Delft University of Technology

Delft Center for Systems and Control

Technical report 12-017

Weight optimisation for iterative distributed model predictive control applied to power networks*

P. Mc Namara, R.R. Negenborn, B. De Schutter, and G. Lightbody

If you want to cite this report, please use the following reference instead:

P. Mc Namara, R.R. Negenborn, B. De Schutter, and G. Lightbody, "Weight optimisation for iterative distributed model predictive control applied to power networks," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 532–543, Jan. 2013. doi:10.1016/j.engappai.2012.06.003

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

Weight Optimisation for Iterative Distributed Model Predictive Control Applied to Power Networks

Paul Mc Namara[a], Rudy R. Negenborn[b], Bart De Schutter[c], Gordon Lightbody[a]

[a]*Control and Intelligent Systems Group, Electrical and Electronic Engineering Department, University College Cork, Cork, Ireland.*
[b]*Department of Marine and Transport Technology, Delft University of Technology, Delft, The Netherlands.*
[c]*Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands.*

Abstract

This paper presents a weight tuning technique for iterative distributed Model Predictive Control (MPC). Particle Swarm Optimisation (PSO) is used to optimise both the weights associated with disturbance rejection and the weights associated with achieving consensus between control agents (while this paper focuses on disturbance rejection, the same techniques could also be used for set-point tracking based weight optimisation). Unlike centralised MPC, where tuning focuses solely on disturbance rejection performance, iterative distributed MPC practitioners must concern themselves with the trade off between disturbance rejection and the overall communication overhead when tuning weights. This is particularly the case in large scale systems, such as power networks, where typically there will be a large communication overhead associated with control. This paper examines the effects of weight optimisation on both the disturbance rejection and the communication overhead. Two PSO fitness functions are employed; the first function evaluates fitness based solely on disturbance rejection ability, and the second is based on achieving a trade off between good disturbance rejection ability and the maximum number of distributed MPC iterations per control step. Simulation experiments illustrate the potential of the proposed approach for weight tuning in two different power system scenarios.

Keywords: Distributed Model Predictive Control, Particle Swarm Optimisation, Smart Grids, Weight tuning, Power networks.

1. Introduction

Model Predictive Control (MPC) (Maciejowski, 2002) is an optimal control technique, in which the controller 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 is a mature technology at this stage, with stability and robustness analysis well established.

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 desirable or necessary to use a number of separate controllers, referred to as agents, to control subsystems, e.g., in a deregulated power market several controllers may be responsible for the control of different sections of the power grid. Likewise, when power systems span several countries, then each country will typically have separate controllers for their own sections of the grid.

There has been much research interest in recent years in distributed MPC (Scattolini, 2009; Sanchez et al., 2011; Liu et al., 2010), in which several agents communicate and cooperate with each other to approximate the behaviour of a centralised MPC agent. Lyapunov-based MPC techniques (Liu et al., 2010, 2009), game-theoretic approaches (Sanchez et al., 2011), and iterative techniques based on the decomposition of the original control problem into several smaller problems (Negenborn et al., 2008; Venkat, 2006) have all been used to distribute the control amongst agents.

Weights are used in MPC to determine the relative importance of the various goals contained in the MPC cost function during optimisation. By tuning the weights appropriately, MPC system performance can be improved, given a specific performance criterion. The equivalent in classical control would be the tuning of PID parameters to achieve desired performance. In classical control scenarios, stochastic search techniques such as Simulated Annealing (Kwok and Sheng, 1994), Genetic Algorithms (GA) (Jones and De Moura Oliveira, 1995), and Particle Swarm Optimisation (PSO) (Gaing, 2004) have proven to be efficient ways of finding optimal solutions for PID gains.

GAs have proven to be particularly popular as a tool for tuning PID gains (Fabijanski and Lagoda, 2008; Jones and De Moura Oliveira, 1995; Lin and Liu, 2010). However, recently the advantages of using PSO over GA for the optimisation of PID gains have been demonstrated, in terms of both the quality of and the efficiency with which a final solution can be found (Gaing, 2004). The reason for this is that PSO performs better than GA when optimising epistatic functions, i.e., functions where there is a high level of correlation between the

Email addresses: paulmcn@rennes.ucc.ie (Paul Mc Namara), r.r.negenborn@tudelft.nl (Rudy R. Negenborn), b.deschutter@tudelft.nl (Bart De Schutter), g.lightbody@ucc.ie (Gordon Lightbody)

parameters being optimised.

MPC gains can be automatically tuned in the same way as PID gains. There are a number of techniques in the literature for 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. PSO is attractive for tuning for a number of reasons. Its capability to optimise on a wide range of surfaces which may be convex, non-convex, discontinuous or multi-modal (Trelea, 2003; Liang et al., 2006) 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 preferable in comparison with GA for weight optimisation due to the level of correlation between weights when optimising in an MPC setting.

While typically in centralised MPC one is concerned with tuning weights so as to fulfil a set-point 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 set-point 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 needed 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 paper, a novel PSO based weight optimisation algorithm is proposed for the serial iterative distributed MPC technique proposed in (Negenborn et al., 2008). 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 the original optimisation that is based solely on the disturbance rejection performance (while the criteria in this paper are based on disturbance rejection, the same technique could be applied to optimise for set-point 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 a smaller scale, continuous-time multiple High Voltage Direct Current (HVDC) link problem. 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.

This paper is organised as follows. In Section 2, centralised MPC scheme and the iterative distributed MPC used in this paper, will be presented. Section 3 will introduce the PSO algorithm in detail. In Section 4 it will be shown how PSO is used to optimise the weights of the iterative distributed MPC algorithm. It will then be seen, in Section 5, how this weight optimisation affects the performance of two multi-agent power networks. Conclusions will be outlined in Section 6.

2. Model Predictive Control

Model Predictive Control (Maciejowski, 2002) is an advanced control technique that utilises real-time optimisation. It uses predictions, based on a suitable model, in order to provide optimal control inputs to a system. One of its main advantages over other control techniques is its ability to incorporate constraints into its formulation. It is at this stage a mature technology, with feasibility, stability, and robustness proofs well established (Rawlings and Mayne, 2009).

2.1. Definition of an agent

For clarity the definition of an agent, as understood in this paper, will now be provided. An agent is defined here as an entity responsible for the 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.

2.2. Description and state space prediction

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 the optimal inputs over this horizon. 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 each time step.

A system consisting of n subsystems is considered, 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. Subsystems are assumed to be non-overlapping, i.e., nodes do not appear in 2 different subsystems. A discrete-time, linear, time-invariant state-space model is used to model the subsystem dynamics. This is given as follows:

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

$$\boldsymbol{y}_a(k) = \boldsymbol{C}_a^{\mathrm{x}}\boldsymbol{x}_a(k) + \boldsymbol{C}_a^{\mathrm{u}}\boldsymbol{u}_a(k) + \boldsymbol{C}_a^{\mathrm{d}}\boldsymbol{d}_a(k) + \boldsymbol{C}_a^{\mathrm{v}}\boldsymbol{v}_a(k), \tag{2}$$

where $\boldsymbol{x}_a(k)$ is the state of the a^{th} subsystem, $\boldsymbol{u}_a(k)$ are local subsystem inputs, $\boldsymbol{d}_a(k)$ are known disturbances, $\boldsymbol{y}_a(k)$ are subsystem outputs, $\boldsymbol{v}_a(k)$ are external inputs from other subsystems

that influence subsystem a at sample step k, and $\boldsymbol{A}_a$, $\boldsymbol{B}_a$, $\boldsymbol{D}_a$, $\boldsymbol{V}_a$, $\boldsymbol{C}_a^{\mathrm{x}}$, $\boldsymbol{C}_a^{\mathrm{u}}$, $\boldsymbol{C}_a^{\mathrm{d}}$, and $\boldsymbol{C}_a^{\mathrm{v}}$ are the state-space matrices.

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 predictions for subsystem a over the prediction horizon are then determined using (1) as follows:

$$\tilde{\boldsymbol{x}}_a(k+1) = \boldsymbol{A}_a^{\mathrm{f}}\boldsymbol{x}_a(k) + \boldsymbol{B}_a^{\mathrm{f}}\tilde{\boldsymbol{u}}_a(k) + \boldsymbol{D}_a^{\mathrm{f}}\tilde{\boldsymbol{d}}_a(k) + \boldsymbol{V}_a^{\mathrm{f}}\tilde{\boldsymbol{v}}_a(k) \tag{3}$$

where $\boldsymbol{A}_a^{\mathrm{f}}$, $\boldsymbol{B}_a^{\mathrm{f}}$, $\boldsymbol{D}_a^{\mathrm{f}}$, $\boldsymbol{V}_a^{\mathrm{f}}$ are the state space prediction matrices. The derivation of these matrices is well established in the literature (Maciejowski, 2002).

2.3. MPC formulations

In a system of n subsystems, with agents $1, \dots, n$, assume initially that agent a has access to $\boldsymbol{x}_a(k)$, $\tilde{\boldsymbol{d}}_a(k)$, and $\tilde{\boldsymbol{v}}_a(k)$. The following optimisation problem is then solved at each time step:

$$\begin{aligned} \tilde{\boldsymbol{u}}_a(k) = \arg\min_{\tilde{\boldsymbol{u}}_a(k)} & J_a^{\mathrm{local}}(\boldsymbol{x}_a(k), \tilde{\boldsymbol{u}}_a(k), \tilde{\boldsymbol{d}}_a(k), \tilde{\boldsymbol{v}}_a(k)) \\ \text{subject to constraints} & \begin{cases} c_i(\boldsymbol{x}) = 0, & i \in \mathcal{E} \\ d_a(\boldsymbol{x}) \geq 0, & a \in \mathcal{I}. \end{cases} \end{aligned} \tag{4}$$

where $\mathcal{I}$ and $\mathcal{E}$ are two finite sets of indices, and c_i, for $i \in \mathcal{E}$, and d_a, for $a \in \mathcal{I}$, are the equality and inequality constraints of the problem, respectively. The local cost of subsystem a at sample time k is (henceforth, $J_a^{\mathrm{local}}(\boldsymbol{x}_a(k), \tilde{\boldsymbol{u}}_a(k), \tilde{\boldsymbol{d}}_a(k), \tilde{\boldsymbol{v}}_a(k))$ is denoted as $J_a^{\mathrm{local}}(k)$),

$$J_a^{\mathrm{local}}(k) = \sum_{l=0}^{N-1} J_a^{\mathrm{stage}}(k, p), \tag{5}$$

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

$$\begin{aligned} J_a^{\mathrm{stage}}(k, p) = & \boldsymbol{e}_a^{\mathrm{T}}(k+p+1)\boldsymbol{Q}_a\boldsymbol{e}_a(k+p+1) \\ & + \Delta\boldsymbol{u}_a^{\mathrm{T}}(k+p)\boldsymbol{R}_a\Delta\boldsymbol{u}_a(k+p), \end{aligned} \tag{6}$$

where $\boldsymbol{e}_a(k+p)$ is the vector of errors for agent a at the p^{th} stage of the prediction horizon, at sample time k. The error, $\boldsymbol{e}_a(k+p) = \boldsymbol{y}_a(k+p) - \boldsymbol{r}_a(k+p)$, where $\boldsymbol{r}_a(k+p)$ is a vector of the set-points of agent a.

The weighting matrices $\boldsymbol{Q}_a$ and $\boldsymbol{R}_a$ 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. Using the stage cost in (6), $J_a^{\mathrm{local}}(k)$ represents the desire to minimise the square of the error over the prediction horizon, i.e., to follow as closely as possible the set-point over the prediction horizon.

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

The vector of all interconnecting inputs $\tilde{\boldsymbol{w}}_a^{\mathrm{in}}(k)$, and all interconnecting outputs $\tilde{\boldsymbol{w}}_a^{\mathrm{out}}(k)$ to agent a are typically defined as follows:

$$\begin{aligned} \tilde{\boldsymbol{w}}_a^{\mathrm{in}}(k) &= \left[(\tilde{\boldsymbol{w}}^{\mathrm{in}}_{\mathcal{N}_a^{\mathrm{in}}\{1\}a}(k))^{\mathrm{T}} \dots (\tilde{\boldsymbol{w}}^{\mathrm{in}}_{\mathcal{N}_a^{\mathrm{in}}\{m_a\}a}(k))^{\mathrm{T}}\right]^{\mathrm{T}} = \tilde{\boldsymbol{v}}_a(k), \\ \tilde{\boldsymbol{w}}_a^{\mathrm{out}}(k) &= \left[(\tilde{\boldsymbol{w}}^{\mathrm{out}}_{\mathcal{N}_a^{\mathrm{out}}\{1\}a}(k))^{\mathrm{T}} \dots (\tilde{\boldsymbol{w}}^{\mathrm{out}}_{\mathcal{N}_a^{\mathrm{out}}\{g_a\}a}(k))^{\mathrm{T}}\right]^{\mathrm{T}} = \boldsymbol{K}_a^{\mathrm{out}}\tilde{\boldsymbol{x}}_a^{\mathrm{T}}(k), \end{aligned} \tag{7}$$

where $\mathcal{N}_a^{\mathrm{in}}\{i\}$ denotes the i^{th} agent connected to agent a by an interconnecting input, $\mathcal{N}_a^{\mathrm{out}}\{i\}$ denotes the i^{th} agent connected to agent a by an interconnecting output, m_a agents are connected to agent a by an interconnecting input, g_a agents are connected to agent a by an interconnecting output, and $\boldsymbol{K}_a^{\mathrm{out}}$ is a matrix of zeros, with entries of 1 used in the positions that pick out the states in $\tilde{\boldsymbol{x}}_a(k)$ that connect agent a to other subnetworks.

When many subsystems are interconnected, then knowledge of $\tilde{\boldsymbol{w}}_{ja}^{\mathrm{in}}(k)$ cannot be assumed, as $\tilde{\boldsymbol{w}}_{ja}^{\mathrm{in}}(k)$ is actually dependent on the dynamics of other subsystems. Hence, subsystems must reach a consensus on values for these interconnecting variables.

Centralised, decentralised and distributed control schemes based on the above formulation will now be presented.

2.3.1. Centralised MPC

In centralised MPC, instead of each subsystem having its own control agent, one central agent controls the whole system, solving all the individual subsystem MPC problems simultaneously. For a system of n subsystems, the combined overall optimisation problem can be formed as follows:

$$\min_{\tilde{\boldsymbol{u}}_1(k), \dots, \tilde{\boldsymbol{u}}_n(k)} \sum_{a=1}^{n} J_a^{\mathrm{local}}(k), \text{ subject to constraints,} \tag{8}$$

and subject to the following equality constraints,

$$\tilde{\boldsymbol{w}}_{ja}^{\mathrm{in}}(k) = \tilde{\boldsymbol{w}}_{aj}^{\mathrm{out}}(k), \text{ for } j \in \mathcal{N}_a. \tag{9}$$

That is, all interconnecting variables are made equal to each other over the prediction horizon according to the dynamics of each subsystem, as given in (3).

However, often the implementation of centralised MPC can be impractical due to technical constraints, including excessive computational load and the fact that several separate agents may be responsible for the control of different connected subsystems. This later situation can occur when different controllers in different countries control sections of connected power grids. Therefore, several agents are used to control their respective subsystems and the behaviour of these agents together approximates the behaviour of the centralised MPC.

2.3.2. Decentralised MPC

Decentralised MPC schemes assume that interconnected subsystems interact weakly and so ignore the effects of interactions with other subsystems in their MPC problems. Agents do not communicate with each other and independently solve an optimisation problem using only the local state variables of each subsystem, without seeking to achieve consensus amongst connected subsystems. When using decentralised MPC it is presumed that the effect of feedback in subsystems is sufficient

to overcome the effects of interactions with other subnetworks. However, ignoring these interactions between subsystems can lead to highly suboptimal behaviour and instability (Venkat, 2006).

2.3.3. Distributed case

In distributed MPC systems, inter-agent communication is used in order to coordinate system control actions. An augmented Lagrangian formulation can be developed from (8) to incorporate the equality constraints (9) into the cost function. In (Negenborn et al., 2008) the quadratic terms of the augmented Lagrangian formulation are distributed across the agents using a Block Coordinate Descent approach (Royo, 2001; Tosserams et al., 2008). This is an iterative algorithm, where a distributed MPC cycle, consisting of a number of iterations, occurs at each sample step. In this approach, one agent at a time optimises values for its inputs, $\tilde{\boldsymbol{u}}_a(k)$, and its desired interconnecting input variables $\tilde{\boldsymbol{w}}^{\text{in}}_{ja}(k)$, 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 time step k is:

$$\min_{\tilde{\boldsymbol{u}}_a(k,l),\{\tilde{\boldsymbol{w}}^{\text{in}}_{ja}(k,l):\, j\in\mathcal{N}_a\}} \left(J^{\text{local}}_a(k) + J^{\text{inter}}_a(k,l) \right), \tag{10}$$

where $J^{\text{inter}}_a(k,l)$ is the interconnection cost for agent a, given by:

$$J^{\text{inter}}_a(k,l) = \sum_{j\in\mathcal{N}_a} J^{\text{inter}}_{ja}(k,l), \tag{11}$$

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

$$\begin{aligned} J^{\text{inter}}_{ja}(k,l) = &\begin{bmatrix} \tilde{\boldsymbol{\lambda}}^{\text{in}}_{ja}(k,l) \\ -\tilde{\boldsymbol{\lambda}}^{\text{in}}_{aj}(k,l) \end{bmatrix}^{\text{T}} \begin{bmatrix} \tilde{\boldsymbol{w}}^{\text{in}}_{ja}(k,l) \\ \tilde{\boldsymbol{w}}^{\text{out}}_{ja}(k,l) \end{bmatrix} \\ &+ \frac{c}{2} \left\| \begin{bmatrix} \tilde{\boldsymbol{w}}^{\text{in}}_{aj,\text{prev}}(k,l) - \tilde{\boldsymbol{w}}^{\text{out}}_{ja}(k,l) \\ \tilde{\boldsymbol{w}}^{\text{out}}_{aj,\text{prev}}(k,l) - \tilde{\boldsymbol{w}}^{\text{in}}_{ja}(k,l) \end{bmatrix} \right\|^2_2 . \end{aligned} \tag{12}$$

Here c is a positive constant and $\tilde{\boldsymbol{\lambda}}^{\text{in}}_{ja}(k,l)$ are the Lagrange multipliers associated with the interconnecting constraints $\tilde{\boldsymbol{w}}^{\text{in}}_{ja}(k,l) = \tilde{\boldsymbol{w}}^{\text{out}}_{aj}(k,l)$ at iteration l, and time step k.

Each agent optimises this cost in a serial fashion, communicating the interconnecting variables with its neighbours. The values $\tilde{\boldsymbol{w}}^{\text{out}}_{aj,\text{prev}}(k,l)$ and $\tilde{\boldsymbol{w}}^{\text{in}}_{aj,\text{prev}}(k,l)$ are taken as the most recently updated values of $\tilde{\boldsymbol{w}}^{\text{out}}_{aj}(k,l)$ and $\tilde{\boldsymbol{w}}^{\text{in}}_{aj}(k,l)$, respectively.

One optimisation cycle has completed when all agents have performed an optimisation. When the optimisation cycle is finished, the Lagrange multipliers are updated as follows:

$$\tilde{\boldsymbol{\lambda}}^{\text{in}}_{ja}(k,l+1) = \tilde{\boldsymbol{\lambda}}^{\text{in}}_{ja}(k,l) + c\left(\tilde{\boldsymbol{w}}^{\text{in}}_{ja}(k,l) - \tilde{\boldsymbol{w}}^{\text{out}}_{aj}(k,l)\right), \tag{13}$$

Iterations continue until:

$$\begin{aligned} &\|\tilde{\boldsymbol{\lambda}}^{\text{in}}_{ja}(k,l+1) - \tilde{\boldsymbol{\lambda}}^{\text{in}}_{ja}(k,l)\|_\infty \le \epsilon \\ &\qquad \text{for } a = 1,\ldots,n \text{ and for all } j \in \mathcal{N}_a, \end{aligned} \tag{14}$$

where ϵ is a specified tolerance and $\|.\|_\infty$ denotes the infinity norm.

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 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. Therefore, it is of interest to develop methods that can be used to tune these parameters so as to give the desired control performance.

2.4. Application to shared inputs

In typical control applications, agents have their own local control inputs which are not shared among agents. However, in the multiple HVDC link application, discussed in Section 5.2 in this paper, all 4 agents must determine actions for the 2 control inputs. In other circumstances different agents' local inputs may be coupled, for example, via the objective function or through the system dynamics.

The algorithm in (Negenborn et al., 2008), used in this paper, naturally extends to such cases in the following way. Agents can create a duplicate variable vector, $\tilde{\boldsymbol{w}}^u_p(k)$, for agents $p = 1,\ldots,m_u$, that share the control input, $\tilde{\boldsymbol{u}}(k)$. These agents 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 can then be placed on the duplicate variables as follows: $\tilde{\boldsymbol{w}}^u_1(k) = \tilde{\boldsymbol{w}}^u_2(k)$, $\tilde{\boldsymbol{w}}^u_2(k) = \tilde{\boldsymbol{w}}^u_3(k)$,..., $\tilde{\boldsymbol{w}}^u_{m_u-1}(k) = \tilde{\boldsymbol{w}}^u_{m_u}(k)$, such that $\tilde{\boldsymbol{w}}^u_1(k) = \ldots = \tilde{\boldsymbol{w}}^u_{m_u}(k)$ 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 $\tilde{\boldsymbol{w}}^u_a(k)$ 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.

3. Particle Swarm Optimisation

Particle Swarm Optimisation (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 or non-convex, continuous or discontinuous surfaces. It works by initialising a number of candidate solutions, called particles, in the parameter space being searched, and then updating their positions over a number of iterations, in such a way so as to converge to a final (ideally global) optimal solution.

In PSO a population of P particles, each of dimension d, are initially distributed across the parameter space. Particle q at

iteration i of the PSO algorithm, has a position $\boldsymbol{x}_q(i)$, and associated fitness $f_q(i)$. Each of these particles has a memory of its own previous best position $\boldsymbol{x}_q^{\mathrm{b}}(i)$ and an associated fitness $f_q^{\mathrm{b}}(i)$. Here $\boldsymbol{x}^{\mathrm{g}}(i)$, the global best position, is the particle position associated with the best fitness $f^{\mathrm{g}}(i)$ that has been found previously across the population of particles. Particle q's position is then updated, biased towards both $\boldsymbol{x}^{\mathrm{g}}(i)$ and $\boldsymbol{x}_q^{\mathrm{b}}(i)$. The general 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 domain of the cost function. The evaluation of these initial positions is used to initialise $\boldsymbol{x}_q^{\mathrm{b}}$ for particle q, and $\boldsymbol{x}^{\mathrm{g}}$ is then initialised from the $\boldsymbol{x}_q^{\mathrm{b}}$ value with the lowest associated f_q^{b} in the entire swarm of P particles.
2. The velocity, $\boldsymbol{v}_{\mathrm{q}}(i)$ of particle q at the i^{th} iteration of the PSO algorithm is updated for the next iteration as follows:

$$\begin{aligned}\boldsymbol{v}_q(i+1) = \omega \boldsymbol{v}_q(i) + c_1 \boldsymbol{r}_1(i) \circ \left(\boldsymbol{x}_q^{\mathrm{b}}(i) - \boldsymbol{x}_q(i)\right) \\ + c_2 \boldsymbol{r}_2(i) \circ \left(\boldsymbol{x}^{\mathrm{g}}(i) - \boldsymbol{x}_q(i)\right)\end{aligned} \tag{15}$$

   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, and c_1 and c_2 are acceleration constants called the cognition and social components, respectively.

   Applied particle velocities are bounded by $\boldsymbol{v}_{q_{\min}} \leq \boldsymbol{v}_{q_{\mathrm{app}}} \leq \boldsymbol{v}_{q_{\max}}$ where $\boldsymbol{v}_{q_{\min}}$ and $\boldsymbol{v}_{q_{\max}}$ are the lower and upper bounds on particle velocities, respectively, and $\boldsymbol{v}_{q_{\mathrm{app}}}$ is the applied particle velocity. If $\boldsymbol{v}_q(i+1)$ exceeds the aforementioned velocity bounds, the applied velocity, $\boldsymbol{v}_{q_{\mathrm{app}}}$, is taken at the upper or lower bound, i.e., if $\boldsymbol{v}_q < \boldsymbol{v}_{q_{\min}}$, let $\boldsymbol{v}_{q_{\mathrm{app}}} = \boldsymbol{v}_{q_{\min}}$; if $\boldsymbol{v}_q > \boldsymbol{v}_{q_{\max}}$, let $\boldsymbol{v}_{q_{\mathrm{app}}} = \boldsymbol{v}_{q_{\max}}$; else let $\boldsymbol{v}_{q_{\mathrm{app}}} = \boldsymbol{v}_q$.

   The position, $\boldsymbol{x}_q(i)$, of particle q at the i^{th} iteration of the PSO algorithm is then updated for the next iteration as follows:

$$\boldsymbol{x}_q(i+1) = \boldsymbol{x}_q(i) + \boldsymbol{v}_{q_{\mathrm{app}}}(i+1). \tag{16}$$

3. Evaluate the cost function values at each of the P particles' positions.
4. If for particle q, $f_q(i+1) < f_q^{\mathrm{b}}(i)$, then let $\boldsymbol{x}_q^{\mathrm{b}}(i+1) = \boldsymbol{x}_q(i+1)$.
   If $f_q(i+1) < f^{\mathrm{g}}(i)$, let $\boldsymbol{x}^{\mathrm{g}}(i+1) = \boldsymbol{x}_q(i+1)$.
   If $f_q(i+1) \geq f_q^{\mathrm{b}}(i) \geq f^{\mathrm{g}}(i)$, then $\boldsymbol{x}_q^{\mathrm{b}}(i+1)$ and $\boldsymbol{x}^{\mathrm{g}}(i+1)$ remain at the same positions as in iteration i.
5. Repeat steps (2)-(4) until termination criteria are met, e.g., a maximum amount of iterations has been reached, $\boldsymbol{x}^{\mathrm{g}}(i)$ has not changed for a given number of iterations, etc.

4. PSO Weight Optimisation for Distributed MPC

PSO has previously been used to optimise the centralised MPC weights (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 $\boldsymbol{\Gamma} = [\gamma_1 \dots \gamma_{n_\gamma}]^{\mathrm{T}}$ 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 paper, 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 PSO optimisation. The sum of the local fitnesses of all n agents, $\sum_{j=1}^{n} f_j^{\mathrm{local}}(q,i)$, 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

$$f_q(i) = \sum_{j=0}^{n} f_j^{\mathrm{local}}(q,i), \tag{17}$$

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

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

$$\rho(q,i) = \max(\boldsymbol{\mu}(q,i)), \tag{18}$$

where $\boldsymbol{\mu}(q,i)$ 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:

$$f_q(i) = \upsilon_\rho \rho(q,i) + \sum_{j=0}^{n} f_j^{\mathrm{local}}(q,i), \tag{19}$$

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 optimisation algorithm is as follows:

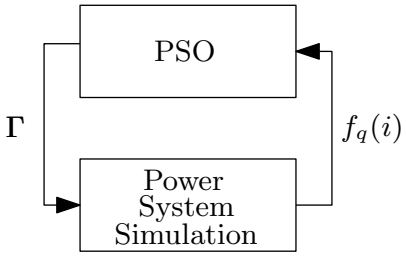


Figure 1: The PSO optimisation of the distributed MPC weights at iteration i of the PSO optimisation.

1. A random population of P particles is initialised in d dimensions, with $\boldsymbol{x}_{\min} \leq \boldsymbol{x}_q \leq \boldsymbol{x}_{\max}$, where $\boldsymbol{x}_{\min} \geq 0$ and $\boldsymbol{x}_{\max} \geq 0$ 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 initialised 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 q's personal best position $\boldsymbol{x}_q^{\text{b}}(i)$ and fitnesses corresponding to these positions, $f_q^{\text{b}}(i)$, for $q = 1,\ldots,P$, 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 (15) and (16). Then with these P particles, the algorithm repeats from step (2).
4. The algorithm terminates when a termination criterion is satisfied; in this paper this happens when $f^{\text{g}}(i)$ has not changed by more than a small specified tolerance for a given number of PSO iterations.

5. 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) controlled using distributed MPC, are given in this section. 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 matches 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 set-point 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. First a discrete-time power network consisting of 20 subsystems is considered, and in the second experiment a continuous-time multiple link HVDC system, 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 (19). The disturbance rejection criterion for the j^{th} agent in the simulation, run by particle q in the swarm, at PSO iteration i, used in both cases presented here is the Integral of the Square of Time by the Squared

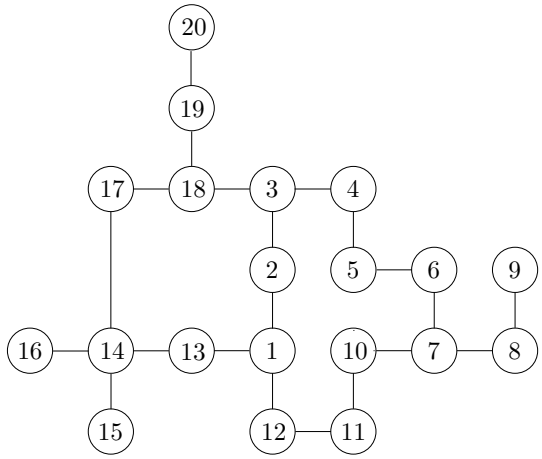


Figure 2: The 20 area discrete-time LFC problem.

Error (ISTSE) (Gambier, 2007), which is used to place greater emphasis on long term errors over short term errors that occur immediately after disturbances, given by

$$f_j^{\text{local}}(q,i) = \sum_{k=0}^{\infty} k^2 e_j^2(k), \tag{20}$$

where $e_j(k)$ is the error in subsystem j at sample time k, where the error at sample time k is the difference between the measured and 1 pu frequencies, 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). 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 for this work. Other parameters used in the PSO toolbox are given in Appendix B.

5.1. System 1: 20 area discrete-time LFC problem

The 20 area discrete time LFC problem is shown in Fig. 2. The continuous-time dynamics of subsystem a are described by the following second-order system (Negenborn et al., 2008):

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t}\Delta\delta_a(t) = 2\pi\Delta f_a(t),\\
&\frac{\mathrm{d}}{\mathrm{d}t}\Delta f_a(t) = -\frac{1}{T_{\mathrm{p}_a}}\Delta f_a(t) + \frac{K_{\mathrm{p}_a}}{T_{\mathrm{p}_a}}\Big(\Delta P_a^{\text{gen}}(t) - \Delta P_a^{\text{dist}}(t)\\
&\qquad\qquad + \sum_{j\in\mathcal{N}_a} \frac{K_{\mathrm{S}_{aj}}}{2\pi}\left(\Delta\delta_j(t) - \Delta\delta_a(t)\right)\Big),\\
&\boldsymbol{y}_a(t) = \begin{bmatrix}\Delta\delta_a(t)\\ \Delta f_a(t)\end{bmatrix}.
\end{aligned} \tag{21}$$

where at time t, $\Delta\delta_a(t)$ represents the angle of a generator a, $\Delta f_a(t)$ the frequency of generator a, $\Delta P_a^{\text{gen}}(t)$ the power generation of generator a and $\Delta P_a^{\text{dist}}(t)$ the load disturbance, and Δ

in each of these variables denotes a deviation from an original equilibrium position. Here $\boldsymbol{y}_a$ represents the measured output states, and subnetwork a's gain K_{P_a}, its time constant T_{P_a}, and the synchronising coefficient between areas a and j, $K_{\mathrm{S}_{aj}}$, are all constants. For the purposes of the simulations output measurements $\boldsymbol{y}_a$ are assumed to be noise free.

In discrete time, the local control input is $u_a(k) = \Delta P_a^{\mathrm{gen}}(k)$, the local disturbance is $d_a(k) = \Delta P_a^{\mathrm{dist}}(k)$, and the local state is $\boldsymbol{x}_a(k) = [\Delta\delta_a(k), \Delta f_a(k)]^{\mathrm{T}}$. The external inputs from other subnetworks are $\boldsymbol{v}_a(k) = [\Delta\delta_{\mathcal{N}_a^{\mathrm{in}}\{1\}}(k), \ldots, \Delta\delta_{\mathcal{N}_a^{\mathrm{in}}\{m_a\}}(k)]^{\mathrm{T}}$, where m_a is the number of subnetworks connected to subnetwork a, and $\mathcal{N}_a^{\mathrm{in}}\{i\}$ is the i^{th} agent connected to agent a. Discretising the continuous time model using an Euler approximation (with step size τ=0.2s), the model can be written as in (1) with:

$$\begin{aligned}
\boldsymbol{A}_a &= \begin{bmatrix} 1 & \tau 2\pi \\ \sum_{j\in\mathcal{N}_a} \tau\dfrac{-K_{\mathrm{P}_a}K_{\mathrm{S}_{aj}}}{2\pi T_{\mathrm{P}_a}} & 1-\frac{\tau}{T_{\mathrm{P}_a}} \end{bmatrix}, \boldsymbol{B}_a = \begin{bmatrix} 0 \\ \tau\frac{K_{\mathrm{P}_a}}{T_{\mathrm{P}_a}} \end{bmatrix}, \\
\boldsymbol{D}_a &= \begin{bmatrix} 0 \\ -\tau\frac{K_{\mathrm{P}_a}}{T_{\mathrm{P}_a}} \end{bmatrix}, \boldsymbol{V}_a = \begin{bmatrix} 0 & \ldots & 0 \\ \tau\dfrac{K_{\mathrm{P}_a}K_{\mathrm{S}_{a\mathcal{N}_a^{\mathrm{in}}\{1\}}}}{2\pi T_{\mathrm{P}_a}} & \ldots & \tau\dfrac{K_{\mathrm{P}_a}K_{\mathrm{S}_{a\mathcal{N}_a^{\mathrm{in}}\{m_a\}}}}{2\pi T_{\mathrm{P}_a}} \end{bmatrix}.
\end{aligned} \tag{22}$$

The above model is used to run the discrete time simulation with a sample time of 0.2s. All subnetworks' parameters are identical and are given as follows: $K_{\mathrm{P}_a} = 120$, $T_{\mathrm{P}_a} = 20$, $K_{\mathrm{S}_{aj}} = 0.5$.

5.1.1. Controller description

An incremental state space model is used for control of the system so as to ensure integral action. The augmented state for agent a at sample k is defined as $\boldsymbol{x}_a^{\mathrm{aug}}(k) = [\Delta\boldsymbol{x}_a(k), \boldsymbol{x}_a(k)]^{\mathrm{T}}$, where $\Delta\boldsymbol{x}_a(k) = \boldsymbol{x}_a(k) - \boldsymbol{x}_a(k-1)$, and incremental values of $\boldsymbol{u}_a(k)$ and $\boldsymbol{v}_a(k)$, $\Delta\boldsymbol{u}_a(k) = \boldsymbol{u}(k) - \boldsymbol{u}(k-1)$ and $\Delta\boldsymbol{v}_a(k) = \boldsymbol{v}(k) - \boldsymbol{v}(k-1)$, respectively, are used with their associated state space models for the control of the system (these are derived as in Wang (2009)). A prediction horizon of $N = 10$ was used to adequately take into account each subnetwork's dynamic response.

The following stage quadratic cost function is used for the distributed MPC:

$$J_a^{\mathrm{stage}}(k,p) = Q_a\Delta f_a^2(k+p+1) + R_a\Delta u_a^2(k+p) \tag{23}$$

where Q_a, 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 (5).

The interconnecting inputs of the a^{th} agent are each $\Delta\delta_j$ for $j \in \mathcal{N}_a$ and the interconnecting output is $\Delta\delta_a$. The following gives the interconnection cost agent a experiences due to its connection to agent j:

$$\begin{aligned}
J_{aj}^{\mathrm{inter}}(k,l) = &\begin{bmatrix} \tilde{\boldsymbol{\lambda}}_{ja}^{\mathrm{in},\Delta\boldsymbol{\delta}_j}(l) \\ -\tilde{\boldsymbol{\lambda}}_{aj}^{\mathrm{in},\Delta\boldsymbol{\delta}_a}(l) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \tilde{\boldsymbol{w}}_{ja}^{\mathrm{in},\Delta\boldsymbol{\delta}_j}(k) \\ \tilde{\boldsymbol{w}}_{ja}^{\mathrm{out},\Delta\boldsymbol{\delta}_a}(k) \end{bmatrix} \\
&+ \frac{c}{2}\left\| \begin{bmatrix} \tilde{\boldsymbol{w}}_{aj,\mathrm{prev}}^{\mathrm{in},\Delta\boldsymbol{\delta}_j}(l) - \tilde{\boldsymbol{w}}_{ja}^{\mathrm{out},\Delta\boldsymbol{\delta}_a}(k) \\ \tilde{\boldsymbol{w}}_{aj,\mathrm{prev}}^{\mathrm{out},\Delta\boldsymbol{\delta}_a}(l) - \tilde{\boldsymbol{w}}_{ja}^{\mathrm{in},\Delta\boldsymbol{\delta}_a}(k) \end{bmatrix} \right\|_2^2,
\end{aligned} \tag{24}$$

for each $j \in \mathcal{N}_a$, where in each of the $\boldsymbol{\lambda}$ and $\boldsymbol{w}$ vectors above there is one entry for each step of the prediction horizon, i.e., for each agent j that agent a is connected to, there are $\boldsymbol{\lambda}_{ja}^{\mathrm{in}}$ and $\boldsymbol{w}_{ja}^{\mathrm{in}}$ terms over the full prediction horizon to determine what $\Delta\delta_j^{\mathrm{in}}$ values agent a would like to receive, and also there are $\boldsymbol{\lambda}_{ja}^{\mathrm{out}}$ and $\boldsymbol{w}_{ja}^{\mathrm{out}}$ terms for each step of the prediction horizon based on the $\Delta\delta_a^{\mathrm{out}}$ terms that agent a would like to send to other agents.

The overall cost function for each agent is formed using (23) and (24), as in (10). Constraints on the inputs and states are as follows:

$$\begin{aligned}
u_a^{\min} \le u_a(k+l) \le u_a^{\max} \\
\boldsymbol{x}_a^{\min} \le \boldsymbol{x}_a(k+l) \le \boldsymbol{x}_a^{\max}
\end{aligned}$$

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

5.1.2. PSO optimisation of the distributed MPC weights

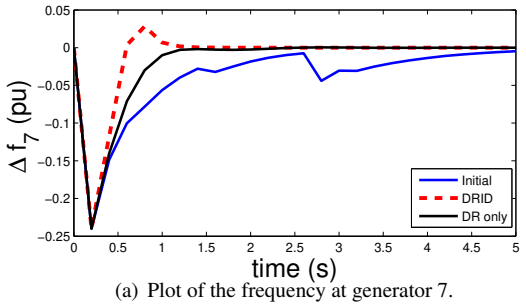
PSO is now used to optimise the weights and parameters of the distributed MPC system for the 20 area LFC system. Given the large number of agents in the system, it is non-trivial to find a combination of weights that give both good disturbance rejection performance and a low communications overhead.

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 $\boldsymbol{\Gamma} = [\, Q_1 \ \ldots \ Q_{20} \ \epsilon\,]^{\mathrm{T}}$. The R_a weights were not optimised and were given a value of 10^{-3}. However, these could be optimised if the practitioner desired so. The constraints for the variables in the PSO optimisation are as follows: $0.1 \le Q_a \le 100$, for agents $a = 1, \ldots, 20$, and $10^{-4} \le \epsilon \le 1$. For the PSO optimisations involving the iteration deterrent, $\upsilon = 2.5$.

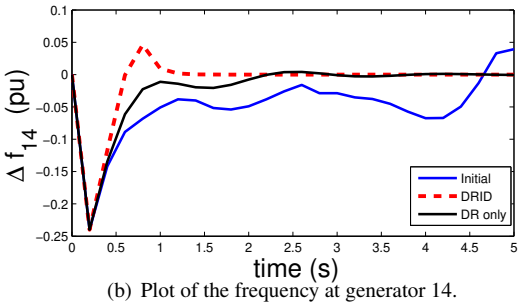
To save on the overall PSO simulation time an upper limit of 50 distributed MPC iterations is allowed in each simulation run of the power system. If this is exceeded at any stage a fitness of 1000 is allocated to the particle at that position and the simulation of the next particle begins. While this 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 $\tau = 0.2$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 being the largest that can be expected to occur in each area. This is 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 done using quadprog. The PSO terminates when f^{g} does not improve by more than 2.5×10^{-4} for 7 consecutive iterations.

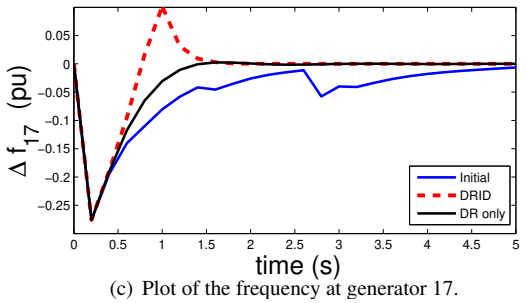
Finding a set a weights to control this system for this scenario is non-trivial. The best performance the authors could achieve before optimisation, by manually tuning parameters, was with $[\, Q_1 \ \ldots \ Q_{20} \ c \ \epsilon\,]^{\mathrm{T}} = [\, 10 \ \ldots \ 10 \ \ 1 \ \ 10^{-2}\,]^{\mathrm{T}}$. Fig. 3 shows the



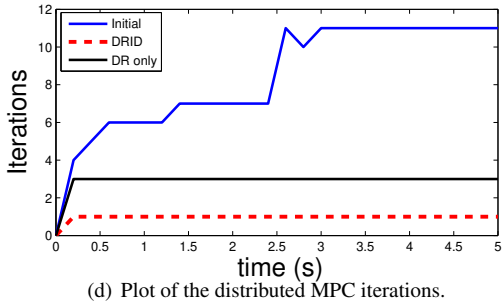
(a) Plot of the frequency at generator 7.



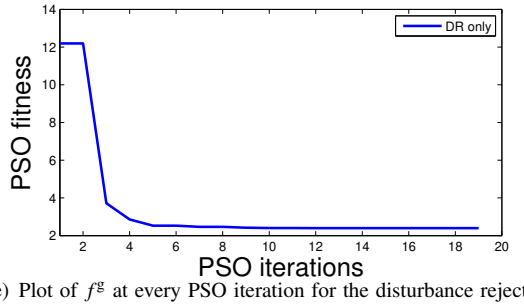
(b) Plot of the frequency at generator 14.



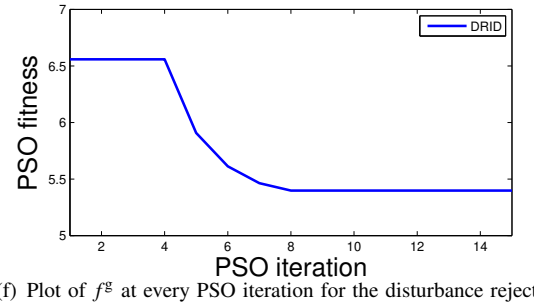
(c) Plot of the frequency at generator 17.



(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 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.

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. 3(b). The ISTSE for this performance is 131. A large number of distributed MPC iterations are needed to achieve this control too.

The frequencies in areas 7, 14, and 17 (used as sample illustrations of the effect of the weight optimisations) can be seen in Figs. 3(a), 3(b), and 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. 3(d), and f^{g} is plotted for each of the PSO iterations in the DR only and DRID cases in Figs. 3(e) and 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 f^{g}=2.3975. The maximum number of distributed MPC iterations needed to achieve this is 3 as can be seen in Fig. 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. The maximum number of iterations needed to achieve this control is only 1 in this case though.

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 achieved with the original tuning.

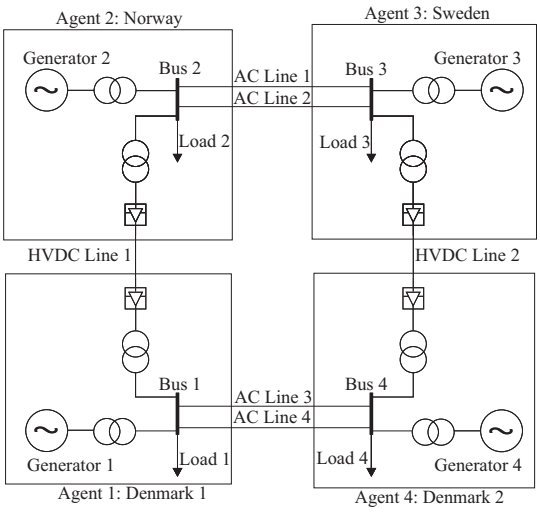


Figure 4: The multiple HVDC link system with areas controlled by agents (Erikkson, 2008).

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. 3(e) and 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.

5.2. System 2: Continuous-time Multiple HVDC link system

The system used in this section, based on the multiple HVDC link system between Denmark, Norway, and Sweden, is depicted in Fig. 4 (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). Large amounts of power are transferred from bus 2, which has the largest generation capacity, to bus 4, which has the largest power load. Generation capacities and loads are kept constant in this paper. A simplification in this paper is to assume no line losses, which means that the amount of power in the system is constant at all times, and so the modulation of the HVDC line powers alone is enough to stabilise the system after line faults.

Due to the high level of interconnectivity between individual agents problems this is quite a challenging distributed MPC problem. Finding a good combination of weights that balances 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 classical swing equations for a generator a are (Kundur, 1994):

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta_{\mathrm{r}_a}(t) = \omega_0 \Delta\omega_{\mathrm{r}_a}(t) \tag{25}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\omega_{\mathrm{r}_a}(t) = \frac{1}{2H_a}(P_{\mathrm{m}_a}(t) - P_{\mathrm{G}_a}(t) - D_a\Delta\omega_{\mathrm{r}_a}(t)), \tag{26}$$

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), $P_{\mathrm{m}_a}(t)$ and $P_{\mathrm{G}_a}(t)$ are the mechanical and generated power (pu), respectively, and D_a is the damping factor (pu).

By modelling the system using an internal node representation, which gives the system's generator currents in terms of the system's voltages and HVDC line currents (Mc Namara et al., 2011), the following swing equation for generator a can be found:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\omega_{\mathrm{r}_a} = \frac{1}{2H_a}\Big(&P_{\mathrm{m}_a} - G_{a,a}E_{\mathrm{q}_a}^{'2} - \\ &\sum_{\substack{l=1\\ l\neq a}}^{n} E_{\mathrm{q}_a}^{'}E_{\mathrm{q}_l}^{'}(G_{a,l}\cos(\delta_{\mathrm{r}_a} - \delta_{\mathrm{r}_l}) + B_{a,l}\sin(\delta_{\mathrm{r}_a} - \delta_{\mathrm{r}_l})) \\ &+ g_{a,1}P_1^{\mathrm{DC}} + \ldots + g_{a,m}P_m^{\mathrm{DC}} - D_a\Delta\omega_{\mathrm{r}_a}\Big),\end{aligned} \tag{27}$$

where 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 (details of how each of these are derived are given in (Mc Namara et al., 2011)).

It can be seen that this equation gives a relationship between the rotor acceleration of generator a, the rotor positions of each of the generators in the network, and the HVDC line powers. From this it can be seen that there is high level of interconnectivity between each of the subsystems' individual distributed MPC problems and a large communications overhead can be expected when agents need to coordinate their actions. The multiple HVDC link system parameters used in the simulation are given in Appendix A.

5.2.1. Controller description

At each sample the state equations for each generator are linearised about the current operating point as follows:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\delta_{\mathrm{r}_a}\\ \omega_{\mathrm{r}_a}\end{bmatrix} &= \begin{bmatrix}0 & \omega_0\\ \frac{\partial f_{\mathrm{r}_a}}{\partial \delta_{\mathrm{r}_a}}\Big|_{\mathrm{op}} & \frac{\partial f_{\mathrm{r}_a}}{\partial \omega_{\mathrm{r}_a}}\Big|_{\mathrm{op}}\end{bmatrix}\begin{bmatrix}\delta_{\mathrm{r}_a}\\ \omega_{\mathrm{r}_a}\end{bmatrix} \\ &+ \begin{bmatrix}0 & 0\\ \frac{\partial f_{\mathrm{r}_a}}{\partial P_1^{\mathrm{DC}}}\Big|_{\mathrm{op}} & \frac{\partial f_{\mathrm{r}_a}}{\partial P_2^{\mathrm{DC}}}\Big|_{\mathrm{op}}\end{bmatrix}\begin{bmatrix}P_1^{\mathrm{DC}}\\ P_2^{\mathrm{DC}}\end{bmatrix}_{\mathrm{op}} + \begin{bmatrix}0\\ \frac{\partial f_{\mathrm{r}_a}}{\partial \delta_{\mathrm{r}_l}}\Big|_{\mathrm{op}}\end{bmatrix}\delta_{\mathrm{r}_l}\end{aligned} \tag{28}$$

where in the above equation $f_{\mathrm{r}_a}(\delta_{\mathrm{r}_a}, \omega_{\mathrm{r}_a}, P_1^{\mathrm{DC}}, P_2^{\mathrm{DC}}, \delta_{\mathrm{r}_l}) = \frac{\mathrm{d}}{\mathrm{d}t}\omega_{\mathrm{r}_a}$, as defined in (27), and op indicates the linearisation of the relevant variable, vector, or function about the current operating point.

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 $\boldsymbol{v}_a = \delta_{\mathrm{r}_l}$. The full system model is discretised using a zero-order hold with a sample time $\tau = 0.01$s, providing the discrete-time state space equations for the distributed MPC system. Predictions are formed using incremental state space models so as to ensure integral action, i.e., the augmented state $\boldsymbol{x}_a^{\mathrm{aug}} = [\Delta\boldsymbol{x}_a^{\mathrm{T}}\ \boldsymbol{x}_a^{\mathrm{T}}]^{\mathrm{T}}$, incremental inputs $\Delta\boldsymbol{u}_a$, and incremental interconnecting inputs

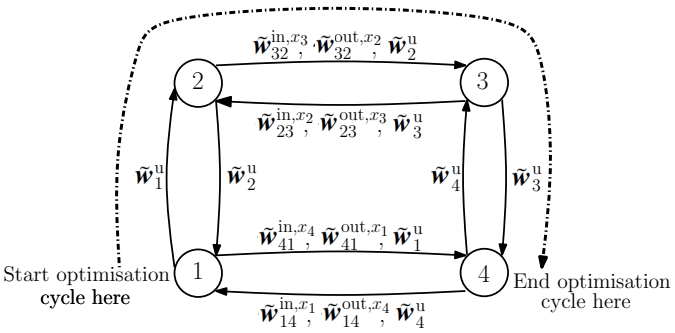


Figure 5: Order of serial distributed MPC optimisations and variables communicated between agents

$\Delta \boldsymbol{v}_a$ 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.

One agent is assigned per generator to control its frequency. Each agent has access to its relevant state space model, the constraints on its variables, and can communicate with agents to which it is connected by an AC or HVDC link.

The stage cost for the a^{th} agent (there is one agent for each generator, so for convenience the subscript a is used to index both), $J_a^{\text{stage}}(k,p)$, for the p^{th} prediction step at sample step k, is given as follows:

$$J_a^{\text{stage}}(k,p) = Q_a(\omega_{\text{r}_a}(k+p+1)-1)^2 + \Delta \boldsymbol{u}_a(k+p)^{\text{T}} \boldsymbol{R}_a \Delta \boldsymbol{u}_a(k+p), \tag{29}$$

where Q_a is the weight corresponding to ω_{r_a}, and $\boldsymbol{R}_a$ is a diagonal weight matrix corresponding to each of the inputs in $\Delta \boldsymbol{u}_a$ in the cost function. This cost function penalises deviations of the frequency from the base frequency and the control effort.

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:

$$\min_{\Delta\tilde{\boldsymbol{u}}_1,\ldots,\Delta\tilde{\boldsymbol{u}}_4} \sum_{a=1}^{4} \left(J_a^{\text{local}} \right) + \begin{bmatrix} \tilde{\boldsymbol{\lambda}}_{41}^{\text{in},\boldsymbol{x}_4} \\ \tilde{\boldsymbol{\lambda}}_{32}^{\text{in},\boldsymbol{x}_3} \\ \tilde{\boldsymbol{\lambda}}_{23}^{\text{in},\boldsymbol{x}_2} \\ \tilde{\boldsymbol{\lambda}}_{14}^{\text{in},\boldsymbol{x}_1} \\ \tilde{\boldsymbol{\lambda}}_{41}^{\text{u}} \\ \tilde{\boldsymbol{\lambda}}_{12}^{\text{u}} \\ \tilde{\boldsymbol{\lambda}}_{23}^{\text{u}} \\ \tilde{\boldsymbol{\lambda}}_{34}^{\text{u}} \end{bmatrix}^{\text{T}} \begin{bmatrix} \tilde{\boldsymbol{w}}_{41}^{\text{in},\boldsymbol{x}_4} - \tilde{\boldsymbol{w}}_{14}^{\text{out},\boldsymbol{x}_4} \\ \tilde{\boldsymbol{w}}_{32}^{\text{in},\boldsymbol{x}_3} - \tilde{\boldsymbol{w}}_{23}^{\text{out},\boldsymbol{x}_3} \\ \tilde{\boldsymbol{w}}_{23}^{\text{in},\boldsymbol{x}_2} - \tilde{\boldsymbol{w}}_{32}^{\text{out},\boldsymbol{x}_2} \\ \tilde{\boldsymbol{w}}_{14}^{\text{in},\boldsymbol{x}_1} - \tilde{\boldsymbol{w}}_{41}^{\text{out},\boldsymbol{x}_1} \\ \tilde{\boldsymbol{w}}_1^{\text{u}} - \tilde{\boldsymbol{w}}_4^{\text{u}} \\ \tilde{\boldsymbol{w}}_2^{\text{u}} - \tilde{\boldsymbol{w}}_1^{\text{u}} \\ \tilde{\boldsymbol{w}}_3^{\text{u}} - \tilde{\boldsymbol{w}}_2^{\text{u}} \\ \tilde{\boldsymbol{w}}_4^{\text{u}} - \tilde{\boldsymbol{w}}_3^{\text{u}} \end{bmatrix} + \frac{c}{2} \left\| \begin{matrix} \tilde{\boldsymbol{w}}_{41}^{\text{in},\boldsymbol{x}_4} - \tilde{\boldsymbol{w}}_{14}^{\text{out},\boldsymbol{x}_4} \\ \tilde{\boldsymbol{w}}_{32}^{\text{in},\boldsymbol{x}_3} - \tilde{\boldsymbol{w}}_{23}^{\text{out},\boldsymbol{x}_3} \\ \tilde{\boldsymbol{w}}_{23}^{\text{in},\boldsymbol{x}_2} - \tilde{\boldsymbol{w}}_{32}^{\text{out},\boldsymbol{x}_2} \\ \tilde{\boldsymbol{w}}_{14}^{\text{in},\boldsymbol{x}_1} - \tilde{\boldsymbol{w}}_{41}^{\text{out},\boldsymbol{x}_1} \\ \tilde{\boldsymbol{w}}_1^{\text{u}} - \tilde{\boldsymbol{w}}_4^{\text{u}} \\ \tilde{\boldsymbol{w}}_2^{\text{u}} - \tilde{\boldsymbol{w}}_1^{\text{u}} \\ \tilde{\boldsymbol{w}}_3^{\text{u}} - \tilde{\boldsymbol{w}}_2^{\text{u}} \\ \tilde{\boldsymbol{w}}_4^{\text{u}} - \tilde{\boldsymbol{w}}_3^{\text{u}} \end{matrix} \right\|_2^2, \tag{30}$$

where $\boldsymbol{w}_a^{\text{u}}(k) = [P_1^{\text{DC}}(k)\ P_2^{\text{DC}}(k)]^{\text{T}}$ is agent a's duplicate vector of the control inputs, and the ks and ls, used to denote the sample step and distributed MPC iteration, are omitted for compactness. 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 $\tilde{\boldsymbol{w}}_a^{\text{u}}(k)$. The order in which agents optimise for the distributed MPC cycles starts with agent 1 and ends with 4. 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 (30) is distributed amongst the agents, $J_a^{\text{inter}}(k,l)$ takes the following distributed form for agent a, where bus j is AC-connected to bus a (this is specific to the system used in this paper, where each agent AC-connected to only one other agent):

$$J_a^{\text{inter}} = \begin{bmatrix} \tilde{\boldsymbol{\lambda}}_{ja}^{\text{in},\boldsymbol{x}_j} \\ -\tilde{\boldsymbol{\lambda}}_{aj}^{\text{in},\boldsymbol{x}_a} \\ \tilde{\boldsymbol{\lambda}}_a^{\text{u}} \\ -\tilde{\boldsymbol{\lambda}}_{a,\text{next}}^{\text{u}} \end{bmatrix}^{\text{T}} \begin{bmatrix} \tilde{\boldsymbol{w}}_{ja}^{\text{in},\boldsymbol{x}_j} \\ \tilde{\boldsymbol{w}}_{ja}^{\text{out},\boldsymbol{x}_a} \\ \tilde{\boldsymbol{w}}_a^{\text{u}} \\ \tilde{\boldsymbol{w}}_a^{\text{u}} \end{bmatrix} + \frac{c}{2} \left\| \begin{matrix} \tilde{\boldsymbol{w}}_{aj,\text{prev}}^{\text{out},\delta_{\text{r}_j}} - \tilde{\boldsymbol{w}}_{ja}^{\text{in},\boldsymbol{x}_j} \\ \tilde{\boldsymbol{w}}_{aj,\text{prev}}^{\text{in},\boldsymbol{x}_a} - \tilde{\boldsymbol{w}}_{ja}^{\text{out},\boldsymbol{x}_a} \\ \tilde{\boldsymbol{w}}_{\text{last,prev}}^{\text{u}} - \tilde{\boldsymbol{w}}_a^{\text{u}} \\ \tilde{\boldsymbol{w}}_{\text{next,prev}}^{\text{u}} - \tilde{\boldsymbol{w}}_a^{\text{u}} \end{matrix} \right\|_2^2. \tag{31}$$

Here $w_{a,\text{next}}^{\text{u}}$ denotes the next agent to optimise and again the sample step k, and distributed MPC iteration l, are dropped for compactness.

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 total cost function for agent a is given by (10). This can be put into quadratic form using simple matrix manipulation, where the optimisation vector is $\Delta\tilde{\boldsymbol{u}}_{opt}(k) = [\Delta\tilde{\boldsymbol{u}}^{\text{T}}(k)\ \Delta\tilde{\boldsymbol{w}}_{\text{in}}^{\text{T}}(k)]^{\text{T}}$.

The HVDC link ranges are $-2 \leq P_i^{\text{DC}}(k) \leq 2$ 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 $a = 1,\ldots,4$. These constraints are applied over the full prediction horizon.

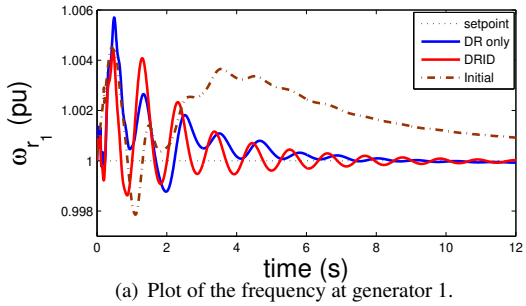
In a centralised MPC case, the optimal values calculated for $P_1^{\text{DC}}(k)$ and $P_2^{\text{DC}}(k)$ would be applied to the system. The 4 agents in the distributed MPC system calculate slightly different values for the HVDC powers to each other, as these powers only have to match to a degree, determined by the distributed MPC parameters c and ϵ.

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 subsystems 2 and 3 to subsystems 1 and 4, and so it is assumed their agents insist on having the final say on what power is transferred).

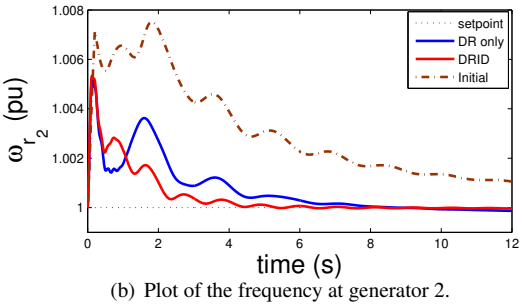
5.2.2. PSO optimisation of the distributed MPC weights

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.

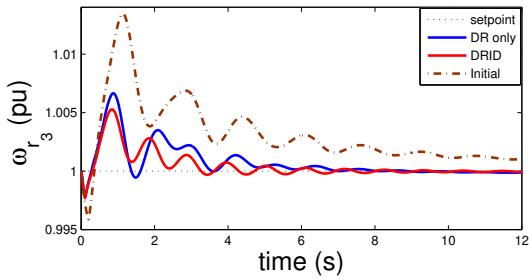
As previously stated, the weights determine the relative importance of the goals of the distributed MPC system. Therefore one weight is set equal to 1 and the rest of the weights are then optimised relative to this weight using PSO. The weight of agent 1, Q_1 is set equal to 1 and the vector of PSO weights is then $\boldsymbol{\Gamma} = [Q_2\ Q_3\ Q_4\ c\ \epsilon]^{\text{T}}$. 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



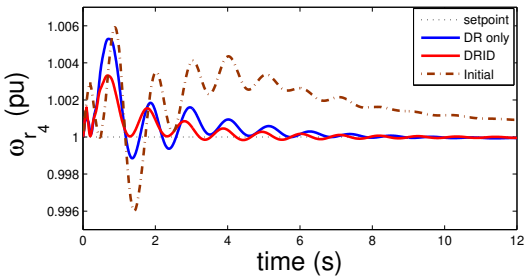
(a) Plot of the frequency at generator 1.



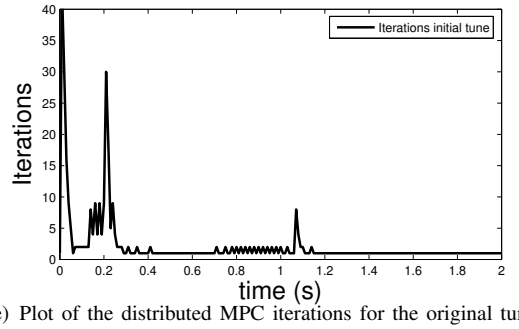
(b) Plot of the frequency at generator 2.



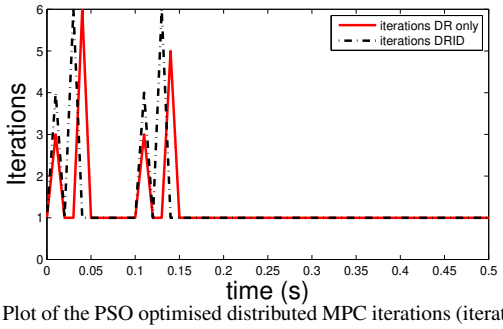
(c) Plot of the frequency at generator 3.



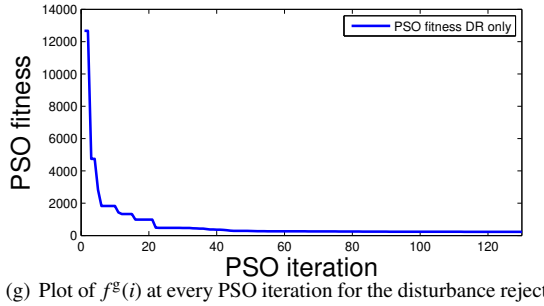
(d) Plot of the frequency at generator 4.



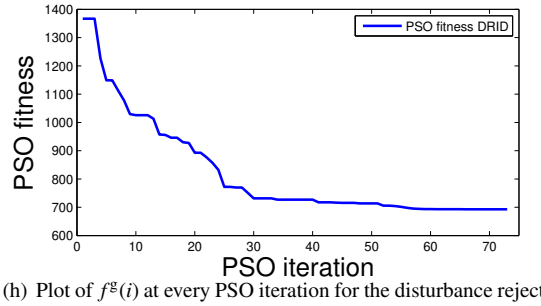
(e) Plot of the distributed MPC iterations for the original tuning (iterations stay at 1 after 2 seconds).



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

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 $a = 2, \ldots, 4$, $0.01 \leq c \leq 5$, 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. Manually tuning the distributed MPC, the best per-

formance 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.

5.2.3. Results

Simulations are run on a 2211.412 MHz Quad-Core AMD Opteron™ 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 (25) and (27), 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 are 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 (DRID) gave $[Q_1\ Q_2\ Q_3\ Q_4\ c\ \epsilon]^{\mathrm{T}}$ = [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(g) and 6(h) for the DR only and DRID cases, respectively. The other plots in Fig. 6 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 amount 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 time the multiple HVDC link simulation took to run. On average simulation runs of these simulations took between 10 and 20 minutes at a time 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 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.

5.3. 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 in this paper, overall times for PSO optimisation could also potentially be reduced

by running simulations for each of the particles in parallel on separate computers.

6. 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 the paper. 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.

Acknowledgements

This work was funded by the Irish Research Council for Science, Engineering and Technology (IRCSET) and supported the VENI project "Intelligent multi-agent control for flexible coordination of transport hubs" (project 11210) of the Dutch Technology Foundation STW.

References

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

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

Erikkson, R., 2008. Security-Centered coordinated control in AC/DC transmission systems. Licentiate thesis, Royal Institute of Technology, School of Electrical Engineering, Electric Power Systems, Stockholm, Sweden.

Fabijanski, P., Lagoda, R., 2008. On-line PID controller tuning using genetic algorithm and DSP PC board. In: Proceedings of the 13th Power Electronics and Motion Control Conference. Poznań, Poland, pp. 2087–2090.

Gaing, Z.-L., 2004. A particle swarm optimization approach for optimum design of PID controller in AVR system. IEEE Transactions on Energy Conversion 19 (2), 384–391.

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

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

Kennedy, J., Eberhart, R. C., 1995. Particle Swarm Optimization. In: Proceedings of the IEEE International Conference on Neural Networks. University of Western Australia, Perth, Australia, pp. 1942–1948.

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

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

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

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

Lin, G., Liu, G., 2010. Tuning PID controller using adaptive genetic algorithms. In: Proceedings of the 5th International Conference on Computer Science and Education (ICCSE). Hefei, China, pp. 519–523.

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

Liu, J., Muñoz de la Peña, D., Christofides, P., 2009. Distributed model predictive control of nonlinear process systems. AIChE Journal 55, 1171–1184.

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

Mc Namara, P., Negenborn, R. R., De Schutter, B., Lightbody, G., 2011. Coordination of a multiple link HVDC system using local communications based distributed model predictive control. In: Proceedings of the 18th IFAC World Congress. Milan, Italy, pp. 1558–1563.

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

Pai, M. A., Padiyar, K., Radhakrishna, C., 1981. 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.

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

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

Royo, C., 2001. Generalized unit commitment by the Radar Multiplier Method. Ph.D. thesis, Departament d'Estadística i Investigació Operativa, Universitat Politècnica de Catalunya, Barcelona, Spain.

Sanchez, G., Giovanini, L., Murillo, M., Limache, A., 2011. Distributed model predictive control based on dynamic games. In: Zheng, T. (Ed.), Advanced Model Predictive Control. Advanced Model Predictive Control. InTech Open, http://www.intechopen.com/articles/show/title/distributed-model-predictive-control-based-on-dynamic-games, Ch. 4, pp. 1–26.

Scattolini, R., 2009. Architectures for distributed and hierarchical model predictive control - A review. Journal of Process Control 19 (5), 723–731.

Suzuki, R., Kawai, F., Ito, H., Nakazawa, C., Fukuyama, Y., Aiyoshi, E., 2007. Automatic tuning of model predictive control using particle swarm optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium. Honolulu, Hawaii, USA, pp. 221–226.

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

Trelea, I. C., 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85, 317–325.

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

Wang, L., 2009. Model Predictive Control System Design and Implementation Using Matlab. Springer, London.

Appendix A. Power system parameters used in the multiple link HVDC simulation

$S_{\text{base}} = 100 \times 10^6$ VA, $U_{\text{base}} = 100 \times 10^3$ V, $f_{\text{base}} = 50$ Hz, $w_0 = 2\pi f_{\text{base}}$ 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		

Appendix B. PSO Toolbox parameters

Parameters used for the PSO Toolbox are given as follows:

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