University of Strathclyde

Department of Electronic and Electrical Engineering

Learning to Trade Power

by

Richard W. Lincoln

A thesis presented in fulfilment of the
requirements for the degree of

Doctor of Philosophy

2010

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

The copyright of this thesis belongs to the author under the terms of the United Kingdom Copyright Acts as qualified by University of Strathclyde Regulation 3.50. Due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

Signed: Date: August 25, 2010

Acknowledgements

I wish to thank Professor Jim McDonald for giving me the opportunity to study at The Institute for Energy and Environment and for giving me the freedom to pursue my own research interests. I also wish to thank my supervisors, Professor Graeme Burt and Dr Stuart Galloway, for their guidance and scholarship. I wish to offer very special thanks to my parents, my big brother and my little sister for all of their support throughout my PhD.

This thesis makes extensive use of open source software projects developed by researchers from other institutions. I wish to thank Dr Ray Zimmerman from Cornell University for his work on optimal power flow, researchers from the Dalle Molle Institute for Artificial Intelligence (IDSIA) and the Technical University of Munich for their work on reinforcement learning algorithms and artificial neural networks and Charles Gieseler from Iowa State University for his implementation of the Roth-Erev reinforcement learning method.

This research was funded by the United Kingdom Engineering and Physical Sciences Research Council through the Supergen Highly Distributed Power Systems consortium under grant GR/T28836/01.

Abstract

In Electrical Power Engineering, learning algorithms can be used to model the strategies of electricity market participants. The objective of this work is to establish if *policy gradient* reinforcement learning methods can provide superior participant models than previously applied *value function based* methods.

Supply of electricity involves technology, money, people, natural resources and the environment. All of these aspects are changing and electricity market designs must be suitably researched to ensure that they are fit for purpose. In this thesis electricity markets are modelled as non-linear constrained optimisation problems that are solved with a primal-dual interior point method. Policy gradient reinforcement learning algorithms are used to adjust the parameters of multi-layer feed-forward neural networks that approximate each market participant's policy for selecting power quantities and prices that are offered in a simulated marketplace.

Traditional reinforcement learning methods that learn a value function have been previously applied in simulated electricity trade, but are largely restricted to discrete representations of a market environment. Policy gradient methods have been proven to offer convergence guarantees in continuous environments, such as in robotic control applications, and avoid many of the problems that mar value function based methods.

Contents

List of Figures

List of Tables

Chapter 4

Modelling Power Trade

This chapter defines the model used in chapters 5 and 6 to simulate electric power trade and to compare learning algorithms. The first section describes how optimal power flow solutions are used to clear offers submitted to a simulated power exchange auction. The second section defines how market participants are modelled as agents that use the reinforcement learning algorithms to adjust their bidding behaviour. It explains the modular structure of a multi-agent system that coordinates interactions between the auction model and market participants.

4.1 Electricity Market Model

A power exchange auction market, based on SmartMarket by Zimmerman (2010, p.92), is used in this thesis to provide a trading environment for comparing reinforcement learning algorithms. In each trading period the auction accepts offers to sell blocks of power from participating agents[1]. A clearing process begins by withholding offers above the price cap, along with those specifying non-positive quantities. Valid offers for each generator are sorted into non-decreasing order with respect to price and converted into corresponding generator capacities and piecewise linear cost functions (See Section 4.1.1 below). The newly configured units form an optimal power flow problem, the solution to which provides generator set-points and nodal marginal prices that are used to determine the proportion of each offer block that is cleared and the associated clearing price. The cleared offers determine each agent's revenue and hence the profit that is used as a reward signal.

A nodal marginal pricing scheme is used in which the price of each offer is

[1]A double-sided auction, in which bids to buy blocks of power may be submitted by agents associated with dispatchable loads, has also been implemented, but this feature is not used.

cleared at the value of the Lagrangian multiplier on the power balance constraint for the bus at which the offer's generator is connected. An alternative a discriminatory pricing scheme may be used in which offers are cleared at the price at which they were submitted (pay-as-bid). The advanced auction types from MATPOWER that scale nodal marginal prices are not used.

4.1.1 Optimal Power Flow

Bespoke implementations of the optimal power flow formulations from MATPOWER are used in the auction clearing process. Both the DC and AC formulations are used in this thesis.

The trade-offs between DC and AC formulations have been examined by Overbye, Cheng, and Sun (2004). DC models were found suitable for most nodal marginal price calculations and are considerably less computationally expensive. The AC optimal power flow formulation is used in this thesis to examine the exploitation of voltage constraints, which are not part of the DC formulation.

As in MATPOWER, generator active power, and optionally reactive power, output costs may be defined by convex n-segment piecewise linear cost functions

$$c^{(i)}(p) = m_i p + b_i \tag{4.1}$$

where p is the generator set-point for $p_i \leq p \leq p_{i+1}$ with $i = 1, 2, \ldots n$, m_i is the variable cost for segment i in \$/MWh where $m_{i+1} \geq m_i$ and $p_{i+1} > p_i$, and b_i is the y-intercept in \$ for segment i. Offers submitted to the market are converted into a piecewise linear cost function for the associated generator. Since these cost functions are non-differentiable, the constrained cost variable approach from H. Wang, Murillo-Sanchez, Zimmerman, and Thomas (2007) is used to make the optimisation problem smooth. For each generator j a helper cost variable y_j is added to the vector of optimisation variables. Figure 2.4 illustrates how the additional inequality constraints

$$y_j \geq m_{j,i}(p - p_i) + c_i, \quad i = 1 \ldots n \tag{4.2}$$

ensure that y_j lies on or above $c^{(i)}(p)$ (Zimmerman, 2010, Figure5-3). The objective function for the optimal power flow formulation used in the auction clearing process is the minimisation of the sum of cost variables for all generators:

$$\min_{\theta, V_m, P_g, Q_g, y} \sum_{j=1}^{n_g} y_j \tag{4.3}$$

The extended optimal power flow formulations from MATPOWER with user-defined cost functions and generator P-Q capability curves are not used.

4.1.2 Unit De-commitment

The optimal power flow formulations constrain generator set-points between upper and lower power limits. The output of expensive generators can be reduced to the lower limit, but they can not be completely shutdown. The online status of generators could be added to the vector of optimisation variables, but being Boolean the problems would become mixed-integer non-linear programs which are typically very difficult to solve.

To compute a least cost commitment and dispatch the unit de-commitment algorithm from Zimmerman (2010, p.57) is used. The algorithm involves shutting down the most expensive units until the minimum generation capacity is less than the total load capacity and then solving repeated optimal power flow problems with candidate generating units, that are at their minimum active power limit, deactivated. The lowest cost solution is returned when no further improvement can be made and no candidate generators remain.

4.2 Multi-Agent System

Market participants are modelled with software agents from PyBrain that use reinforcement learning algorithms to adjust their behaviour (Schaul et al., 2010). Their interaction with the market is coordinated in multi-agent simulations, the structure of which is derived from PyBrain's single player design.

This section describes discrete and continuous market environments, agent tasks and modules that are used for policy function approximation and storing state-action values or action propensities. The process by which each agent's policy is updated by a learning algorithm is explained and the sequence of interactions between multiple agents and the market is described and illustrated.

4.2.1 Market Environment

Each agent has a portfolio of n_g generators associated their environment. Figure ?? illustrates the association and how the environment references an instance of the auction market for offer submission. Each environment is responsible for (i) returning a vector representation of its current state and (ii) accepting an action vector which transforms the environment into a new state. To facilitate testing of value function based and policy gradient learning methods, both discrete and continuous representations of an electric power trading environment are defined.

Discrete Market Environment

For agents operating learning methods that make use of look-up tables an environment with n_s discrete states and n_a discrete action possibilities is defined. The environment produces a state s, where $s \in \mathbb{Z}^+$ and $0 \leq s < n_s$, at each simulation step and accepts an action a, where $a \in \mathbb{Z}^+$ and $0 \leq a < n_a$.

To keep the size of the state space reasonable, the state is derived only from the total system demand $d = \sum P_d$. Each simulation episode of n_t steps has a demand profile vector u of length n_t, where $0 \leq u_i \leq 1$. The load at each bus $P_{dt} = u_t P_{d0}$ in simulation period t, where P_{d0} is the initial demand vector. The state size $d_s = d(\max u - \min u)/n_s$ and the state space vector is $\mathcal{S} = d_s i$ for $i = 1 \dots n_s$. At simulation step t, the state returned by the environment $s_t = i$ if $\mathcal{S}_i \leq P_{dt} \leq \mathcal{S}_{i+1}$ for $i = 0 \dots n_s$. Informally, the state space is n_s states between the minimum and maximum demand and the current state for the environment is the index of the state to which the current demand relates.

The action space for a discrete environment is defined by a vector m, where $0 \leq m_i \leq 100$, of percentage markups on marginal cost with length n_m, a vector w, where $0 \leq w_i \leq 100$, of percentage capacity withholds with length n_w and the number of offers n_o, where $n_o \in \mathbb{Z}^+$, to be submitted for each generator associated with the environment.

A $n_a \times 2n_g n_o$ matrix that contains all permutations of markup and withhold for each offer that is to be submitted for each generator is computed. For example, Table 4.1 shows all possible actions when markups are restricted to 0, 10% or 20% and 0% of capacity may be withheld from two generators with one offer submitted for each. Each row corresponds to an action and the column values specify the percentage price markup and the percentage of capacity to be withheld for each of the $n_g n_o$ offers. The size of the permutation matrix grows rapidly as n_o, n_g, n_m and n_w increase.

Continuous Market Environment

A continuous market environment that outputs a state vector s, where $s_i \in \mathbb{R}$, and accepts an action vector a, where $a_i \in \mathbb{R}$, is defined for agents operating policy gradient methods. Scalar variables m_{max} and w_{max} define the maximum allowable percentage markup on marginal cost and the maximum allowable percentage of

a	m_1	m_2	w_1	w_2
0	0	0	0	0
1	0	10	0	0
2	0	20	0	0
3	10	0	0	0
4	10	10	0	0
5	10	20	0	0
6	20	0	0	0
7	20	10	0	0
8	20	20	0	0

Table 4.1: Example discrete action domain.

capacity that can be withheld, respectively. Again, n_o defines the number of offers to be submitted for each generator associated with the environment.

The state vector may consist of any data from the power system or market model. For example: bus voltages, branch power flows, generator limit Lagrangian multipliers etc. Each element of the vector provides one input to the neural network used for policy function approximation.

The action vector a has length $2n_gn_o$. Element a_i, where $0 \leq a_i \leq m_{max}$, corresponds to the price markup and a_{i+1}, where $0 \leq a_{i+1} \leq w_{max}$, to the withhold of capacity for the $(i/2)^{th}$ offer, where $i = 0, 2, 4, \ldots, 2n_gn_o$.

Not having to discretize the state space and compute a matrix of action permutations greatly simplifies the implementation of a continuous environment and increases in n_g and n_o only impact the number of output nodes in the policy function approximator.

4.2.2 Agent Task

To allow alternative goals, such a profit maximisation or the meeting some target level for plant utilisation, to be associated with a single type of environment, an agent does not interact directly with its environment, but is paired with a particular *task*. A task defines the reward returned to the agent and thus defines the agent's purpose.

For all simulations in this thesis the goal of each agent is to maximise financial profit. Rewards are defined as the sum of earnings from the previous period t as determined by the difference between revenue from cleared offers and marginal cost at the total cleared quantity. As explained in Section 3.4.1, utilising some measure of risk adjusted return might be of interest in the context of simulated electricity trade and this would simply involve the definition of a new task and would not require any modification of the environment.

Agents with policy-gradient learning methods approximate their policy functions using artificial neural networks that are presented with input vector v of length n_s where $v_i \in \mathbb{R}$. To condition the environment state before input to the connectionist system, where possible, a vector s_{min} of minimum sensor values and a vector s_{max} of maxmimum sensor values is defined. These are used to calculated a normalised state vector

$$v = 2\left(\frac{s - s_{min}}{s_{max} - s_{min}}\right) - 1 \tag{4.4}$$

where $-1 \leq v_i \leq 1$.

The output from the policy function approximator y is denormalized using vectors of minimum and maximum action limits, a_{min} and a_{max} respectively, to give an action vector

$$a = \left(\frac{y+1}{2}\right)(a_{max} - a_{min}) + a_{min} \tag{4.5}$$

with valid values for price markups and capacity withholding.

4.2.3 Market Participant Agent

Each agent is defined as an entity capable of producing an action a based on previous observations of its environment s. The UML class diagram in Figure **??** illustrates how each agent in PyBrain is associated with a *module*, a *learner* (variant Roth-Erev in the case of the diagram), a *dataset* and an *explorer*.

The module is used to determine the agent's policy for action selection and returns an action vector a_m when activated with observation s. When using value function based methods the module is a $n_s \times n_a$ table:

$$\begin{array}{c c} & \begin{array}{cccc} a_0 & a_1 & & a_{n_a} \end{array} \\ \begin{array}{c} s_0 \\ s_1 \\ \\ s_{n_s} \end{array} & \begin{bmatrix} v_{0,0} & v_{0,1} & \cdots & v_{0,m} \\ v_{1,0} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ v_{n,0} & \cdots & \cdots & v_{n_s,n_a} \end{bmatrix} \end{array} \tag{4.6}$$

where each element $v_{i,j}$ is the value associated with selecting action j in state i. When using a policy gradient method, the module is a multi-layer feed-forward artificial neural network that outputs a vector a when presented with observation s.

The learner can be any reinforcement learning algorithm that modifies the values/parameters of the module to increase expected future reward. The dataset stores state-action-reward triples for each interaction between the agent and its environment. The stored history is used by value-function learners when computing updates to the table values. Policy gradient learners search directly in the space of the policy network parameters.

Each learner has an association with an explorer that returns an explorative action a_e when activated with the current state s and action a_m from the module. Softmax and ϵ-greedy explorers are implemented for discrete action spaces. Policy gradient methods use a module that adds Gaussian noise to the output of the policy function approximation module. The explorer has a parameter σ that relates to the standard deviation of the normal distribution. The actual standard deviation

$$\sigma_e = \begin{cases} \ln(\sigma + 1) + 1 & \text{if } \sigma \geq 0 \\ \exp(\sigma) & \text{if } \sigma < 0 \end{cases} \tag{4.7}$$

to allow for negative σ values.

4.2.4 Simulation Event Sequence

Each simulation consists of one or more task-agent pairs. Figure **??** shows the class associations for a simulation experiment. At the beginning of each simulation step (trading period) t the market is initialised and all existing offers are removed. Figure **??** is a UML sequence diagram that illustrates the process of choosing and performing an action. For each task-agent tuple an observation s_t is retrieved from the task and integrated into the agent. When an action is requested from the agent its module is activated with s_t and the action a_e is returned. Action a_e is performed on the environment associated with the agent's task.

When all actions have been performed the offers are cleared by the market using the solution to a newly formed optimal power flow problem. Figure **??** illustrates the reward precess that follows. The cleared offers associated with the generators in the task's environment are retrieved from the market and the reward r_t in \$ is computed from the difference between revenue and marginal cost at the total cleared quantity. For each generator in the agent's portfolio that was previously online and is not dispatched, a shutdown cost C_{down} is subtracted from the reward. The reward r_t is given to the associated agent and the value is stored, along with the previous state s_t and selected action a_e, under a new sample is the dataset.

The learning process is illustrated by the UML sequence diagram in Figure **??**. Each agent learns from its actions using r_t, at which point the values or parameters of the module associated with the agent are updated according to the output of the learner's algorithm. Each agent is then reset and the history of states, actions and rewards is cleared.

The combination of action, reward and learning processes for each agent constitutes one step of the simulation and they are repeated until a specified number of steps are complete.

4.3 Summary

The power exchange auction market model defined in this chapter provides a layer of abstraction over the underlying optimal power flow problem and presents agents with a simple interface for selling power. The modular nature of the simulation framework described allows the type of learning algorithm, policy function approximator, exploration technique or task to be easily changed. The framework can simulate competitive electric power trade using any conventional bus-branch power system model with little configuration, but provides the ability to adjust all of the main aspects of a simulation. The modular framework and its support for easy configuration is intended to allow transparent comparison of learning methods in the domain of electricity trade under a number of different scenarios.

Chapter 5

Nash Equilibrium Analysis

This chapter presents a simulation that examines the convergence to a Nash equilibrium of agents competing to sell electricity. Value function based and policy gradient reinforcement learning algorithms are compared in their convergence to an optimal policy using a six bus electric power system model.

5.1 Introduction

This thesis presents the first case of policy gradient reinforcement learning methods being applied to electricity trading problems. As a first step it is necessary to confirm that when using these methods, a system of multiple agents will converge to the same Nash equilibrium[1] that a traditional closed-form simulation would produce.

This is the same approach used by Krause et al. (2006) before performing the study of congestion management techniques that is reviewed in Section 3.2.2. Nash equilibria can be difficult to determine in complex systems so the experiment presented here utilises a model simple enough that it can be determined through exhaustive search.

By observing the actions taken and the reward received by each agent over the initial simulation periods it is possible to compare the speed and consistency with which different algorithms converge to an optimal policy. In the following sections the objectives of the simulations are explicitly defined, the setup of the simulations is explained and simulation results, with discussion and critical analysis, are provided.

[1]Informally, a Nash equlibrium is a point in a non-cooperative game at which no player is motivated to deviate from its strategy, as it would result in lower gain (Nash, 1950, 1951).

5.2 Aims and Objectives

Some elements of the simulations reported in this chapter are similar to those presented by Krause et al. (2006). One initial aim of this work is to reproduce their findings as a means of validating the approach. The additional objectives are to show:

- That policy gradient methods converge to the same Nash equilibrium as value function based methods and tradtional closed-form simulations,
- Differences in the characteristics of policy gradient and value function based methods by examining the nature of their convergence to an optimal policy.

Meeting these objectives aims to provide a basis for using policy gradient methods in more complex simulations, to show that they can learn basic policies and to provide guidance for algorithm parameter selection.

5.3 Method of Simulation

Learning methods are compared in this chapter by repeating the same simulation with different algorithms used by the agents. An alternative might be to use a combination of methods in the same simulation, but the approach used here is intended to be an extension of the work by Krause et al. (2006).

Each simulation uses a six bus electric power system model adapted from Wood and Wollenberg (1996, pp. 104, 112, 119, 123-124, 549). The model provides a simple environment for electricity trade with a small number of generators and branch flow constraints that slightly increase the complexity of the Nash equilibria. The buses are connected by eleven transmission lines at 230kV. The model contains three generating units with a total capacity of 440MW and loads at three locations, each of 70MW. The connectivity of the branches and the locations of the generators and loads is shown in Figure **??**. Data for the power system model was taken from a case provided with MATPOWER and is listed in Appendix B.1.

Two sets of quadratic generator operating cost functions, of the form $c(p_i) = ap_i^2 + bp_i + c$ where p_i is the out put of generator i, are defined in order to create two different equilibria for investigation. The coefficients a, b and c for cost configuration 1 are listed in Table 5.1. This cost configuration defines two low cost generators that can not offer a price greater than the marginal cost of the most expensive generator when they apply the maximum possible markup. The set of coefficients for cost configuration 2 is listed in Table 5.2. This configuration

Gen	C_{down}	a	b	c
1	0	0.0	4.0	200.0
2	0	0.0	3.0	200.0
3	0	0.0	6.0	200.0

Table 5.1: Generator cost configuration 1.

Gen	C_{down}	a	b	c
1	0	0.0	5.1	200.0
2	0	0.0	4.5	200.0
3	0	0.0	6.0	200.0

Table 5.2: Generator cost configuration 2.

narrows the cost differences such that offer prices may overlap and may exceed the marginal cost of the most expensive generator.

As in Krause et al. (2006), no load profile is defined for the simulation. The system load is assumed to be peak in all periods and only one state is defined for methods using look-up tables. Each simulation step is assumed to be one hour in length.

For all generators $P^{min} = 0$ so as to simplify the equilbria and avoid the need to use the unit de-commitment algorithm. The maximum capacity for the most expensive generator $P_3^{max} = 220$MW such that it may almost supply all of the load if dispatched. This generator is associated with a passive agent that always offers full capacity at marginal cost. For the less expensive generators $P_1^{max} = P_2^{max} = 110$MW. These two generators are each associated with an active learning agent whose activity in the market is restricted to one offer of maximum capacity in each period, at a price representing a markup of between 0 and 30% on marginal cost. Methods restricted to discrete actions may markup in steps of 10%, giving possible markup actions of 0, 10%, 20% and 30%. No capacity withholding is implemented. Discriminatory pricing (pay-as-bid) is used in order to provide a clearer reward signal to agents with low cost generators.

The algorithms which are compared are Q-learning, ENAC, REINFORCE and the modified Roth-Erev technique (See Section 2.4). Default algorithm parameter values from PyBrain are used and no attempt to study parameter sensitivity or variations in function approximator design is made.

For the Q-learning algorithm $\alpha = 0.3$, $\gamma = 0.99$ and ϵ-greedy action selection is used with $\epsilon = 0.9$ and $d = 0.98$. For the Roth-Erev technique $\epsilon = 0.55$, $\phi = 0.3$ and Boltzmann action selection is used with $\tau = 100$ and $d = 0.99$.

		G_1							
		0.0%		10.0%		20.0%		30.0%	
		r_1	r_2	r_1	r_2	r_1	r_2	r_1	r_2
G_2	0.0%	0.0	0.0	40.0	0.0	80.0	0.0	120.0	0.0
	10.0%	0.0	33.0	40.0	33.0	80.0	33.0	120.0	33.0
	20.0%	0.0	66.0	40.0	66.0	80.0	66.0	120.0	66.0
	30.0%	0.0	99.0	40.0	99.0	80.0	99.0	120.0*	99.0*

Table 5.3: Agent rewards under cost configuration 1

		G_1							
		0.0%		10.0%		20.0%		30.0%	
		r_1	r_2	r_1	r_2	r_1	r_2	r_1	r_2
G_2	0.0%	0.0	0.0	51.0	0.0	0.0	0.0	0.0	0.0
	10.0%	0.0	49.5	51.0	49.5	0.0	49.5	0.0	49.5
	20.0%	0.0	92.2	51.0	99.0	0.0	99.0	0.0	99.0
	30.0%	0.0	126.8	54.8*	138.4*	0.0	148.5	0.0	148.5

Table 5.4: Agent rewards under cost configuration 2

Both REINFORCE and ENAC use a two-layer neural network with one linear input node, one linear output node, no bias nodes and with the connection weight initialised to zero. A two-step episode is defined for the policy gradient methods and five episodes are performed per learning step. The exploration paramter σ for these methods is initialised to zero and adjusted manually after each episode such that:

$$\sigma_t = d(\sigma_{t-1} - \sigma_n) + \sigma_n \tag{5.1}$$

where $d = 0.998$ is a decay parameter and $\sigma_n = -0.5$ specifies the value that is converged to asymtotically. In each simulation the learning rate $\gamma = 0.01$ for the policy gradient methods, apart from for ENAC under cost configuration 2 where $\gamma = 0.005$. Both active agents use the same parameter values in each simulation.

As in Krause et al. (2006), the point of Nash equilibrium is established by computing each agent's reward for all possible combinations of markup. The rewards for Agent 1 and Agent 2 under cost configuration 1 are given in Table 5.3. The Nash equilibrium points are marked with a *. The table shows that the optimal policy for each agent is to apply the maximum markup to each offer as their generators are always dispatched. The rewards under cost configuration 2 are given in Table 5.4. This table shows that the optimal point occurs when Agent 2 applies its maximum markup and Agent 1 offers a price just below the marginal cost of the passive agent's generator.

5.4 Simulation Results

Each action taken by an agent and the consequent reward is recorded for each simulation. Values are averaged over the ten simulation runs and standard deviations are calculated using the formula

$$SD = \sqrt{\frac{1}{N-1}\sum_{i=0}^{N}(x_i - \bar{x})^2} \tag{5.2}$$

where x_i is the action or reward value in simulation i of N simulation runs and $\bar{x}$ is the mean of the values.

Figure **??** shows the average markup on marginal cost and the standard deviation over the ten simulation runs for Agent 1 under price configuration 1 using the four learning methods. The second y-axis in each plot realtes to the exploration parameter for each method. Figure **??** shows the same quantities for Agent 2. Plots of reward are not given as generator prices and the market are configured such that an agent's reward is directly proportional to its action. The plots are vertically aligned and have equal x-axis limits to assist algorithm comparison.

Figures **??** and **??** plot the average markup and reward over ten simulation runs for Agent 1 and Agent 2, respectively, under price configuration 2 for the variant Roth-Erev, Q-learning learning methods. The plots for REINFORCE and ENAC in these figures are for actual values in one simulation run as the number of interactions and variation in values makes the results difficult to observe otherwise. Not all x-axis extents are equal in these two figures.

5.5 Discussion and Critical Analysis

Under cost configuration 1 the agents face a relatively simple control task and receive a clear reward signal that is directly proportional to their markup. The results show that all of the methods consistently converge to the Nash equilibrium point. The variant Roth-Erev method shows very little variation around the mean once converged due to the use of Boltmann exploration with a then low temperature parameter value. The constant variation around the mean that can be seen for Q-learning once it has converged is due to the use of ϵ-greedy action selection and can be removed if a Boltmann explorer is used.

Empirical studies have also shown that the speed of convergence is largely determined by the rate at which the exploration parameter value is reduced. However, the episodic nature of the policy gradient methods requires them to make several interactions per learning step and therefore a larger number of initial exploration steps are needed. Policy gradient methods have also been found to be highly sensitive to the choice of learning rate. High values cause large changes to policy parameters to be made at each step and may cause the algorithm to fail to converge, but low values cause the algorithm to learn very slowly.

Cost configuration 2 provides a more challenging control problem in which Agent 1 must learn to undercut the passive agent. The results show that the variant Roth-Erev and Q-learning methods both consistently learn their optimal policy and converge to the Nash equilibrium. However, there is space for Agent 1 to markup its offer by slightly more than 10% and still undercut the passive agent, but methods with discrete actions are not able to exploit this and do not receive the additional profit.

The results for the policy gradient methods under cost configuration 2 show that these methods learn to reduce their markup if their offer price starts to exceed that of the passive agent and the reward signal drops. However, a chattering effect below the Nash equilibrium point can be clearly seen for ENAC and the method does not learn to always undercut the other agent. These methods also require a much larger number of simulation steps and for the exploration parameter to be decayed more slowly if they are to produce this behaviour. This is due to the need for a lower learning rate that ensures fine policy adjustments can be made and for several interactions to be performed between each learning step.

5.6 Summary

By observing the state to which a multi-learning-agent system converges, it is possible to verify that algorithms produce the same Nash equilibrium that closed-form simulations provide. The results presented in this chapter closely correspond with those from Krause et al. (2006) for Q-learning and show equivalent behaviour for the variant Roth-Erev method. The simulations illustrate how challenging unsupervised learning in a continuous environment can be, even for simple problems. Tasks in which a large reward change can occur for a very small change in policy prove difficult for policy gradient methods to learn and require low learning rates and lengthy periods of exploration. The operation of policy gradient methods with noisy, multi-dimensional state data is not examined in this chapter and deserves investigation.

Chapter 6

System Constraint Exploitation

This chapter explores learning agents exploitation of constraints in electric power system models. Value function based and policy gradient reinforcement learning methods are compared using a dynamic 24-bus power system model from the IEEE Reliability Test System.

6.1 Introduction

Having examined the basic learning characterisitics of four algorithms in Chapter 5, this experiment extends the approach to examine their operation in a complex dynamic environment. It explores the ability of policy gradient methods to operate with multi-dimensional, continuous state and action spaces in the context of *learning to trade power*.

A well established electric power system model from the IEEE Reliability Test System (Application of Probability Methods Subcommittee, 1979) provides a realistic environment in which agents compete with their portfolios of generating plant to supply dynamic loads. System constraints change as agents adjust their behaviour and the loads follow a daily profile that varies over the course of a simulated year. By observing profits at different times of day, the ability of methods to successfully observe and exploit constraints is examined.

6.2 Aims and Objectives

This experiment aims to compare policy gradient and traditional learning methods in a dynamic electricity trading environment. Specifically, the objectives are to determine:

- If the policy gradient methods can achieve greater profitability under dynamic system constraints.
- The value of using an AC optimal power flow formulation in agent based electricity market simulation.

Meeting these objectives would demonstrate some of the value of using policy gradient methods in electricity market participant modelling and determine if they warrant further research in this domain.

6.3 Method of Simulation

In this experiment learning methods are compared by repeating simulations of competitive electricity trade with different algorithms used by the competing agents. Some simplification of the state and action representations for value function based methods is required, but the portfolios of generation and the load profiles are the same for each algorithm test.

The IEEE Reliability Test System (RTS) provides the power system model and load profiles used in each simulation. The model has 24 bus locations that are connected by 32 transmission lines, 4 transformers and 2 underground cables. The transformers tie a 230kV area to an area at 138kV. The original model has 32 generators of 9 different types with a total capacity of 3.45GW. To reduce the size of the discrete action domain, five 12MW and four 20MW generators are removed. This is deemed reasonable as their combined capacity is only 4.1% of the original total generation capacity and the remaining capacity is more than sufficient to meet demand. To further reduce action space sizes all generators of the same type at the same bus are aggregated into one generating unit. The model has loads at 17 locations and the total demand at system peak is 2.85GW.

Generator costs are quadratic functions of output, defined by the parameters in Table 6.1. Figure **??** shows the cost functions for each of the seven types of generator and illustrates their categorisation by fuel type. Generator cost function coefficients were taken from a website hosted by Georgia Tech Power Systems Control and Automation Laboratory[1] that assumes Coal costs of 1.5 \$/MBtu[2], Oil costs of 5.5 \$/MBtu and Uranium costs of 0.46 \$/MBtu. Data for the modified model is provided in Appendix B.2 and the connectivity of branches and the location of generators and loads is illustrated in Figure **??**.

[1]http://pscal.ece.gatech.edu/testsys/

[2]1 Btu $\approx$ 1055 Joules

Code	C_{down}	a	b	c	Type
U50	0	0.0	0.001	0.001	Hydro
U76	0	0.01414	16.0811	212.308	Coal
U100	0	0.05267	43.6615	781.521	Oil
U155	0	0.00834	12.3883	382.239	Coal
U197	0	0.00717	48.5804	832.758	Oil
U350	0	0.00490	11.8495	665.109	Coal
U400	0	0.00021	4.4231	395.375	Nuclear

Table 6.1: Cost parameters IEEE RTS generator types.

The generating stock is divided into 4 portfolios (See Table 6.2) that are each endowed to a learning agent. Portfolios were chosen such that each agent has: a mix of base load and peaking plant, approximately the same total generation capacity and generators in different areas of the network. The generator labels in Figure **??** specify the associated agent. The synchronous condenser is associated with a passive agent that always offers 0 MW at 0 \$/MWh (the unit can be dispatched to provide or absorb reactive power).

Markups on marginal cost are restricted a maximum of 30% and discrete markups of 0 or 30% are defined for value function based methods. Upto 30% of the total capacity of each generator can be withheld and discrete withholds of 0 or 30% are defined. Agent 3 has the largest discrete action space with XX possible actions to be explored in each state.

The state for all algorithm tests contains a forecast of the total system demand for the period that capacity is being offered for. The system demand follows an hourly profile that is adjusted according to the day of the week and the time of year. The profiles are taken from the RTS and are shown in Figure **??**. For tests of value function based methods or the Roth-Erev learning algorithm, the continuous state is divided into XX discrete states between minimum and maximum total system load. The state vector for agents using policy gradient methods additionally contains the voltage magnitude at each bus. Branch flows are not included in the state vector as the flow limits in the RTS are high and none are reached when the system is at peak demand. Generator capacity limits are binding in most states of the RTS, but the output of other generators is deemed to be hidden from the agents.

The nodal marginal pricing scheme is used in which cleared offer prices are determined by the Lagrangian multiplier on the power balance constraint for the bus at which the generator associated with the offer is connected.

Typical parameter values are used for each of the algorithms. Learning rates

Agent	U50 Hydro	U76 Coal	U100 Oil	U155 Coal	U197 Oil	U350 Coal	U400 Nuclear	Total (MW)
1		2×		1×			1×	707
2		2×		1×			1×	707
3	6×				3×			891
4			3×	2×		1×		960

Table 6.2: Agent portfolios.

are set low and the exploration parameters are decayed slowly due to the length and complexity of each simulation. For Q-learning $\alpha = 0.3$, $\gamma = 0.99$ and ϵ-greedy action selection is used with $\epsilon = 0.9$ and $d = 0.98$. For Roth-Erev learning $\epsilon = 0.55$, $\phi = 0.3$ and Boltzmann action selection is used with $\tau = 100$ and $d = 0.99$.

6.4 Simulation Results

6.5 Discussion and Critical Analysis

6.6 Summary

Chapter 7

Conclusions and Further Work

This chapter summarizes the conclusions that can be drawn from the results that are presented in this thesis and presents ideas for further development of the contributions that have been made.

7.1 Summary and Conclusions

This thesis has introduced the use of policy gradient reinforcement learning algorithms for modelling electricity market participant strategies. Over the last two decades markets have become an essential component in the electricity supply industries of many large countries. They will play an important role in the future as the world population grows and finite primary energy fuel resources become increasingly scarce. Market designs for electricity are unique amongst commodity markets and new architectures are expensive and risky to implement.

Computational simulation is a well established technique for evaluating market design concepts and agent-based simulation is an approach that allows large complex systems to be modelled. There are many examples of learning algorithms being used to model electricity market participants in the literature, but policy gradient methods have not been previously applied. They are a method that can use function approximation techniques to operate in continuous state and action spaces and have been used successfully in network routing and robot control applications.

To examine the properties of policy gradient methods and compare their performance with previously applied value function based methods a modular simulation framework has been defined and implemented. The framework uses a power exchange auction market model with nodal marginal pricing to provide an environment in which agents learn to trade electricity competitively.

The framework is first used in a simulation that compares the convergence to Nash equilibria of four different learning algorithms. The simulation reproduced the findings of Krause et al. (2006) and presented similar results for policy gradient methods. Policy gradient methods were found to require a larger number of interactions before learning an optimal policy and for learning rate and exploration rate decay parameters to be low for the more complex equilibrium to be approached.

In a second simulation the same algorithms were compared in a complex dynamic electricity trading environment. A reference electric power system model for reliability analysis that experiences a variety of constraint conditions as load follows an annual profile was used. The algorithms were compared in their ability to observe and exploit systems constraints. Policy gradient methods were found to . . .

In conclusion, policy gradient methods are a valid alternative to previously applied methods that require discrete environment representations. They have been shown to develop similar policies as value function based methods in simple problems. It has been how even moderately complex electricity market simulations produce state and action spaces that are too large for value function based methods to explore. Policy gradient methods have been shown to produce consistent behaviour in increasingly complex dynamic trading problems. Further developement of this research could provide an opportunity for policy gradient methods to be used in descision support and automated energy trade applications.

7.2 Further Work

This final section describes some of the shortcommings of the simulations presented in this thesis and how the models could be further developed. It introduces some alternative learning algorithms that might also be used to simulate electricity market participant behaviour. Finally, it explains is how a model formulated using data from National Grid Ltd. could be used in practical simulations of the UK electricity market and describes some further possibilities for using AC optimal power flow in agent-based electric power market simulation.

7.2.1 Parameter Sensitivity and Delayed Reward

The simulations presented in this thesis use typical algorithm parameters that are either the default values from PyBrain or taken from the literature. No investigation of parameter sensitivity is performed. Alternative function approximation and back-propagation techniques for use with policy gradient methods also deserve investigation. Parameter sensitivity analysis is typically conducted by the algorithm developers using standard benchmark problems, such as mazes and pole balancing problems, that are familar to researchers in Artificial Intelligence and allow results to be compared. The shortage of published results and lack of standardised electricity trading models might limit the benefits of using this problem for general parameter sensitivity analysis.

The reward signals received by agents in all of the simulations presented in this thesis result directly from the agent's previous action. In reality, market settlement processes introduce delays to payments for electricity production. Time did not permit value function based methods with eligibility traces (See Section **??**) to be compared with policy gradient methods, but the ability to learn under delayed reward is a fundamental part of reinforcement learning and deserves investigation in this context.

7.2.2 Alternative Learning Algorithms

This thesis has concentrated on traditional value function based and two policy gradient reinforcement learning methods. However, there are other learning algorithms that have been published recently and might also be used in electric power trade simulations.

Riedmiller (2005) presented Neuro-Fitted Q-Iteration (NFQ) algorithms that attempt to overcome many of the problems experienced when implementing Q-learning methods with value function approximation using neural networks. They store all transition experiences and perform off-line updates using supervised learning techniques such as RProp (Riedmiller & Braun, 1993). The method has been shown to be robust against parameterization and to learn quickly in standard benchmark tests and in real-world applications (Kietzmann & Riedmiller, 2009).

The GQ(λ) algorithm by Maei and Sutton (2010) is another extension of Q-learning for operation in continuous environments. Convergence guarantees have been shown and the scaling properties suggest the method is suitable for large-scale reinforcement learning applications. A software implementation of GQ(λ) has been developed by the authors and made available as open source.

Four new Natural Actor-Critic algorithms have been presented by Bhatnagar, Sutton, Ghavamzadeh, and Lee (2009). Like ENAC (Peters & Schaal, 2008), they too use function approximation techniques and are suitable for large-scale applications of reinforcement learning. Three of the algorithms are extensions to ENAC, but are fully incremental: the gradient computation is never reset while the policy is updated at every simulation step. The authors state a need to assess the ultimate utility of these algorithms through application in real-world problems.

This thesis provides a framework that would allow implementations of these algorithms to be assessed and used to examine many aspects of electricity markets.

7.2.3 UK Transmission System

Some of the more ambitious agent-based electricity market simulations have used stylised models of national transmission systems (Rastegar, Guerci, & Cincotti, 2009; Weidlich & Veit, 2006). This work has often been motivated by recent or expected changes to the arrangements in the associated regions. In the UK, nine large power stations are due to be decommissioned by 2016 in accordance with EU Large Combustion Plant Directive (National Electricity Transmission System Operator, 2007). Coupled with obligations, made in the Climate Change Act 2008, to cut greenhouse gas emissions by 80% by 2050, coming years are likely to see major changes in the way the UK power system is operated. Examination of the situation could be enhanced by advanced participant behavioural models and accurate electric power system simulations such as those presented in this thesis.

Figure **??** illustrates a model of the UK transmission system that has been formulated from data provided by the National Electricity Transmission System Operator (2010). This model has been converted into a PSS/E version 30 raw file that is distributed with the code developed for this thesis (See Appendix A.11). It is currently too computationally expensive to be solved repeatedly in an agent-based simulation, but optimisation efforts might allow it to be used to study issues highly pertinent to the UK energy industry.

7.2.4 AC Optimal Power Flow

This thesis presents the first application of AC optimal power flow in electricity market simulation using reinforcement learning agents. AC optimal power flow formulations are more difficult to implement and more computationally expensive when solving than than their linearised DC counterparts. The additional time and effort required for their use does not always add sufficient value to simulations. However, the option to use AC formulations does provide certain opportunities for further work.

The inclusion of reactive power costs in the objective function of an AC optimal power flow problem means that parallel auctions for voltage support could be added to simulations. This could be open to agents associated with reactive compensation equipment such as that commonly needed for wind farm developments. Traditionally, reactive power markets have been largely academic, but as the UK makes greater use of on and off-shore wind power the topic could become of increasing interest.

Bus voltages are not all assumed to be 1 per-unit in AC optimal power flow problems, but are part of the vector of optimisation variables. Adjusting phase shift angles, θ_{ph}, can offer a degree of control over power flow directions. The control the transformer tap ratios, τ, and the phase shift angles by learning agents could be of particular interest in congestion management scheme evaluations.

7.2.5 Multi-Market Simulation

The global economy is a holistic system of systems and the analysis of markets independently must be of limited value. Recent agent-based electricity market studies have investigated the interaction between electricity, gas and emissions allowance markets (Kienzle, Krause, Egli, Geidl, & Andersson, 2007; J. Wang, Koritarov, & Kim, 2009).

The information on the UK gas network provided by the National Electricity Transmission System Operator (2010) is relatively limited compared to that on the electricity transmission system, but suitable models could be used in conjunction to study the the relationships between UK gas and electricity markets. As in Kienzle et al. (2007), actions in the gas market would constrain the generators options to sell power in subsequent electricity auctions. Add to this the option to trade in emissions allowance markets and the associated state and action spaces for agents would be very large and require the use of suitable learning methods.

Bibliography

Alam, M. S., Bala, B. K., Huo, A. M. Z., & Matin, M. A. (1991). A model for the quality of life as a function of electrical energy consumption. Energy, 16(4), 739-745.

Amerongen, R. van. (1989, May). A general-purpose version of the fast decoupled load flow. Power Systems, IEEE Transactions on, 4(2), 760-770.

Application of Probability Methods Subcommittee. (1979, November). IEEE reliability test system. Power Apparatus and Systems, IEEE Transactions on, PAS-98(6), 2047-2054.

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2003). The non-stochastic multiarmed bandit problem. SIAM Journal of Computing, 32(1), 48-77.

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In Proceedings of the Twelfth International Conference on Machine Learning (p. 30-37). Morgan Kaufmann.

Bellman, R. E. (1961). Adaptive control processes – A guided tour. Princeton, New Jersey, U.S.A.: Princeton University Press.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., & Lee, M. (2009). Natural actor-critic algorithms. Automatica, 45(11), 2471–2482.

Bishop, C. M. (1996). Neural networks for pattern recognition (1st ed.). Oxford University Press, USA. Paperback.

Bower, J., & Bunn, D. (2001, March). Experimental analysis of the efficiency of uniform-price versus discriminatory auctions in the england and wales electricity market. Journal of Economic Dynamics and Control, 25(3-4), 561-592.

Bower, J., Bunn, D. W., & Wattendrup, C. (2001). A model-based analysis of strategic consolidation in the german electricity industry. Energy Policy, 29(12), 987-1005.

Bunn, D., & Martoccia, M. (2005). Unilateral and collusive market power in the electricity pool of England and Wales. Energy Economics.

Bunn, D. W., & Oliveira, F. S. (2003). Evaluating individual market power in electricity markets via agent-based simulation. Annals of Operations Research, 57-77.

Carpentier, J. (1962, August). Contribution à l'étude du Dispatching Economique. Bulletin de la Society Francaise Electriciens, 3(8), 431-447.

Cole, S. (2010, February 4). MatDyn [Computer software manual]. Katholieke Universiteit Leuven.

Department of Energy and Climate Change. (2009). Digest of United Kingdom Energy Statistics 2009. In (chap. 5). National Statistics – Crown.

Erev, I., & Roth, A. E. (1998). Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. The American Economic Review, 88(4), 848-881.

Ernst, D., Minoia, A., & Ilic, M. (2004, June). Market dynamics driven by the decision-making of both power producers and transmission owners. In Power Engineering Society General Meeting, 2004. IEEE (p. 255-260).

Fausett, L. (Ed.). (1994). Fundamentals of neural networks: architectures, algorithms, and applications. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Gieseler, C. (2005). A Java reinforcement learning module for the Repast toolkit: Facilitating study and implementation with reinforcement learning in social science multi-agent simulations. Unpublished master's thesis, Department of Computer Science, Iowa State University.

Glimn, A. F., & Stagg, G. W. (1957, April). Automatic calculation of load flows. Power Apparatus and Systems, Part III. Transactions of the American Institute of Electrical Engineers, 76(3), 817-825.

Goldfarb, D., & Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. Mathematical Programming, 27, 1-33.

Gordon, G. (1995). Stable function approximation in dynamic programming. In Proceedings of the Twelfth International Conference on Machine Learning (p. 261-268). Morgan Kaufmann.

Grainger, J., & Stevenson, W. (1994). Power system analysis. New York: McGraw-Hill.

Guo, M., Liu, Y., & Malec, J. (2004, October). A new Q-learning algorithm based on the metropolis criterion. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 34(5), 2140-2143.

ICF Consulting. (2003, August). The economic cost of the blackout: An issue paper on the northeastern blackout. (Unpublished)

IEEE Working Group. (1973, November). Common format for exchange of solved load flow data. Power Apparatus and Systems, IEEE Transactions on, 92(6), 1916-1925.

Kallrath, J., Pardalos, P., Rebennack, S., & Scheidt, M. (2009). Optimization in the energy industry. Springer.

Kienzle, F., Krause, T., Egli, K., Geidl, M., & Andersson, G. (2007, September). Analysis of strategic behaviour in combined electricity and gas markets using agent-based computational economics. In 1st European workshop on energy market modelling using agent-based computational economics (p. 121-141). Karlsruhe, Germany.

Kietzmann, T. C., & Riedmiller, M. (2009). The neuro slot car racer: Reinforcement learning in a real world setting. Machine Learning and Applications, Fourth International Conference on, 0, 311-316.

Kirschen, D. S., & Strbac, G. (2004). Fundamentals of power system economics. Chichester: John Wiley & Sons.

Krause, T., & Andersson, G. (2006). Evaluating congestion management schemes in liberalized electricity markets using an agent-based simulator. In Power Engineering Society General Meeting, 2006. IEEE.

Krause, T., Andersson, G., Ernst, D., Beck, E., Cherkaoui, R., & Germond, A. (2004). Nash Equilibria and Reinforcement Learning for Active Decision Maker Modelling in Power Markets. In Proceedings of 6th IAEE European Conference 2004, modelling in energy economics and policy.

Krause, T., Beck, E. V., Cherkaoui, R., Germond, A., Andersson, G., & Ernst, D. (2006). A comparison of Nash equilibria analysis and agent-based modelling for power markets. International Journal of Electrical Power & Energy Systems, 28(9), 599-607.

Li, H., & Tesfatsion, L. (2009a, July). The ames wholesale power market test bed: A computational laboratory for research, teaching, and training. In IEEE Proceedings, Power and Energy Society General Meeting. Alberta, Canada.

Li, H., & Tesfatsion, L. (2009b, March). Capacity withholding in restructured wholesale power markets: An agent-based test bed study. In Power systems conference and exposition, 2009 (p. 1-11).

Lincoln, R., Galloway, S., & Burt, G. (2007, May 23-25). Unit commitment and system stability under increased penetration of distributed generation. In Proceedings of the 4th International Conference on the European Energy Market, 2007. EEM 2007. Cracow, Poland.

Lincoln, R., Galloway, S., & Burt, G. (2009, May). Open source, agent-based energy market simulation with Python. In Proceedings of the 6th International Conference on the European Energy Market, 2009. EEM 2009. (p. 1-5).

Lincoln, R., Galloway, S., Burt, G., & McDonald, J. (2006, 6-8). Agent-based simulation of short-term energy markets for highly distributed power systems. In Proceedings of the 41st international universities power engineering conference, 2006. UPEC '06. (Vol. 1, p. 198-202).

Maei, H. R., & Sutton, R. S. (2010). Gq(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In In proceedings of the third conference on artificial general intelligence. Lugano, Switzerland.

McCulloch, W., & Pitts, W. (1943, December 21). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biology, 5(4), 115-133.

Micola, A. R., Banal-Estañol, A., & Bunn, D. W. (2008, August). Incentives and coordination in vertically related energy markets. Journal of Economic Behavior & Organization, 67(2), 381-393.

Micola, A. R., & Bunn, D. W. (2008). Crossholdings, concentration and information in capacity-constrained sealed bid-offer auctions. Journal of Economic Behavior & Organization, 66(3-4), 748-766.

Minkel, J. R. (2008, August 13). The 2003 northeast blackout–five years later. Scientific American.

Momoh, J., Adapa, R., & El-Hawary, M. (1999, Feb). A review of selected optimal power flow literature to 1993. I. Nonlinear and quadratic programming approaches. Power Systems, IEEE Transactions on, 14(1), 96-104.

Momoh, J., El-Hawary, M., & Adapa, R. (1999, Feb). A review of selected optimal power flow literature to 1993. II. Newton, linear programming and interior point methods. Power Systems, IEEE Transactions on, 14(1), 105-111.

Moody, J., & Saffell, M. (2001, July). Learning to trade via direct reinforcement. IEEE Transactions on Neural Networks, 12(4), 875-889.

Moody, J., Wu, L., Liao, Y., & Saffell, M. (1998). Performance functions and reinforcement learning for trading systems and protfolios. Journal of Forecasting, 17, 441-470.

Naghibi-Sistani, M., Akbarzadeh-Tootoonchi, M., Javidi-D.B., M., & Rajabi-Mashhadi, H. (2006, November). Q-adjusted annealing for Q-learning of bid selection in market-based multisource power systems. Generation, Transmission and Distribution, IEE Proceedings, 153(6), 653-660.

Nash, J. F. (1950, January). Equilibrium points in n-person games. Proceedings of the National Academy of Sciences of the United States of America, 36(1), 48-49.

Nash, J. F. (1951, September). Non-cooperative games. The Annals of Mathematics, 54(2), 286-295. Available from `http://dx.doi.org/10.2307/1969529`

National Electricity Transmission System Operator. (2007, September). Large combustion plant directive (Tech. Rep.). National Grid Electricity Transmission plc. (GCRP 07/32)

National Electricity Transmission System Operator. (2010, May). 2010 National Electricity Transmission System Seven Year Statement (Tech. Rep.). National Grid Electricity Transmission plc.

Nicolaisen, J., Petrov, V., & Tesfatsion, L. (2002, August). Market power and efficiency in a computational electricity market with discriminatory double-auction pricing. Evolutionary Computation, IEEE Transactions on, 5(5), 504-523.

Nicolaisen, J., Smith, M., Petrov, V., & Tesfatsion, L. (2000). Concentration and capacity effects on electricity market power. In Evolutionary Computation. Proceedings of the 2000 Congress on (Vol. 2, p. 1041-1047).

Overbye, T., Cheng, X., & Sun, Y. (2004, Jan.). A comparison of the AC and DC power flow models for LMP calculations. In System sciences, 2004. Proceedings of the 37th annual Hawaii international conference on (p. 9-).

Peshkin, L., & Savova, V. (2002). Reinforcement learning for adaptive routing. In Neural Networks, 2002. IJCNN 2002. Proceedings of the 2002 International Joint Conference on (Vol. 2, p. 1825-1830).

Peters, J., & Schaal, S. (2006, October). Policy gradient methods for robotics. In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on (p. 2219-2225).

Peters, J., & Schaal, S. (2008). Natural actor-critic. Neurocomputing, 71(7-9), 1180-1190.

Rastegar, M. A., Guerci, E., & Cincotti, S. (2009, May). Agent-based model of the Italian wholesale electricity market. In Energy Market, 2009. 6th International Conference on the European (p. 1-7).

Riedmiller, M. (2005). Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In In 16th European conference on machine learning (pp. 317–328). Springer.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: the rprop algorithm.

Robbins, H. (1952). Some aspects of the sequential design of experiments. Bulletin American Mathematical Society, 58(5), 527-535.

Roth, A. E., Erev, I., Fudenberg, D., Kagel, J., Emilie, J., & Xing, R. X. (1995). Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. Games and Economic Behavior, 8(1), 164-212.

Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., et al. (2010). PyBrain. Journal of Machine Learning Research, 11, 743-746.

Schweppe, F., Caramanis, M., Tabors, R., & Bohn, R. (1988). Spot pricing of electricity. Dordrecht: Kluwer Academic Publishers Group.

Sharpe, W. F. (1966, January). Mutual fund performance. Journal of Business, 119-138.

Sharpe, W. F. (1994). The Sharpe ratio. The Journal of Portfolio Management, 49-58.

Stott, B., & Alsac, O. (1974, May). Fast decoupled load flow. Power Apparatus and Systems, IEEE Transactions on, 93(3), 859-869.

Sun, J., & Tesfatsion, L. (2007a). Dynamic testing of wholesale power market designs: An open-source agent-based framework. Computational Economics, 30(3), 291-327.

Sun, J., & Tesfatsion, L. (2007b, June). Open-source software for power industry research, teaching, and training: A DC-OPF illustration. In Power Engineering Society General Meeting, 2007. IEEE (p. 1-6).

Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction. MIT Press. Gebundene Ausgabe.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems (Vol. 12, p. 1057-1063).

Tellidou, A., & Bakirtzis, A. (2007, Novemeber). Agent-based analysis of capacity withholding and tacit collusion in electricity markets. Power Systems, IEEE Transactions on, 22(4), 1735-1742.

Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. Neural Computation, 6(2), 215-219.

Tesfatsion, L., & Judd, K. L. (2006). Handbook of computational economics, volume 2: Agent-based computational economics (handbook of computational economics). Amsterdam, The Netherlands: North-Holland Publishing Co.

Tinney, W., & Hart, C. (1967, Novemeber). Power flow solution by Newton's method. Power Apparatus and Systems, IEEE Transactions on, 86(11), 1449-1460.

Tsitsiklis, J. N., & Roy, B. V. (1994). Feature-based methods for large scale dynamic programming. In Machine learning (p. 59-94).

United Nations. (2003, December 9). World population in 2300. In Proceedings of the United Nations, Expert Meeting on World Population in 2300.

U.S.-Canada Power System Outage Task Force. (2004, April). Final report on the august 14, 2003 blackout in the united states and canada: Causes and recommendations (Tech. Rep.). North American Electric Reliability Corporation.

Veit, D., Weidlich, A., Yao, J., & Oren, S. (2006). Simulating the dynamics in two-settlement electricity markets via an agent-based approach. International Journal of Management Science and Engineering Management, 1(2), 83-97.

Vengerov, D. (2008). A gradient-based reinforcement learning approach to dynamic pricing in partially-observable environments. Future Generation Computer Systems, 24(7), 687-693.

Visudhiphan, P. (2003). An agent-based approach to modeling electricity spot markets. Unpublished doctoral dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Visudhiphan, P., & Ilic, M. (1999, February). Dynamic games-based modeling of electricity markets. In Power Engineering Society 1999 Winter Meeting, IEEE (Vol. 1, p. 274-281).

Wang, H., Murillo-Sanchez, C., Zimmerman, R., & Thomas, R. (2007, Aug.). On computational issues of market-based optimal power flow. Power Systems, IEEE Transactions on, 22(3), 1185-1193.

Wang, J., Koritarov, V., & Kim, J.-H. (2009, July). An agent-based approach to modeling interactions between emission market and electricity market. In Power Energy Society General Meeting, 2009. PES 2009. IEEE (p. 1-8).

Weidlich, A., & Veit, D. (2006, July 7-10). Bidding in interrelated day-ahead electricity markets - insights from an agent-based simulation model. In Proceedings of the 29th IAEE International Conference.

Weidlich, A., & Veit, D. (2008, July). A critical survey of agent-based wholesale electricity market models. Energy Economics, 30(4), 1728-1759.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In Machine learning (p. 229-256).

Wood, A. J., & Wollenberg, B. F. (1996). Power Generation Operation and

Control (second ed.). New York: Wiley, New York.

Yao, J., Adler, I., & Oren, S. S. (2008). Modeling and computing two-settlement oligopolistic equilibrium in a congested electricity network. Operations Research, 56(1), 34-47.

Yao, J., Oren, S. S., & Adler, I. (2007). Two-settlement electricity markets with price caps and cournot generation firms. European Journal of Operational Research, 181(3), 1279-1296.

Zimmerman, R. (2010, March 19). MATPOWER 4.0b2 User's Manual [Computer software manual]. School of Electrical Engineering, Cornell University, Ithaca, NY 14853.

Zimmerman, R., Murillo-Sánchez, C., & Thomas, R. J. (2009, July). MATPOWER's extensible optimal power flow architecture. In IEEE PES General Meeting. Calgary, Alberta, Canada.

Appendix A

Open Source Electric Power Engineering Software

For the purposes of this thesis the Matlab source code from Matpower was translated into the Python programming language and released under the project name: Pylon. It was translated to allow existing implementations of policy gradient reinforcement learning methods, from the PyBrain machine learning library (Schaul et al., 2010), to be coupled with Matpower's scalable and extensible optimal power flow formulations. With permission from the Matpower developers, the resulting code was released under the terms of the Apache License version 2.0 (Lincoln, Galloway, & Burt, 2009) and this section describes the project in the context of other open source Electrical Power Engineering software tools to illustrate the contribution made.

A.1 MATPOWER

Since 1996, a team of researchers from the Power Systems Engineering Research Center (PSERC) at Cornell University have been developing Matpower: a package of Matlab[1] workspace files for solving power flow and optimal power flow problems (Zimmerman, Murillo-Sánchez, & Thomas, 2009). Initial development was part of the PowerWeb project in which the team created a power exchange auction market simulator that could be accessed by multiple users simultaneously through a web-based interface. Matpower was originally available under a custom license that permitted use for any purpose providing the project and authors were cited correctly, but since version 4.0b2 it has been released under the less permissive Gnu General Public License (GPL), version 3. Matpower has

[1]Matlab is a registered tradeamark of The Mathworks, Inc.

Package	Language	Licence	PF	MPF	DCOPF	ACOPF	CPF	SSSA	TDS	SE	SP	GUI	RL
AMES	Java	GPL			●							●	●
DCOPFJ	Java	GPL			●								
GridLab-D	C++	BSD	●	●							●		
MatDyn	Matlab								●		●		
MATPOWER	Matlab	GPL	●		●	●	●			●	●		
OpenDSS	Pascal	BSD	●	●							●	●	
PSAT	Matlab	GPL	●			●	●	●	●		●	●	
Pylon	Python	Apache	●		●	●				●	●	●	●
TEFTS	C						●		●		●		
VST	Matlab		●				●	●	●		●	●	
UWPFLOW	C						●				●		

Table A.1: Open source electric power engineering software feature matrix.

become very popular in education and research and has an active mailing list that is moderated by Dr Ray Zimmerman of PSERC.

Matpower includes five solvers for AC and DC power flow. The default solver uses Newton's method (Tinney & Hart, 1967) with the full Jacobian matrix updated in each iteration. Two variations on the fast decoupled method (Stott & Alsac, 1974) described in Amerongen (1989) provide quicker convergence for certain networks. The standard Gauss-Seidel method (Glimn & Stagg, 1957) is provided for academic purposes and the DC solver provides non-iterative solutions. The properties of Matlab sparse matrices are exploited to allow solvers to scale well with very large systems. All functions are run from the Matlab command-line or from within users programs and no graphical user interface is provided.

Starting with version 4.0, Matpower includes the Matlab Interior Point Solver (MIPS) that can be used for solving DC and AC optimal power flow problems (H. Wang et al., 2007). Previously, FMINCON from the Matlab Optimization Toolbox[2] was required or one of a suite of high performance closed-source solvers: TSPOPF is a collection of three AC optimal power flow solvers, implemented in the C programming language and released as Matlab MEX files. It includes the original implementation of the step-controlled interior point method from which MIPS was derived. MINOPF provides an interface to the Fortran based MINOS[3] solver, developed at the Systems Optimization Laboratory at Stanford University, and is available only for educational and research purposes. Since version 4.0b4 Matpower has also included an interface to IPOPT from the COIN-OR project that provides an alternative open source solution to MIPS. DC optimal power flow problems can be solved with a Quadratic Programming interface to MIPS or using a MEX interface to BPMPD: a commercial interior point method for linear and quadratic programming.

Matpower has an extensible optimal power flow formulation that allows users to introduce additional optimisation variables and problem constraints. It is used internally to extend the standard optimisation formulation to support piecewise linear cost functions, dispatchable loads, generator PQ capability curves and branch angle difference limit constraints. Examples of possible additional extensions include: reserve requirements, environmental costs and contingency constraints.

Matpower currently runs on Matlab, a commercial software product from The Mathworks that is supported on all major platforms, or on Gnu/Octave, a

[2]Optimization Toolbox is a registered trademark of The Mathworks, Inc.
[3]MINOS is trademark of Stanford Business Software, Inc.

free program for numerical computation with strong Matlab compatibility.

A.2 MATDYN

MATDYN is an extension to MATPOWER developed by Stijn Cole from the Katholieke Universiteit Leuven for dynamic analysis of electric power systems. It was first released in 2009 under MATPOWER's custom license. It uses the same programming style and extends the MATPOWER case format with structs for dynamic generator and event data. MATDYN uses MATPOWER to obtain a power flow solution that is then used in solving a system of differential algebraic equations representing the power system. Results from MATDYN have been validated by Cole (2010) against those obtained from PSS/E[4] and the Power System Analysis Toolbox (See Section A.3, below) and show good correspondence.

A.3 PSAT

The Power System Analysis Toolbox (PSAT) is a Matlab toolbox for static and dynamic analysis of electric power systems developed by Federico Milano, currently an Assistant Professor at the University of Castilla in Spain. It is released under the terms of the GNU GPL version 2 and offers routines for:

- Power flow,
- Bifurcation analysis,
- Optimal power flow,
- Small signal stability analysis,
- Time domain simulation and
- Phasor measurement unit placement.

A large number of input data formats are supported through Perl scripts and simulation reports can be exported as plain text, Excel spreadsheets or $\LaTeX\,2_\varepsilon$ code. PSAT may be run from the Matlab command-line or through a Matlab based graphical user interface. The interface can be used with Simulink[5] to construct cases such as the network from the UK Generic Distribution System shown

[4]PSS/E is a registered trademark of Siemens Power Transmission & Distribution, Inc. Power Technologies International.

[5]Simulink is a registered trademark of The Mathworks, Inc.

in Figure ??. A slightly modified version of PSAT that can be run from the Gnu/Octave command-line is also available.

Optimal power flow problems are solved via an interface to the General Algebraic Modeling System (GAMS). GAMS defines optimisation problems using a high-level modelling language and has a large solver portfolio that includes all of the major commercial and academic solvers. The interface can be used for solving single period optimal power flow problems where the objective function can model maximisation of social benefit, maximisation of the distance to the maximum loading condition or multi-objective of a combination of these. Multi-period optimal power flow is formulated as a mixed integer problem with linearised power balance constraints. The objective function models maximisation of social welfare, but is extended to include start-up and shutdown costs.

Power flow and dynamic data are often separated in electric power simulation tools, but in PSAT they are integrated. This combined with the large number of routines supported by PSAT can make the code base difficult to understand and modify. However, comprehensive documentation is included with PSAT and the mailing list is very active. The price of GAMS licenses and the need for optimal power flow problems to be converted to the GAMS language before being solved may be considered barriers to its selection for certain projects.

A.4 UWPFLOW

UWPFLOW is a research tool for voltage stability analysis developed at the University of Waterloo, Ontario, and the University of Wisconsin-Madison. It is written in ANSI-C and is available as open source for research purposes only. The program can be run with the terminal command

```
$ uwpflow [-options] input_file
```

where `input_file` is the path to a data file in the IEEE common data format (CDF) (IEEE Working Group, 1973) that may contain High-Voltage Direct Current (HVDC) and Flexible Alternating Current Transmission System (FACTS) device data. Output is also in CDF and can include additional data for post-processing, including values for nose curve plots. An interface to UWPFLOW is provided with PSAT and can be used for bifurcation analysis.

A.5 TEFTS

The University of Waterloo also hosts TEFTS – a transient stability program for studying energy functions and voltage stability phenomena in AC/HVDC dynamic power system models. It too is written in ANSI-C and is licensed for research purposes only. An executable file for DOS is provided and the source package contains a simple example.

A.6 VST

The Voltage Stability Toolbox (VST) is a Matlab toolbox, developed at the Center for Electric Power Engineering at Drexel University in Philidelphia, for investigating stability and bifurcation issues in power systems. The source is available for any purpose providing that the authors are suitably cited. VST features routines for:

- Power flow,
- Time domain simulation,
- Static and dynamic bifurcation analysis,
- Singularity analysis and
- Eigenvalue analysis.

The feature matrix in Table A.1 shows the similar capabilities of VST and PSAT. It was developed around the same time and has the same goals for educational and research applications. However, VST does not have the same quality of documentation, graphical user interface or such an active community of users and developers.

A.7 OpenDSS

In November 2008, the Open Distribution System Simulator (OpenDSS) was released by the Electric Power Research Institute (EPRI) as open source. Development of OpenDSS began in April 1997 and it has been used extensively in distributed generation impact assessments.

OpenDSS supports steady-state analysis in the frequency domain, including power flow, harmonics and dynamics. Arbitrary n-phase unbalanced circuit analysis is supported using an object orientated data model. Circuit elements are defined in Object Pascal and solutions are found using KLUSolve: a linear sparse matrix solver written in C and C++. The OpenDSS Pascal code is available under the Berkeley Software Distribution (BSD) license, which allows use for almost any purpose. KLUSolve, is available under the Gnu Lesser GPL. Circuits are defined in scripts, using a domain specific language, that may be executed through a graphical user interface or a Common Object Model (COM) interface. The user interface also provides circuit data editing, plotting and power flow visualisation tools.

The power flow solver is fast and can be configured for repeated studies using daily, yearly or duty-cycle data. The multi-phase circuit model allows complex transformer models and fault conditions to be defined and three short-circuit analysis methods are provided. The heritage of OpenDSS is in harmonics and dynamics analysis and it does not support system optimisation.

A.8 GridLab-D

GridLAB-D is an energy system simulation and analysis tool designed to investigate the latest energy technologies. The project was initiated by the U.S. Department of Energy in 2004 and developed at Pacific Northwest National Laboratory. It was released under a BSD-style license in September 2009 and has since been developed in collaboration with industry and acedemia.

A distributed simulation architecture is used to coordinate energy system component interactions over short and long timescales. The core of GridLAB-D is made up of modules for simulating: distribution and transmission systems, commercial and residential buildings, energy markets, power system faults and meteorological systems. GridLAB-D is written in C++ and uses a domain specific language to define models. Additional modules can be written in C++ or Java and Python is under consideration. It is designed for multicore/multiprocessor parallelism and the developers intend to simulate large areas of the U.S. on supercomputers with it. The source code includes reports and data from the Olympic Peninsula Project: a futuristic energy pricing experiment that was used to provide a practical demonstration of GridLAB-D in operation.

GridLAB-D is a unique simulation tool that has the potential to play an important role in future energy system development. Its size and complexity can make for a steep learning curve, but extensive documentation is provided and training courses are run periodically. Activity on the mailing lists is low, suggesting poor uptake, but they are actively supported and a new version is under development.

A.9 AMES

The AMES (Agent-based Modeling of Electricity Systems) power market testbed is a software package that models core features of the Wholesale Power Market Platform: a market design proposed by the Federal Energy Regulatory Commission (FERC) in April 2003 for common adoption in regions of the U.S. (Sun & Tesfatsion, 2007a). The market design features:

- A centralised structure managed by an independent market operator,
- Parallel day-ahead and real-time markets and
- Locational marginal pricing.

Learning agents represent load serving entities or generating companies and learn using Roth-Erev reinforcement learning methods, implemented using the Repast agent simulation toolkit (Gieseler, 2005). Agents learn from the solutions of hourly bid/offer based DC-OPF problems formulated as quadratic programs using the DCOPFJ package (Sun & Tesfatsion, 2007b), described in Section A.10, below.

The capabilities of AMES are demonstrated using a 5-bus network model in Li and Tesfatsion (2009a). The model is provided with AMES and a step-by-step tutorial describes how it may be used. AMES comes with a Swing-based graphical user interface with plotting and table editor tools and is released under the GNU GPL, version 2.

A.10 DCOPFJ

To solve market problems defined in AMES, researchers at Iowa State University developed a stand-alone DC optimal power flow solver in Java named DCOPFJ. It formulates optimal power flow problems as convex quadratic programs which are solved using QuadProgJ. The same researcher developed QuadProgJ as an independent solver that uses a dual active set strictly convex quadratic programming algorithm (Goldfarb & Idnani, 1983). DCOPFJ requires generator costs to be modelled as polynomial functions, of second order or less, and does not use sparse matrices for application to large systems.

A.11 PYLON

Pylon is a translation of Matpower and Matdyn to the Python programming language. It has extensions for agent-based electricity market simulation that provide features similar to those of AMES. Both the DC and AC formulations of the extensible optimal power flow model (Zimmerman et al., 2009) from Matpower are implemented. Either a Python version of MIPS or an interface to IPOPT from COIN-OR can be used to compute solutions. The sparsity of the problems is exploited throughout the solution process using matrix packages from SciPy and bindings to SuperLU or UMFPACK for solving sparse sets of linear equations. Scripts are provided for reading and writing data files in PSS/E, Matpower and PSAT format. A wide variety of learning methods are available in Pylon due to its use of the PyBrain machine learning library (Schaul et al., 2010). PyBrain also provides the artificial neural network models used for policy function approximation, that may be accelerated using C extension modules from ARAC.

In addition to its market simulation capabilities, Pylon also features solvers for power flow problems (using fast decoupled or Newton's method), state estimation, continuation power flow and time domain simulation. Pylon includes both a text interface and a graphical user interface (GUI) that uses Tkinter, which is included with Python as standard and thus imposes no additional dependencies. A more feature rich GUI is provided by plug-ins for Puddle: an extensible, GUI toolkit independent integrated development environment, developed for the purposes of this thesis.

The use of matrix libraries from NumPy and SciPy has allowed Pylon (with the permission of the Matpower developers) to be released under the Apache license, version 2.0. This allows Pylon to be used as a library in proprietary software as well as free and open source tools since derivatives of the source code may be made available under more restrictive terms than the original Apache license. This is in contrast to strong "copyleft" licenses, such as the Gnu GPL, that require the same rights to be preserved in modified versions of the work.

A.12 Summary

Appendix B

Case Data

This appendix provides the data for the electric power system models used in Chapters 5 and 6.

B.1 6-Bus Case

The data for the six bus case adapted from Wood and Wollenberg (1996, pp. 104, 112, 119, 123-124, 549) is presented in this section. The data was imported from the "case6ww.m" case file provided with MATPOWER. Figure **??** illustrates the structure of the model and shows the bus injections for the AC unit decommitment optimal power flow solution. Table B.1 lists the bus data, Table B.2 lists the generator data and Table B.3 lists the branch data.

Bus	P_d	Q_d	G_s	B_s	V_{base}	V_{max}	V_{min}
1	0	0	0	0	230	1.05	1.05
2	0	0	0	0	230	1.05	1.05
3	0	0	0	0	230	1.07	1.07
4	70	70	0	0	230	1.05	0.95
5	70	70	0	0	230	1.05	0.95
6	70	70	0	0	230	1.05	0.95

Table B.1: 6-bus case bus data.

B.2 IEEE Reliability Test System

This appendix provides the data from the IEEE Reliability Test System (Application of Probability Methods Subcommittee, 1979) that was imported from the "case24_ieee_rts.m" case file that is provided with MATPOWER and was originally contributed by

Bus	P_{max}	P_{min}	V_g	Q_{max}	Q_{min}
1	1.05	200	50	100	-100
2	1.05	150	37.5	100	-100
3	1.07	180	45	100	-100

Table B.2: 6-bus case generator data.

From	To	r	x	b_c	S_{max}	τ	θ_{ph}
1	2	0.1	0.2	0.04	40	0	0
1	4	0.05	0.2	0.04	60	0	0
1	5	0.08	0.3	0.06	40	0	0
2	3	0.05	0.25	0.06	40	0	0
2	4	0.05	0.1	0.02	60	0	0
2	5	0.1	0.3	0.04	30	0	0
2	6	0.07	0.2	0.05	90	0	0
3	5	0.12	0.26	0.05	70	0	0
3	6	0.02	0.1	0.02	80	0	0
4	5	0.2	0.4	0.08	20	0	0
5	6	0.1	0.3	0.06	40	0	0

Table B.3: 6-bus case branch data.

Bruce Wollenberg. Figure **??** illustrates the model structure and shows the bus injections for the AC unit de-commitment optimal power flow solution for all generators at marginal cost. Table B.4 lists the bus data, Table B.5 lists the generator data, Table B.6 lists the branch data and Table B.7 lists the generator cost data provided by Georgia Tech Power Systems Control and Automation Laboratory.

Bus	P_d	Q_d	G_s	B_s	V_{base}	V_{max}	V_{min}
1	108	22	0	0	138	1.05	0.95
2	97	20	0	0	138	1.05	0.95
3	180	37	0	0	138	1.05	0.95
4	74	15	0	0	138	1.05	0.95
5	71	14	0	0	138	1.05	0.95
6	136	28	0	-100	138	1.05	0.95
7	125	25	0	0	138	1.05	0.95
8	171	35	0	0	138	1.05	0.95
9	175	36	0	0	138	1.05	0.95
10	195	40	0	0	138	1.05	0.95
11	0	0	0	0	230	1.05	0.95
12	0	0	0	0	230	1.05	0.95
13	265	54	0	0	230	1.05	0.95
14	194	39	0	0	230	1.05	0.95
15	317	64	0	0	230	1.05	0.95
16	100	20	0	0	230	1.05	0.95
17	0	0	0	0	230	1.05	0.95
18	333	68	0	0	230	1.05	0.95
19	181	37	0	0	230	1.05	0.95
20	128	26	0	0	230	1.05	0.95
21	0	0	0	0	230	1.05	0.95
22	0	0	0	0	230	1.05	0.95
23	0	0	0	0	230	1.05	0.95
24	0	0	0	0	230	1.05	0.95

Table B.4: IEEE RTS bus data.

Bus	P_{max}	P_{min}	V_g	Q_{max}	Q_{min}	Type
1	20	16	1.035	10	0	U20
1	20	16	1.035	10	0	U20
1	76	15.2	1.035	30	-25	U76
1	76	15.2	1.035	30	-25	U76
2	20	16	1.035	10	0	U20
2	20	16	1.035	10	0	U20
2	76	15.2	1.035	30	-25	U76
2	76	15.2	1.035	30	-25	U76
7	100	25	1.025	60	0	U100
7	100	25	1.025	60	0	U100
7	100	25	1.025	60	0	U100
13	197	69	1.02	80	0	U197
13	197	69	1.02	80	0	U197
13	197	69	1.02	80	0	U197
14	0	0	0.98	200	-50	SynCond
15	12	2.4	1.014	6	0	U12
15	12	2.4	1.014	6	0	U12
15	12	2.4	1.014	6	0	U12
15	12	2.4	1.014	6	0	U12
15	12	2.4	1.014	6	0	U12
15	155	54.3	1.014	80	-50	U155
16	155	54.3	1.017	80	-50	U155
18	400	100	1.05	200	-50	U400
21	400	100	1.05	200	-50	U400
22	50	10	1.05	16	-10	U50
22	50	10	1.05	16	-10	U50
22	50	10	1.05	16	-10	U50
22	50	10	1.05	16	-10	U50
22	50	10	1.05	16	-10	U50
22	50	10	1.05	16	-10	U50
23	155	54.3	1.05	80	-50	U155
23	155	54.3	1.05	80	-50	U155
23	350	140	1.05	150	-25	U350

Table B.5: IEEE RTS generator data.

From	To	r	x	b_c	S_{max}	τ	θ_{ph}
1	2	0.0026	0.0139	0.4611	175	0	0
1	3	0.0546	0.2112	0.0572	175	0	0
1	5	0.0218	0.0845	0.0229	175	0	0
2	4	0.0328	0.1267	0.0343	175	0	0
2	6	0.0497	0.192	0.052	175	0	0
3	9	0.0308	0.119	0.0322	175	0	0
3	24	0.0023	0.0839	0	400	1.03	0
4	9	0.0268	0.1037	0.0281	175	0	0
5	10	0.0228	0.0883	0.0239	175	0	0
6	10	0.0139	0.0605	2.459	175	0	0
7	8	0.0159	0.0614	0.0166	175	0	0
8	9	0.0427	0.1651	0.0447	175	0	0
8	10	0.0427	0.1651	0.0447	175	0	0
9	11	0.0023	0.0839	0	400	1.03	0
9	12	0.0023	0.0839	0	400	1.03	0
10	11	0.0023	0.0839	0	400	1.02	0
10	12	0.0023	0.0839	0	400	1.02	0
11	13	0.0061	0.0476	0.0999	500	0	0
11	14	0.0054	0.0418	0.0879	500	0	0
12	13	0.0061	0.0476	0.0999	500	0	0
12	23	0.0124	0.0966	0.203	500	0	0
13	23	0.0111	0.0865	0.1818	500	0	0
14	16	0.005	0.0389	0.0818	500	0	0
15	16	0.0022	0.0173	0.0364	500	0	0
15	21	0.0063	0.049	0.103	500	0	0
15	21	0.0063	0.049	0.103	500	0	0
15	24	0.0067	0.0519	0.1091	500	0	0
16	17	0.0033	0.0259	0.0545	500	0	0
16	19	0.003	0.0231	0.0485	500	0	0
17	18	0.0018	0.0144	0.0303	500	0	0
17	22	0.0135	0.1053	0.2212	500	0	0
18	21	0.0033	0.0259	0.0545	500	0	0
18	21	0.0033	0.0259	0.0545	500	0	0
19	20	0.0051	0.0396	0.0833	500	0	0
19	20	0.0051	0.0396	0.0833	500	0	0
20	23	0.0028	0.0216	0.0455	500	0	0
20	23	0.0028	0.0216	0.0455	500	0	0
21	22	0.0087	0.0678	0.1424	500	0	0

Table B.6: IEEE RTS branch data.

Gen	C_{up}	a	b	c	Type
1	1500	0	130	400.685	U20
2	1500	0	130	400.685	U20
3	1500	0.01414	16.0811	212.308	U76
4	1500	0.01414	16.0811	212.308	U76
5	1500	0	130	400.685	U20
6	1500	0	130	400.685	U20
7	1500	0.01414	16.0811	212.308	U76
8	1500	0.01414	16.0811	212.308	U76
9	1500	0.05267	43.6615	781.521	U100
10	1500	0.05267	43.6615	781.521	U100
11	1500	0.05267	43.6615	781.521	U100
12	1500	0.00717	48.5804	832.758	U197
13	1500	0.00717	48.5804	832.758	U197
14	1500	0.00717	48.5804	832.758	U197
15	1500	0	0	0	SynCond
16	1500	0.32841	56.564	86.3852	U12
17	1500	0.32841	56.564	86.3852	U12
18	1500	0.32841	56.564	86.3852	U12
19	1500	0.32841	56.564	86.3852	U12
20	1500	0.32841	56.564	86.3852	U12
21	1500	0.00834	12.3883	382.239	U155
22	1500	0.00834	12.3883	382.239	U155
23	1500	0.00021	4.4231	395.375	U400
24	1500	0.00021	4.4231	395.375	U400
25	1500	0	0.001	0.001	U50
26	1500	0	0.001	0.001	U50
27	1500	0	0.001	0.001	U50
28	1500	0	0.001	0.001	U50
29	1500	0	0.001	0.001	U50
30	1500	0	0.001	0.001	U50
31	1500	0.00834	12.3883	382.239	U155
32	1500	0.00834	12.3883	382.239	U155
33	1500	0.00490	11.8495	665.109	U350

Table B.7: IEEE RTS generator cost data.