, the PSO has almost converged to the final result.

6.3.3 Discussion of Overall Results

In both of the PSO weight optimisation experiments in this section it can be seen that the optimised weight values give simultaneously both improved disturbance rejection performance and a reduced number of distributed MPC iterations. Also it can be seen that the iteration deterrent has the potential to further minimise the maximum number of distributed MPC iterations needed in a simulation. Therefore, the use of the iteration deterrent can be useful in scenarios where it is desirable to minimise the number of iterations needed for convergence of the distributed MPC algorithm, while seeking to simultaneously improve disturbance rejection performance.

The PSO, in both power system experiments, manages to reach a near optimal result in the early stage of optimisation. However the exploitation stage of the algorithm takes a significant number of iterations and so can be wasteful in terms of the overall computational overhead. Another criterion, based on the performance being within satisfactory bounds, could be used in cases where the system simulations take quite a long time to run, in order to terminate the PSO optimisation at an earlier stage.

The duration of the PSO weight optimisations is significantly influenced by the time taken to run the individual power system simulations used to evaluate PSO particle fitnesses. While the PSO updates are calculated quite efficiently, the vast majority of the time taken to run the optimisations is based on the length of time taken for the simulation scenarios. It is for this reason that the multiple HVDC link weight optimisation experiment took significantly longer than the 20 area LFC weight case. It is worth carefully considering the length of time the system simulations run for during each fitness evaluation, how efficient simulation runs are, and what termination criteria should be used to terminate the PSO, in order to reduce the overall PSO weight optimisation time. For computationally intensive simulations, such as the multiple HVDC link system, overall times for PSO optimisation could also potentially be reduced by running simulations for each of the particles in parallel on separate computers.

6.4 Conclusions

A PSO-based weight optimisation algorithm is proposed here for distributed Model Predictive Control (MPC). Two criteria are used to evaluate PSO particle fitness. The first is based only on the disturbance rejection performance of the system. The second is based on the disturbance rejection performance of the system and the communication overhead of the system. The communication overhead is measured as the number of iterations needed for the distributed MPC to reach convergence.

As a general strategy for tuning distributed MPC systems, PSO is advantageous as it works effectively on a wide range of surfaces and so is quite flexible in terms of what fitness criteria can be used to tune the system. Also, practitioners do not need any in depth knowledge of the distributed MPC algorithm they are tuning in order to tune the weights using PSO, as described in this chapter. Using an iteration deterrent for the weight optimisation, it is possible to tune the control system to achieve a desirable trade off between the closed loop performance and the communication overhead needed to achieve this control.

Two power systems examples are used to evaluate this weight optimisation technique. In both cases the weight optimisation results in an improvement in both the disturbance rejection overhead and the communication overhead. However, it was possible to further reduce the communications overhead using the disturbance rejection with an iteration deterrent criterion.

References

B. Birge. PSOt - a Particle Swarm Optimization toolbox for use with Matlab. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 182–186, Indianapolis, Indiana, USA, April 2003.

S. Di Cairano and A. Bemporad. Model predictive control tuning by controller matching. *IEEE Transactions on Automatic Control*, 55(1):185 –190, jan. 2010.

A. Gambier. Parametric optimization for practical control systems design. In *16th IEEE International Conference on Control Applications*, pages 301–306, Singapore, Oct. 2007.

J.H. Lee and Z.H. Yu. Tuning of model predictive controllers for robust performance. *Computers & Chemical Engineering*, 18(1):15 – 37, 1994.

J.J. Liang, A.K. Qin, P.N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3):281 – 295, June 2006.

R. R. Negenborn, B. De Schutter, and J. Hellendoorn. Multi-agent model predictive control for transportation networks: Serial versus parallel schemes. *Engineering Applications of Artificial Intelligence*, 21(3):353–366, April 2008.

C. Rowe and J. Maciejowski. Tuning mpc using H infinity loop shaping. In *Proceedings of the American Control Conference*, pages 1332 –1336, Chicago, Illinois, June 2000.

G. Sanchez, L. Giovanini, M. Murillo, and A. Limache. *Advanced Model Predictive Control*, chapter Distributed Model Predictive Control Based on Dynamic Games, pages 1–26. InTech, July 2011.

R. Suzuki, F. Kawai, H. Ito, C. Nakazawa, Y. Fukuyama, and E. Aiyoshi. Automatic Tuning of Model Predictive Control Using Particle Swarm Optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 221–226, Honolulu, Hawaii, USA, April 2007.

I.C. Trelea. The Particle Swarm Optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85:317 – 325, 2003.

A. Venkat. *Distributed Model Predictive Control: Theory and Applications*. PhD thesis, University of Wisconsin-Madison, Wisconsin, 2006.

Chapter 7

Conclusions and Future Work

This thesis has looked at the optimal control of Smart Grids, focusing in particular on iterative distributed Model Predictive Control (MPC). A novel convergence and stability proof was derived for distributed MPC. The performance of distributed MPC was then compared to that of an optimised PID and a centralised MPC controller. The application in this case was a highly interconnected, non-linear, MIMO system based on part of the Nordic power grid. The final contribution was to develop a tuning algorithm for distributed MPC that considered both the closed-loop performance and communications overhead involved in attaining a desired system control performance. A summary of the contents of each chapter will now be given, outlining the contributions made in each chapter.

7.1 Thesis summary

In chapter 1, a brief history of control theory was given, showing how decentralised control structures were originally developed for the control of large scale systems, but that there were disadvantages inherent in decentralised control systems. Among these disadvantages was the fact that decentralised control was often not suitable for the control of highly interconnected systems and that a certain control system structure was often necessary in order for decentralised control techniques to be viable. It was then shown how networked

and distributed control systems grew in prominence and how they are the preferred control method for use in the Smart Grid.

Chapter 2 described how and why power networks are evolving from their traditional centralised form into the new distributed Smart Grid form. An overview of the power electronics devices which will enable a high level of controllability in future power networks was then given. Following this, issues surrounding the design of distributed control systems for large scale systems were discussed. In particular it was noted that the level of communication between control agents affects the level of control performance that can be achieved in a system. It was seen that while the control performance of a centralised MPC can be achieved with a Pareto equilibrium seeking algorithm, these algorithms required an excessive amount of inter-agent communication and so are not feasible for the control of large-scale systems such as power networks. Nash seeking control algorithms, on the other hand, only exchange information with directly connected agents. It was seen that while these did not always achieve the performance of Pareto-equilibrium seeking algorithms, they generally achieve significantly better performance than decentralised algorithms and that they provide adequate control performance when applied to power networks.

In chapter 3, a detailed description of Model Predictive Control was given which included a history of MPC, and demonstrated how the control problem is formulated in both the unconstrained and constrained cases. MPC was then applied to a 3 area Load Frequency Control (LFC) problem. Two scenarios were given; one in which a Static Synchronous Series Compensator (SSSC) was present in the power system and another where the SSSC was not present. It could be seen that in both cases MPC provided adequate control performance to stabilise the system within the allocated constraints; however when the SSSC was present there was a significant improvement in the control performance. This example demonstrated how power electronic devices used with advanced control techniques can provide a high level of control performance in power systems.

In chapter 4, the distributed MPC algorithm was introduced. A novel convergence and stability proof was then developed for the algorithm. The effectiveness of this stability proof was then tested on a 2 area discrete-time LFC which was controlled using the distributed MPC algorithm. The point at which the system was theoretically found to go unstable was compared to the point at which this system went unstable in simulation. The system was driven to instability by increasing the interconnection coefficient between

the two areas. It was found that the stability proof could be used to accurately determine the point at which the system went unstable. Also, suggestions were given as to how the c and ϵ parameters should be chosen in order to improve the accuracy with which the stability could be predicted.

The performance of a number of optimal controllers, when used for the control of a highly interconnected, nonlinear, MIMO system, were evaluated and compared in Chapter 5. The system was based on part of the Nordic power grid. The system consisted of 4 areas which were connected by 2 AC and 2 HVDC lines. The level of interconnection in this system is exacerbated due to the fact that the generation capacities in each generator in the system were kept constant. Therefore, when the system was subjected to temporary 3-phase to ground faults, it was the action in the HVDC lines alone that was used to restabilise the system generator frequencies. A centralised and distributed MPC, and a multi-loop PID control scheme, whose gains had been optimised using Particle Swarm Optimisation (PSO), were used to control the system. This work also demonstrated how the distributed MPC technique could naturally be used to deal with cases where agents have shared input variables. Of interest (in this example in particular) was the comparison of performance between the optimised PID control scheme and the distributed MPC controller. PID controllers are the most prominent controllers in power systems at the moment. Distributed MPC provides a scalable form of optimal control that can be used on large-scale systems and so it has the potential to replace PID controllers. Therefore, it was interesting to see if there were any major advantages in using distributed MPC over PID controllers. It was seen that distributed MPC did in fact provide sufficiently better control than the PID controllers for serious line faults. However, there was a large computational and communication overhead associated with the distributed MPC.

In chapter 6, a novel PSO based tuning algorithm was proposed for use with distributed MPC. This algorithm enabled the simultaneous minimisation of the communication overhead and optimisation of the closed-loop performance of the system. Each PSO particle ran a simulation, in every PSO iteration, that evaluated the achievable performance using a combination of distributed MPC weights. The reduction in communication was achieved by using an iteration deterrent that allocated a penalty based on the maximum number of iterations needed to achieve convergence during distributed MPC cycles over the course of a simulation run. The Integral of Time Squared by the the Square of Error (ISTSE) criterion was used to determine the closed-loop performance for each simulation run. The

weights were optimised for two different systems which were controlled using distributed MPC. One was a 20 area discrete time LFC simulation; the other was the multiple HVDC link system described in Chapter 5. Two PSO optimisations were performed for each system. In one, the optimisation was based solely on the closed loop performance only, and in the other optimisation the iteration deterrent was also used. With both systems it was found that optimising solely based on the closed loop performance resulted in a significant reduction in the number of distributed MPC iterations that were needed to achieve convergence. When the iteration deterrent was included, it was then possible to further reduce the communications overhead associated with these iterations.

7.2 Thesis Contributions

The primary contributions of this thesis are as follows:

- The application of optimal control systems to a variety of power systems was given. These examples reaffirmed the fact that optimal control techniques allow a high level of control performance to be achieved in Smart Grids. Furthermore, it was seen how power electronics devices could be used to further enhance the controllability of power systems. This was illustrated by adding an SSSC to a 3 area LFC problem and comparing the performance of this system to the system where the SSSC was not in place.
- A novel convergence and stability proof was derived for the distributed MPC algorithm which was originally developed in (Negenborn et al., 2008). This stability proof was then tested by increasing the interconnection coefficients in a 2 area LFC system and observing at what point the stability proof determined instability, which was then compared to the point at which the system was observed to go unstable in simulation. It was found that when small values of the ϵ variable were used that the point of instability was accurately found.
- A number of optimal control techniques were applied to a nonlinear multiple HVDC link system. The application of a PSO optimised multi-loop PID controller, and a centralised and distributed MPC controller to this system had not been conducted previously. Also, it was demonstrated using this system how distributed MPC could

naturally be used in cases where control agents shared inputs. It was found that the distributed MPC gave the best performance, centralised MPC gave the second best performance, and that the optimised multi-loop PID controller gave the third best performance. However, it was seen that the optimised PID controller violated the desired frequency bounds for the system for the given fault scenario, whereas these were satisfied with the MPC controllers. Decentralised MPC was also applied to the system but this was not capable of stabilising the system under the multi-agent deregulated control structure that was desired for system control. This system highlighted the improvement in performance that can be achieved by using distributed MPC rather than PID controllers in power networks.

- A PSO based tuning algorithm was developed for iterative distributed MPC which allows the closed-loop performance and communication overhead of the system to be tuned simultaneously. The PSO tuning was applied to two systems; a discrete time 20 area LFC system, and the multiple HVDC link system. The results found when optimising for the closed-loop performance alone, and the results found when optimising for both the closed-loop performance and the communication overhead were then compared. It was found that in both cases that optimising the weights according to the ISTSE criterion significantly improved both the closed loop performance and reduced the communications overhead. However, when the communication overhead was optimised along with the closed-loop performance it was possible to cause a further reduction in the level of communication used.

7.3 Future Work

There is still the potential for much research in the areas covered in this thesis. It would be useful to develop a proof of stability and convergence for the constrained version of the distributed MPC algorithm presented in this thesis. The PSO based tuning algorithm in this thesis could also be used to further reduce the communication overhead by minimising the prediction horizon used in the distributed MPC. The stability proof developed in Chapter 4 could also be used to determine if a set of weights will provide stabilising control when distributed MPC weights are being tuned. If weights were not stabilising it would mean that a simulation would not have to be run for this combination of weights during the PSO weight tuning procedure. This would aid the efficiency of the tuning

algorithm. Also, it would be useful to see how well other optimisation methods such as the Nelder-Mead Simplex perform when used for the optimisation of the distributed MPC weights.

The multiple link HVDC system was a challenging testbed for the distributed MPC algorithm in this thesis due to the high level of interconnection between agents. It would be interesting to compare the performance of several distributed MPC algorithms on this testbed and also to see how these algorithms perform when model uncertainties and delays are introduced into the system. It would also be useful to see how a hierarchical distributed MPC system which utilised feedback linearisation performed when applied to a nonlinear system, such as the multiple HVDC link system. The feedback linearisation could be carried out in an upper layer that had access to full network information and could then pass the linearised model down to lower layer controllers.

Before distributed MPC could be implemented in real power networks, the effects of the communication network on the control will need to be considered. Typically control algorithms assume a perfect communications system which may have some delays. However, in certain communication systems information may not be sent at a constant rate, and the communication medium is usually subject to packet losses. These can have serious impacts on closed loop behaviour and should be considered when developing distributed MPC for real systems. Also, a vital part of Smart Grids will be their ability to adapt to changes in the network. The author is not aware at this time of any significant body of work on model adaptive distributed MPC and so would suggest this as an important area for future research. Also, automated methods for partitioning the system into suitable subsystems for control would be necessary for the implementation of distributed MPC in Smart Grids.

There is huge potential for the application of distributed MPC in new areas. As discussed previously, distributed MPC has been applied to power networks, water networks, supply chains, and chemical plants to name a few. Given the recent economic turmoil that has engulfed the world, it would be interesting to see how distributed MPC techniques could be applied to economic systems, particularly macroeconomic systems. Inevitably the distributed MPC would need to be a robust or stochastic formulation given the large amount of uncertainty involved in economic systems. However, similar issues arise in economic systems as are found in deregulated power markets, in terms of the scalability of

the control algorithm and constraints on how much information can be used, and so Nash equilibrium seeking distributed MPC algorithms could be suitable for use in this context.

References

R. R. Negenborn, B. De Schutter, and J. Hellendoorn. Multi-agent model predictive control for transportation networks: Serial versus parallel schemes. *Engineering Applications of Artificial Intelligence*, 21(3):353–366, April 2008.

Appendix A

Deterministic constrained optimisation techniques

A.1 Constrained Optimisation and Optimality Conditions

Deterministic optimisation techniques allow for the efficient solution of a given optimisation problem based on rigorous mathematical foundations. Constrained optimisation problems can typically be stated as follows:

$$\boldsymbol{x} = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \begin{cases} \kappa_i(\boldsymbol{x}) \geq 0, & i \in \mathcal{I} \\ \kappa_j(\boldsymbol{x}) = 0, & j \in \mathcal{E}. \end{cases} \tag{A.1}$$

where $\boldsymbol{x} \in \Re^n$ is a vector of variables that are used to minimise an objective function $f(\boldsymbol{x})$, $\mathcal{I}$ and $\mathcal{E}$ are two finite sets of indices, and $\kappa_i(\boldsymbol{x})$, for $i \in \mathcal{I}$, and $\kappa_j(\boldsymbol{x})$, for $j \in \mathcal{E}$, are the inequality and equality constraints of the problem, respectively. Typically deterministic optimisation is most suited to problems where $f(\boldsymbol{x})$, and κ_k for $k \in \mathcal{E} \cup \mathcal{I}$ are all smooth, real valued functions on a subset of $\Re^n$. The Karush-Kuhn-Tucker (KKT) optimality conditions are used to establish optimality in constrained optimisation theory. Before stating these theorems a few relevant definitions will be made (material from this section is taken from (Nocedal and Wright, 2006), where detailed proofs of the stated theorems and algorithms can be found).

The active set $\mathcal{A}(\boldsymbol{x})$ at any feasible point $\boldsymbol{x}$, where a feasible point is a point in the interior of the solution space, consists of the equality constraint indices from $\mathcal{E}$ together with the indices of the inequality constraints $i \in \mathcal{I}$ for which $\kappa_i(\boldsymbol{x}) = 0$; that is,

$$\mathcal{A}(\boldsymbol{x}) = \mathcal{E} \cup \{i \in \mathcal{I} | \kappa_i(\boldsymbol{x}) = 0\}. \tag{A.2}$$

At any feasible point $\boldsymbol{x}$, the inequality constraint $\kappa_i(\boldsymbol{x}) \geq 0$ for $i \in \mathcal{I}$ is said to be active if $\kappa_i(\boldsymbol{x}) = 0$ and inactive if the strict inequality $\kappa_i(\boldsymbol{x}) > 0$ is satisfied.

Given a point $\boldsymbol{x}$ in the active set $\mathcal{A}(\boldsymbol{x})$, the linear independence constraint qualification (LICQ) holds if the set of active constraint gradients $\{\nabla_{\boldsymbol{x}} \kappa_i(\boldsymbol{x}),\ i \in \mathcal{A}(\boldsymbol{x})\}$ is linearly independent. In general if the LICQ holds, none of the active constraint gradients can be zero.

Theorem A.1.1. First-Order Necessary Conditions.
Suppose that $\boldsymbol{x}^*$ is a local solution of (A.1)*, such that the functions f, κ_i, and κ_j in* (A.1) *are continuously differentiable, and that the LICQ holds at $\boldsymbol{x}^*$. Then there is vector $\boldsymbol{\lambda}^*$, called a Lagrange multiplier, with components λ_k^*, for each $k \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions are satisfied at $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$*

$$\begin{aligned}
\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) &= 0, \\
\kappa_i(\boldsymbol{x}^*) &\geq 0, \quad \textit{for all } i \in \mathcal{I}, \\
\kappa_j(\boldsymbol{x}^*) &= 0, \quad \textit{for all } j \in \mathcal{E}, \\
\lambda_i^* &\geq 0, \quad \textit{for all } i \in \mathcal{I}, \\
\lambda_i^* \kappa_i(\boldsymbol{x}^*) &= 0, \quad \textit{for all } i \in \mathcal{I}, \\
\lambda_j^* \kappa_j(\boldsymbol{x}^*) &= 0, \quad \textit{for all } j \in \mathcal{E}.
\end{aligned} \tag{A.3}$$

where $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{k \in \mathcal{I} \cup \mathcal{E}} \lambda_k \kappa_k(\boldsymbol{x})$, is called the Lagrangian function.

Theorem A.1.1 establishes the relationship between the first derivatives of f and the active constraints at the solution $\boldsymbol{x}^*$. Second order derivatives contain further important information that can be used during optimisation to determine if travelling along a given direction will increase or decrease f. A number of definitions are made, again, before stating the second order optimality conditions.

The set of linearised feasible directions $\mathcal{F}(\boldsymbol{x})$, given a feasible point $\boldsymbol{x}$ and the active

constraint set $\mathcal{A}(x)$, is defined as:

$$\mathcal{F}(\boldsymbol{x}) = \left\{ \boldsymbol{d} | \begin{array}{ll} \boldsymbol{d}^{\mathrm{T}} \nabla \kappa_j(\boldsymbol{x}) = 0, & j \in \mathcal{E} \\ \boldsymbol{d}^{\mathrm{T}} \nabla \kappa_k(\boldsymbol{x}) \geq 0, & k \in \mathcal{A}(\boldsymbol{x}) \cap \mathcal{I}. \end{array} \right\}. \tag{A.4}$$

Given $\mathcal{F}(\boldsymbol{x}^*)$ and the associated Lagrange multiplier vector $\boldsymbol{\lambda}^*$ satisfying the KKT conditions of Theorem A.1.1, the critical cone is then defined as follows:

$$\mathcal{C}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = \{\boldsymbol{w} \in \ \mathcal{F}(\boldsymbol{x}) | \nabla \kappa_k(\boldsymbol{x}^*)^{\mathrm{T}} \boldsymbol{w} = 0, \text{ all } k \in \mathcal{A}(\boldsymbol{x}^*) \cap \mathcal{I} \text{ with } \lambda_k^* > 0\}. \tag{A.5}$$

The critical cone defines the set of feasible directions from $\boldsymbol{x}^*$ that still fulfil the active constraints $\kappa_i(x)$.

The second order necessary optimality condition states that the Hessian of the Lagrangian function has non-negative curvature when moving along directions in $\mathcal{C}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$.

Theorem A.1.2. Second-Order Necessary Condition.
Suppose that $\boldsymbol{x}^*$ is a local solution of (A.1)*, with corresponding Lagrange multipliers $\boldsymbol{\lambda}^*$, and that the LICQ condition is satisfied. Then*

$$\boldsymbol{w}^{\mathrm{T}} \nabla^2_{\boldsymbol{xx}} \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) \boldsymbol{w} \geq 0, \text{ for all } \boldsymbol{w} \in \mathcal{C}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*). \tag{A.6}$$

These theorems provide the necessary conditions for the optimality of the solution of constrained optimisation problems. The development of duality theory provides further insite into the nature of constrained optimisation problems and has been central to the development of methods for solving constrained optimisation problems.

A.2 Duality Theory

Duality theory shows how for a constrained optimisation problem, a corresponding "dual" problem can be constructed from the original problem. The following problem is given:

$$\min_{\boldsymbol{x} \in \Re^n} f(\boldsymbol{x}), \text{ subject to } \boldsymbol{\kappa}(\boldsymbol{x}) \geq 0, \tag{A.7}$$

where $\boldsymbol{\kappa}(\boldsymbol{x}) = [\kappa_1(\boldsymbol{x}), \ldots, \kappa_m(\boldsymbol{x})]^{\mathrm{T}}$ is a vector of m inequality constraints. The Lagrangian function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ with Lagrange multiplier $\boldsymbol{\lambda} \in \Re^m$ is given as:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{\kappa}(\boldsymbol{x}). \tag{A.8}$$

The dual objective function $q(\boldsymbol{\lambda}) : \Re^n \rightarrow \Re$ is given as follows:

$$q(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}). \tag{A.9}$$

Often this infinum is $-\infty$ for some values of λ and so the domain of $q(\boldsymbol{\lambda})$, $\mathcal{D}$, is defined as the set of values for which $q(\boldsymbol{\lambda})$ is finite, that is,

$$\mathcal{D} = \{\boldsymbol{\lambda} | q(\boldsymbol{\lambda}) > -\infty\} \tag{A.10}$$

If $f(\boldsymbol{x})$ and -$\kappa_i(\boldsymbol{x})$ are convex functions and $\boldsymbol{\lambda} > 0$, then the function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is also convex and so all local minimizers are global minimisers, which makes the computation of $q(\boldsymbol{\lambda})$ practical.

The dual problem to A.7 is then defined as follows:

$$\max_{\boldsymbol{\lambda} \in \Re^n} q(\boldsymbol{\lambda}), \text{ subject to } \lambda \geq 0. \tag{A.11}$$

Provided f and -κ_i are convex the dual problem can be used to help solve the original primal problem and also provides a lower bound on the solution of the primal problem. Duality theory is central to the development of optimisation methods for constrained problems.

A.3 Constrained Optimisation Methods

There are a number of methods that have been developed for solving constrained optimisation problems. Most of these utilise results from duality theory in their formulation. Methods such as the simplex method, interior point methods, and quadratic programming all directly solve constrained optimisation problems within the constrained optimisation space using duality theory and the results of Theorems A.1.1 and A.1.2. Other methods, however, instead of searching within the constrained feasible region, convert the constrained optimisation problem into an unconstrained optimisation problem and use a number of iterations in order to converge on the final solution. These methods work by adding a penalty cost proportional to the constraints to $f(\boldsymbol{x})$ and solving the problem over a number of iterations to converge on $\boldsymbol{x}^*$.

Quadratic penalty methods convert (A.1) into a series of unconstrained minimisation

problems:

$$\begin{aligned}\boldsymbol{x}(k) &= \arg\min_{\boldsymbol{x}} \mathcal{Q}(\boldsymbol{x}(k), c(k)) \\ &= \arg\min_{\boldsymbol{x}(k)} f(\boldsymbol{x}(k)) + \frac{c(k)}{2}\sum_{i\in\mathcal{I}}\left([\kappa_i^2(\boldsymbol{x}(k))]^-\right)^2 + \frac{c(k)}{2}\sum_{j\in\mathcal{E}}\kappa_j^2(\boldsymbol{x}(k)),\end{aligned} \tag{A.12}$$

where y^- denotes $\max(-y, 0)$, and $\boldsymbol{x}(k)$ is the initial guess for $\boldsymbol{x}(k+1)$ at iteration k of the algorithm. When there are no inequality constraints the unconstrained problem is smooth and so can be easily solved. The addition of inequality constraints results in discontinuities and so $\mathcal{Q}(\boldsymbol{x}(k), c(k))$ is no longer twice continuously differentiable in this case. The algorithm works by starting with a small value of c and finding $\boldsymbol{x}^*$. This solution informs the starting point for the next iteration of the algorithm, where c is increased in value to put further emphasis on the constraints. As iterations continue $c \to \infty$ until it converges on a final solution of $\boldsymbol{x}^*$. One of the major issues with this algorithm is that as c gets larger the problem becomes increasingly ill-conditioned, which can cause unconstrained optimisation algorithms such as quasi-Newton and conjugate gradient methods to perform poorly.

Non-smooth penalty functions yield the exact solution of the non-linear programming programming after a single minimisation with respect to $\boldsymbol{x}$. An example of such a penalty function is the l_1 penalty function, defined by

$$\phi(\boldsymbol{x}; c) = f(\boldsymbol{x}) + c\sum_{i\in\mathcal{I}}[\kappa_i(\boldsymbol{x})]^- + c\sum_{j\in\mathcal{E}}|\kappa_j(\boldsymbol{x})|. \tag{A.13}$$

This penalty function penalises moves into the infeasible region sharply enough to produce an increase in the penalty function that is greater than $f(\boldsymbol{x}^*)$, thereby forcing the minimiser to lie at $\boldsymbol{x}^*$. While the advantage of these methods is that only one iteration of the minimisation is used, the minimisation of the functions is made difficult by the non-smoothness of the function.

The method of multipliers or the augmented Lagrangian method overcomes the difficulties associated with the previous two penalty methods. The method is related to the quadratic penalty algorithm, but it reduces the possibility of ill-conditioning by introducing explicit Lagrange multiplier estimates into the functions being minimised, known as the augmented Lagrangian function. Unlike the non-smooth penalty methods, the augmented Lagrangian generally preserves smoothness and so can easily be used with software designed to minimise unconstrained or bound constrained problems.

The augmented Lagrangian function, $\mathcal{L}_A(\boldsymbol{x}, \boldsymbol{\lambda}; c)$, is given incorporating equality constraints, $\kappa_i(\boldsymbol{x})$, $i \in \mathcal{E}$ into the cost function:

$$\mathcal{L}_A(\boldsymbol{x}, \boldsymbol{\lambda}; c) = f(\boldsymbol{x}) + \sum_{j \in \mathcal{E}} (\lambda_j \kappa_j(\boldsymbol{x}) + \frac{c}{2} \kappa_j^2(\boldsymbol{x})) \tag{A.14}$$

If inequality constraints, $\kappa_i(\boldsymbol{x})$, $i \in \mathcal{I}$ are present it is possible to incorporate these into the cost function by adding slack variables, such that inequality constraint $i \in \mathcal{I}$ becomes $\kappa_i(\boldsymbol{x}) - s_i = 0$, with $s_i \geq 0$. The slack variables s_i are added to the optimisation vector $\boldsymbol{x}$ to be optimised.

The augmented Lagrangian optimisation algorithm is an iterative process, performing alternating updates of the Lagrange multiplier estimates and optimisations with respect to $\boldsymbol{x}$. Also convergence of the method to $(\boldsymbol{x}^*, \lambda^*)$ is assured without increasing c indefinitely. The augmented Lagrangian algorithm used in this thesis is given as follows:

1. Pick a suitable c, and a starting point $(\boldsymbol{x}(1), \boldsymbol{\lambda}(1))$.
2. Keeping $\boldsymbol{\lambda}(k)$ constant, at iteration k of the algorithm, minimise $\mathcal{L}_A(\boldsymbol{x}(k), \boldsymbol{\lambda}(k); c)$ with respect to $\boldsymbol{x}$.
3. After optimisation, update the Lagrange multipliers as follows:

$$\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + c\boldsymbol{\kappa}(\boldsymbol{x}(k)) \tag{A.15}$$

4. The algorithm terminates when

$$\|\boldsymbol{\lambda}(k+1) - \boldsymbol{\lambda}(k)\|_\infty \leq \epsilon \tag{A.16}$$

   where ϵ is a small positive constant and $\|.\|_\infty$ tolerance term.

Due to its intuitive formulation and it's advantages over the previous penalty methods, as mentioned previously, the use of the augmented Lagrangian is quite popular. Also, as will be seen in the following chapter, due to its construction the augmented Lagrangian formulation allows certain large-scale optimisation problems to be decomposed into a number of smaller scale problems which can then be solved in a sequential fashion.

The methods outlined in this section allow optimisation problems to be solved in an efficient manner and in short time spans, thus enabling methods like MPC to be implemented quickly to allow real time control of systems.

References

J. Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operation Research and Financial Engineering. Springer, New York, USA, New York, USA, 2006.