Real-Time Application of Optimization-Enabled Electromagnetic Transient Simulation

by
In Kwon Park

A Thesis Submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, MB

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made available to the public electronically.

Acknowledgement

The journey in pursuit of this academic goal began some time ago. The journey is now coming close to the end, but probably the end will signify the beginning of the next journey. During this journey, which this thesis will chronicle in detail, I have been privileged to be under the shade of several great people. They were great with the brightness of their minds, and sometimes with their deep generosity and shrewd wisdom.

Probably Dr. Ani Gole, my academic advisor of this research, has been the individual whom I have been dependent upon during this journey. His encouragement has always been a spur to push me forward. His insight has always inspired me with a new horizon where a new road revealed itself.

During the journey, I was able to enjoy new wind and fresh fragrance from the trees along with the passage. Some of my colleagues in the school were some of those trees. The valuable discussions with Dr. Xi Lin, Dr. Yuefung Liang, Dr. Ali Dekhordi and Mr. Il Do Yoo are some of the invaluable during the journey. Those bright minds in RTDS office, Dr. Trevor Maguire, Dr. Yi Zhang and Mr. Rick Kuffel were the trees in the other side of the road, which the scenery bears their imprints.

My parents in Korea, my wife, Ji Youn, and two children, Youngsu and Eunha, have been the fountains of renewed strength during this long journey. Without their understanding and sacrifice, this journey would not come to its terminal.

Lastly, nothing in this journey was possible without the Lord's providence and planning. All the praise and honor is His. Only the faults and blames are mine.

Call to me and I will answer you, and will tell you
great and hidden things that you have not known
(Jeremiah 33:3 English Standard Version)

Abstract

This thesis presents a new way of combining non-linear optimization algorithms and electromagnetic transient (EMT) simulation. In this new combination, a non-linear optimization algorithm utilizes a real-time EMT simulation environment as objective function evaluator. In previous research work, off-line EMT simulation software was used as the objective function evaluator in the combination. However, as more applications of the combination were made, the combination between non-linear optimization algorithms and off-line EMT simulation software revealed new need, which this research work addresses.

The first need arose from the speed of simulation of the off-line EMT simulation software. With a certain breed of non-linear optimization algorithms, heuristics-based algorithms in particular, a large number of objective function evaluations are required before the termination or convergence criterion in the selected algorithms is satisfied. Sometimes, the number of evaluations as well as the complexity of the simulation case where the objective function is based upon translates into a very long simulation time, which exceeds the boundary of given resources. This research work attempts to address this simulation timing issue by capitalizing on the real timeness of the simulation environment used as well as utilizing multiple instances of the simulation environment in parallel.

The second need arose from the modeling requirement of the off-line EMT simulation software. In order to properly design the necessary objective function evaluator, which is largely a simulation case, a large amount of information needs to be embedded into the case. Under certain circumstances, the necessary information would not be available. Therefore, the simulation case needs to include approximations which may compromise the end result. This limitation becomes more obvious when a real world device such as a commercial controller becomes involved. On the contrary, this limitation can be addressed by the use of a real-time simulation environment because this environment can be directly interfaced with the real world device. In this way, the need for detailed information regarding the device is eliminated. This elimination extends the application of the

combination, between the non-linear optimization algorithm and EMT type simulation environment.

The effectiveness of the proposed approach is demonstrated by various examples throughout this thesis.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

NOMENCLATURE

A/D: Analog to digital

AIS: Artificial immune system

ALU: Arithmetic and logic unit

COTS: Commercial off-the-shelf

D/A: Digital to analog

DSC: Digital signal controller

DSP: Digital signal processor

DSK: DSP starter kit

DOE: Design of experiment

EMT: Electromagnetic transient

EMTP: Electromagnetic transient program

FEM: Finite element method

GA: Genetic algorithm

GPU: Graphical processing unit

GUI: Graphical user interface

HILS: Hardware in loop simulation

HVDC: High voltage DC

I/O: Input and output

ISE: Integrated squared error

MADD: Multiply and add

MMC: Multi module converter

NLO: Non-linear optimization

OE: Optimization enabled

OE-EMT: Optimization enabled electromagnetic transient

OF: Objective function

PSO: Particle swarm algorithm

PI: Proportional-Integral

PSS: Power system stabilizer

PWM: Pulse width modulation

RSM: Response surface method

RTDS: Real time digital simulator

SA: Simulated annealing

TCP/IP: Transmission control protocol/Internet protocol

TNA: Transient network analyzer

TSA: Transient stability analysis

UDP/IP: User datagram protocol/Internet protocol

1 INTRODUCTION

Modern society is increasingly dependent on electricity and thus the maintenance of a healthy power system, as well as the prevention of any electrical failure is crucial. The catastrophic results of the massive power blackout in north-eastern America in the year 2003 [1-3] emphasized the necessity for a constant electricity supply in modern society. Ensuring a steady supply of power as well as enhancing power system reliability are the two most important goals pursued ceaselessly by power system engineers. Many computer-aided power system simulation programs have been developed for that purpose. Traditionally, load-flow programs or transient stability analysis programs have been used by mainstream power system engineers. However, in recent years the need for more precise modeling and simulation has arisen because many new power system devices (some of which are based on power electronics) have been introduced to power systems. As well, the deregulation of the utility industry is imposing additional stress on power systems. These trends have propelled the operating conditions of power system devices into new and extreme regions seldom sought before, thus intensifying the need for precise modeling.

Time domain simulation of power systems

Simulation of power systems transients plays a significant role in the design and operation of modern power systems [4]. If the complexity of a power system problem increases beyond the scope of an analytical solution, simulation becomes the only practical option. Simulation is widely used in the determination of critical component ratings such as insulation levels and energy requirements for arrestors. In the analysis and design of power electronic apparatuses and controls, simulation can be used for optimizing the control system and for determining unforeseen harmonic problems. Simulation also plays a significant role in power system relaying and protection. Complex relaying algorithms can be developed and checked using transient simulation [5]. The development of real-time simulation capability has enabled real world equipment evaluation and testing. Such techniques are widely employed in the protective relay industry.

EMTP Program

Since Herman Dommel published his pioneering works [6-8], electromagnetic transient simulation programs (i.e., EMT-type programs) have been expanding in the power system simulation realm [9-11]. The EMT-type programs can model a power system apparatus at a higher level of accuracy than transient stability level analysis. However, higher accuracy is accompanied by a heavier computational burden. Transient stability programs also simulate system dynamics but are limited to slow transients only, such as electromechanical transients. Thus, rare situations that occur in power systems and cannot be fully explained by more traditional power system simulation tools can now be investigated with EMT-type programs. As the processing power of the PC (i.e., personal computer) increases and many widely used EMT-type simulation programs are developed, the power system analysis areas once considered too complex for modeling with available computing resources and time can now be considered. One EMT-type simulation program now widely accepted is the PSCAD software [12, 13].

Need for better controller tuning method

As power systems expand, complex devices such as High Voltage Direct Current (i.e., HVDC) transmission systems [14, 15], Static Var Compensation (i.e., SVC) systems [16, 17] and Flexible AC Transmission Systems (i.e., FACTS) [18] are being used increasingly. Thus, designing an effective controller becomes challenging because describing a plant in a mathematically closed formulation becomes difficult or sometimes impossible. Hence, traditional controller design approaches such as root locus analysis [19] can only be used in an approximate way. Instead, the use of EMT simulation is now more common. The desirable controller parameters are identified by a trial and error process using the simulation programs. Exhaustive search techniques such as the Monte Carlo method [20] are frequently employed to determine appropriate parameters for a specific situation. However, much time and effort can be spent while searching for suitable parameters.

Strategy in controller tuning

A recent improvement of the trial and error approach is a new non-linear optimization technique that uses EMT-type programs as objective function (OF) evaluators [21]. A non-linear optimization algorithm (NLO) controls and iterates with a simulation case in order to search for parameters that will produce a desired system response. This optimization technique directs the search for new trial points and can reduce the required number of iterations. The PSCAD simulation software is an EMT-type program into which this non-linear optimization technique has been successfully incorporated, with the resulting benefit of utilizing the established simulation engine as the trial point evaluator. The total number of iterations can be significantly reduced when the new optimization technique is employed with the off-line simulation programs. However, the time required for computations in off-line simulation programs such as PSCAD increases as the size of the simulation case expands in computational complexity or as a greater number of iterations is required.

The gap between off-line simulation and the real world

Discrepancies can exist between the modeled controller and its real world implementation. One such discrepancy is caused by parasitic filtering effects in the analog input/output stages of the real world controller and other Analog/Digital (A/D) or Digital/Analog (D/A) conversion errors. The limits of the precision in the floating point number representation on the physical hardware of the controller platform can aggravate the discrepancies. These discrepancies make the optimum controller parameters obtained from the optimization routine with an off-line simulation program ineffective in the actual controller. This deficiency identifies a significant issue in the design procedure using the optimization-enabled EMT simulation technique (i.e., OE-EMT technique). Rectifying the identified deficiency with proper utilization of real-time digital simulation is one research objective.

1.1 Real-time digital simulator

The EMT algorithm for power system simulation has been successfully implemented in real-time applications [22, 23]. Real-time can be interpreted in various ways according to the requirement in response time. Real-time in power system simulation usually means that the necessary execution time for a given amount of calculation in simulation cannot exceed the size of the simulation time step. The implication of the meaning brings the real-time simulation in power system closer to the usual definition of hard real-time system [24]. In a hard real-time system, the execution time step of simulation must be synchronized with a real world clock, as real world physical hardware is to be interfaced with the simulation. Hence, a single overflow of execution time with regard to the given time step is considered as simulation failure. The Real Time Digital Simulator (RTDS) from RTDS Technologies in Winnipeg is one of the real-time EMT implementations now widely accepted in the power system industry [25-27]. The power system to be simulated is divided into parts by utilizing the traveling wave characteristic of transmission lines [28]. If the simulation timestep is small enough so that the distance travelled by light in one time step is smaller than the line length, there can be no immediate influence on one side of the line from a remote side connected on the other side. Events on the other side are only sensed at a later time. Thus, the system separates into smaller blocks that can be simulated independently on individual processors. Generated information is conveyed to other processors participating in the simulation only after the travel time has elapsed. By taking advantage of partitioning power systems using the aforementioned characteristics, the size of the system to be simulated is not limited by the computational power of a single processor. For example, large and complex power systems have been successfully simulated in real-time [29, 30].

The benefits derived from executing power system simulation in real-time are not confined to computational time reduction; more importantly, actual physical devices such as protective relays and control equipment can be connected to real-time simulators in a closed loop manner [31]. Those real physical devices can be tested and evaluated as if they were interacting with real power systems. One application area in which the real-time simulator, RTDS in particular, has been established as a de facto standard for testing is the protective relay development and testing [32]. The input to a protective relay (i.e., system voltage and

current values in most of the testing cases) derives from simulations in real-time. Thus, the testing subject, i.e., a protective relay, is unable to discern whether it is connected to a real power system or to a simulated system. The output from the testing subject is reflected immediately to the simulations, also in real-time, ensuring the truthfulness of the effect caused by the relay operation in the simulations. This true closed loop testing environment allows users of the real-time simulators to evaluate complex protective algorithms involving communication schemes or multiple relays operation coordination [33].

In contrast, when real-time simulation is not possible and off-line EMT simulation is used, conceptualization or modeling of the real physical devices under test becomes inevitable because such devices cannot interact directly with the non real-time simulation by means of the off-line EMT simulation tools. The information regarding the real physical device, which is necessary for the conceptualization, is not always available. In such cases, certain assumptions have to be made when the necessary piece of information is missing. All the aforementioned factors aggravate the gap between the simulation results and the results derived from real system testing. On the contrary, those negative factors accompanying off-line simulations can be overcome with real-time simulation. The real physical system can be interconnected with real-time simulations without any conceptualization or intermediary. In other words, the real physical system can be considered as a black box and the testing can still produce the intended outcomes by using real-time simulations with the test subjects.

1.2 Optimization and simulation

Exploring the optimality of system designs using a formal mathematical approach has been attracting researchers' attention since the early 1970's [34]. In practical applications of an optimization technique, obtaining full mathematical descriptions of the system, either controller or plant, is not always achievable. When such situations occur, running a time simulation of the system and evaluating the objective function would be one readily available solution. An example is the 'fmincon()' functions in MATLAB [19]. The function, 'fmincon()', runs constrained optimizations in order to locate a function minimum using various optimization techniques. If the target for the optimization can be described in a

mathematically closed form, then the function can assume the mathematical description and perform the optimization. Otherwise, when such mathematical descriptions are unavailable, the target system can be described as a simulation model (Simulink model) and the objective function evaluation, necessary for the optimization, can be obtained from the simulation results of the target system model. In this way, a time domain simulation can be used as the necessary objective function evaluation for a given optimization algorithm.

An Objective Function (OF) is a function of the parameters to be optimized. A non-linear optimization (NLO) algorithm can evaluate an optimization candidate through the objective function. Usually the OF is selected as a positive semi-definite function[35] that has a lower value, the closer the results are to the desired objective. A zero value indicates perfect attainment of the objective. The goal of the optimization process is to minimize the OF as much as possible.

The optimization algorithm starts with a set of parameter values, and evaluates the OF. By inspecting the OF, a mathematical algorithm proposes trial parameter values for the next run that are likely to improve the performance, i.e., result in a smaller value of the OF. This iterative process continues, until a predefined exit criterion is satisfied.

Although a large number of NLO algorithms exist, these methods can be classified into three categories: direct search methods, gradient-based methods, and heuristic methods [36]. Direct search methods only use the values of the objective function (obtained at a limited number of sample points) in the search. The next step for the evaluation can be directly determined from these previous evaluations. In other words, the next candidate for the evaluation is generated by selecting the new samples in the vicinity of the points with lower objective function values (assuming a minimization problem). The methods of Nelder-Mead [37], Hooke-Jeeves [38], and Powell's conjugate directions [39] are typical examples. The main advantage of direct search methods is their simplicity, because these methods do not require explicit calculations of derivatives as would be required in gradient-based methods.

Gradient-based methods not only utilize the collection of objective function evaluation values, but they also incorporate derivative information of the function at each evaluation point. Derivatives of a function indicate the local rate of change of the function with respect to its parameters. Thus, selecting an effective and suitable direction for reducing the

objective function can be facilitated by this additional information. Examples of gradient-based methods are Cauchy's method, Newton's method, and Marquardt's method [40].

Heuristic methods are mainly inspired by nature (as opposed to direct and gradient-based methods with mathematical foundations). These methods are mainly used for their capability for getting close to the global optimum of a function. In the past few decades, several heuristic methods have been developed and examples include genetic algorithms (GAs) [41], particle swarm optimization (PSO) [42], simulated annealing (SA) [43], and ant colony optimization algorithms [44]. However, these methods require a relatively large number of samples to provide reasonably accurate results. As computational platforms have advanced and parallel computing techniques, such as multi core processors and GPUs (Graphical Processing Units) are approaching main stream computing [45], the Heuristic-based methods gain momentum in many applications, including OE-EMT (Optimization-enabled electromagnetic transient) analysis.

The incorporation of NLO with EMT simulations utilizes the EMT program to evaluate the OF, hence introducing a unique challenge. In the past, because the numerical derivatives required several simulations, a formal gradient-based approach was not regarded as suitable for OE-EMT analysis. Alternatively, two classes of NLO algorithms which have been used for OE-EMT analysis are as follows: (i) direct search optimization algorithms including the down-hill non-linear Simplex algorithm (non-linear Nelder-Mead Simplex algorithm) and (ii) heuristic optimization algorithms, including GA (i.e., Genetic Algorithm), SA (i.e., Simulated Annealing) and PSO (i.e., Particle Swarm Optimization). Neither of these categories is dependent on the existence of a closed form mathematical description of the problem and thus they offer solution(s) without using the derivatives. As a pre-requisite step prior to the application of the OE-EMT procedure, each of these algorithms was carefully evaluated. Many minor variations in those algorithms, such as applying different sets of controlling parameters were also tested and the final results, including the quality of the results and the total execution time, were compared in order to choose the appropriate algorithms.

1.3 OE-EMT using real-time (RT) simulator

The principal contribution of the research described in this thesis is to apply non-linear optimization to real-time EMT-type simulation. Non-linear optimization (NLO) algorithms have been combined with an off-line EMT-type simulation program in previous research. The combination, off-line EMT-type simulation programs and NLO algorithms, was able to locate the solution to an optimization problem by simulation iterations. In this combination, the offline EMT-type simulation programs evaluate the objective function through time simulations [46, 47]. However, the limitations associated with the combination of the NLO algorithms and the off-line EMT-type simulation program reveal shortcomings, once a real world system becomes involved in the process. The shortcomings of the combination in previous research, namely off-line EMT-type simulation program and NLO algorithms, are as follows:

Shortcoming 1: Modeling inadequacies

Often, the real world device under test, (e.g., a real world controller), will not allow the off-line simulation at all, thus eliminating the applicability of the previous combination, off-line EMT-type simulation program and NLO algorithms. One example of this situation arises when a real controller, manufactured by a commercial company, needs to be described in off-line simulation. Usually, the detailed implementation information regarding the controller is very difficult, if not impossible, to acquire because the manufacturer of the commercial product will not divulge the arithmetic/algorithmic details, attempting to safeguard the information as a trade secret. Therefore, describing the controller in off-line simulation software with sufficient level of detail is often exceedingly difficult.

Shortcoming 2: Challenges in transferring optimized parameters from simulation model to actual hardware

In off-line simulators, the optimized parameters are data in a computer. These must be transferred to the real world. This transfer involves a manual step, thereby increasing the likelihood of human error. Also, the results may be suboptimal because of shortcomings identified above, in that the modeled external hardware may not be exactly equivalent to the physical hardware.

Shortcoming 3: Extreme simulation time

Most off-line simulation programs are slow compared to a real-time implementation. Even with the use of multiple cores in modern computers, the simulation time increases to very large values as the complexity and detail of the represented system increases.

In the light of aforementioned shortcomings, the real-time EMT–type simulation environment with an ample interface capability becomes an attractive option for the application of the NLO algorithm with EMT-type simulation. This new approach uses real-time simulation as the objective function evaluation vehicle in the optimization process. This approach has been developed and is evaluated in this research. This new development extends the practicality of the combination of the NLO algorithm with the EMT-type simulation. Enabling real-time EMT-type simulation with non-linear optimization capability offers several advantages, which are summarized below:

Advantage 1: Direct interface with real world systems

Optimization with real-time simulation can directly interact with the real world system, not with its conceptual representation. By connecting the real world device to the real-time simulation, the device can express its behavior in parallel with the simulation, thus exchanging real input and output signals in synchrony with the simulation. Thus, the non-linear optimization technique in conjunction with real-time simulation capability can

directly change parameters in the real world system. If the real physical device has a storage capability, which would retain the final optimization result, then the device can be removed from the real-time simulator with optimization capability and directly installed in the field without any further measure such as reinterpretation.

Furthermore, this combination does not entail the necessity for information regarding the implementation details or any confidential or proprietary information from the manufacturer. A real world device can be interconnected with real-time simulator 'as-is', in exactly the same way as it was originally intended for. Also, the update of the necessary parameters during the optimization process can be carried out in the same way as when the device is deployed in the real world. For example, if a controller receives the necessary control parameters through a certain means of communication, the optimization program can transmit the necessary information in the same way requested by the real world device.

Advantage 2: Faster execution of the optimization process

The real-time EMT-type simulator can reduce the total amount of simulation time necessary for optimization-enabled electromagnetic simulation. The effective utilization of parallel processin0g enables the real-time simulation environment to shorten the necessary calculation time for the simulation. In one implementation example, RTDS-Real Time Digital Simulator, the network solution which provides the solution for the node voltage from the matrix multiplication between the system admittance matrix and the current injection vector runs on a dedicated processor. The collection of calculation engines, (i.e., a processor card in RTDS), is responsible for calculating the overall network behavior. Different components are assigned to different processors so that their contribution to the subsystem response can be calculated in parallel. In general, RTDS executes real-time simulation from two different types of calculations, performed by the participating processors cards: the network solution (i.e. nodal analysis) and the auxiliary power system or control system components. The network solution solves the node voltages and branch currents based on the network impedances and the contribution of the auxiliary components. Auxiliary components are, for example, transmission lines, transformers and synchronous machines. The components provide admittance matrix overlays and current injections to the

network solution. The combined solution of the network solution and the auxiliary components simulates the overall network response [27].

The capability of the real-time simulation environment can be utilized in two different ways. The first is the faster execution of simulations, resulting from the inherent distributed and parallel characteristics of the simulation environment. This capability offers an opportunity to overcome the slow execution time issue of off-line simulation environments. As simulation cases become more complex, for example due to inclusion of power electronics systems with high switching frequency or a large number of sub modules [48], the execution time expands prohibitively, thereby making multiple runs of the simulation required for the optimization impractical. In contrast, a real-time simulation executes the simulation runs in real-time (or possibly, faster than real-time), and the necessary number of simulation runs can be accommodated in the available amount of time.

The second way that the capability of the real-time simulation environment can be utilized is the parallelism in execution. A typical real-time simulation environment is founded on the principle of parallel processing. By taking advantage of this characteristic, most of the heuristic optimization algorithm implementations can obtain substantial advantage of evaluating several optimization candidates concurrently. In the case of genetic algorithms (GAs), the candidate population assumes a certain size. Therefore, each epoch or evolution step is composed of evaluating the individual candidates in the population. The candidates do not have a cause and effect relationship. This type of algorithm with no cause and effect relationship (i.e., dependencies) among the candidates permits evaluation of those candidates in parallel. Thus, the parallel processing foundation of the real-time simulation environment not only shortens the execution of a single candidate by faster calculations, but also allows savings in evaluation time of the candidate population by utilizing parallelism.

1.4 Research objectives

The previous discussion identifies the advantages to be obtained from an optimization-enabled real-time simulation environment. The objective of this thesis is to develop, for the first time, such a simulation environment for the design of controllers for power systems. The objectives are as follows:

1.4.1 To construct a methodology for optimization with a real controller in loop

As previously discussed, the real-time EMT-type simulation environment has the potential to extend the usefulness of the OE-EMT capability. A certain level of disparity is inevitably introduced when the controller or system optimization problems impose another layer of abstraction between the real world and the simulation world. The level of disparity defies any guarantee that the same set of optimized parameters will work in exactly the same way as in the optimization through the off-line simulation.

Furthermore, certain circumstances associated with the origin of the controller or hardware will not allow for the abstraction of the real physical device. One example is the unwillingness of a manufacturer to reveal any trade secrets. The real-time EMT simulation with optimization capability proposed in this thesis assumes no artificial efforts like the abstraction process, or the availability of the implementation specific information of the optimization subject. The ability to interconnect real physical controllers with the simulation environment eliminates the requirement to model the devices in the simulation. Once interconnected with a real-time EMT simulation environment, the real physical device can run in exactly the same environment in terms of the inputs and outputs as in the real world. This process allows the real physical controller to be optimized without any additional process of conceptualization of controllers or re-interpretation of the optimization results. In addition to the convenience of not requiring conceptualization during the optimization, the proposed method does not require the confidential information, which the manufacturer may not be willing to provide. Once the optimization process is finished, the device is ready to be deployed in the real field with no additional measure.

Simply speaking, the optimization subject can be considered as a black box, but still the optimization can proceed. This direct optimization of the external hardware became possible as the real-time EMT simulation environment offered the same optimization facility as was already available in the off-line OE-EMT software. This thesis presents the design and implementation of the real-time EMT simulation environment. The effectiveness of the

development was evaluated with real physical controllers and different optimization algorithms in the research and the results are presented in this thesis.

1.4.2 To improve the performance of the optimization process

As a target system expands or becomes more complex, additional time is usually required for simulation and for optimization using the simulation results. This situation is aggravated as the required iteration count becomes increasingly larger for optimization as a target system size grows. When the optimization goal is very narrow or when a target has to be optimized globally instead of being optimized in certain regions in the parameter space, the number of iterations, and subsequently the total simulation time expands significantly.

The expected total evaluation time for the optimization application can be improved in two ways: First, each individual evaluation can be expedited by utilizing the parallel processing power of the real-time digital simulation platform. Second, many heuristic optimization algorithms offer the possibility of parallel evaluations of the necessary candidates during the optimization process. This characteristic of the algorithms allows a parallel evaluation using the hardware parallelism, resulting in total execution time reduction.

This thesis presents the effectiveness of the developed simulation environment in saving the total evaluation time in executing optimization algorithms with real-time simulation. The possibility of saving in both approaches, (i.e., faster execution and utilization of the parallelism), are explored and the results are reported.

1.5 Organization of this thesis

Chapter 1 has provided an introduction to the time domain simulation of power system. The introduction of the building components in the research followed the first introduction. This introduction included real-time digital simulation, optimization algorithms and the combination of both-the OE-EMT applications using real-time simulations. This outlined the foundation for the research.

In Chapter 2, the basic theory of power system modeling is introduced. The beginning of the chapter is composed of a brief explanation of digital time domain simulation in power system studies. The oldest techniques, transient network analyzer and analog simulator are explained first. Then, explanations of the transient stability program and the Eigen-value analysis based method follow. The chapter finishes with the explanation of the Electro-Magnetic Transient (EMT) method.

Chapter 3 presents a discussion of non-linear optimization algorithms. The beginning part is concerned with direct optimization methods. The Nelder-Mead downhill Simplex method is selected and is explained as a typical algorithm in this category. The explanation of the heuristics-based algorithms follows. This explanation includes simulated annealing, genetic algorithms and particle swarm algorithm.

Chapter 4 describes how to incorporate non-linear optimization algorithms with EMT-type simulation. First, the method for laying down the bridge between an off-line EMT type simulation program and a non-linear optimization algorithm is explained. The method is exemplified by a widely used off-line EMT type simulation software, namely PSCAD (Power System Computer Aided Design). Then, the discussion moves onto the combination of non-linear optimization algorithms and a real-time EMT type simulation environment. This method for the incorporation is exemplified with a well-known real-time EMT type simulation environment, namely RTDS (Real Time Digital Simulator).

In Chapter 5, the entire process of the real-time optimization (explained in the previous chapter) is presented by means of case studies. The effectiveness of the real-time optimization is reviewed by analyzing the data from the case study results.

Chapter 6 presents the application of the developed approach in HILS (Hardware in loop simulation). Several example studies and their results are presented.

Chapter 7, provides conclusions of the research and presents suggestions for further research.

2 POWER SYSTEM MODELING METHODS

A certain device in power system simulation can be modeled in many different ways. The source of diversity can be attributed to the difference in the modeling requirements. Each method of modeling has its own advantages and disadvantages. Usually each modeling method addresses a specific aspect of behavior in the subject device. Hence, the output from the selected modeling method for a certain device is usually valid only in an intended application. For instance, modeling a synchronous machine can be performed at many different levels: load flow analysis level, transient stability analysis level, electromagnetic transient (EMT) analysis level and electromagnetic field analysis level (mostly with a finite element method (FEM) type approach) [49]. Sometimes, a real, miniature model (i.e. a scale d down model) of a real synchronous machine might be necessary in order to meet certain analysis requirements, such as an absolute real-time response from the model, in conjunction with an analog simulator. Much effort and resources must be committed to the modeling process of the subject device in order to achieve the necessary fidelity of the model in such a case.

Each analysis level entails different requirements for the input parameters as well as the necessary accuracy of those input parameters. Thus, a clear understanding of the available modeling methods is one of the important foundations for building an effective simulation case. This understanding enables the simulation engineer to select the simplest feasible model which still meets all the representation details required for a specific study.

2.1 Time domain simulation in power system studies

Power systems, which are the subject of analysis, are large and highly dynamic systems by nature. Dynamic interactions between multiple pieces of equipment are critical to the power system. Thus, steady state analysis methods like power-flow analysis, which are adequate for determining transmission line loading, are often not sufficient for studying power system transients. Furthermore, many devices in the study subject present nonlinear characteristics, which a traditional, linear system analysis based approach is unable to account for the up to

the desired accuracy. On the other hand, many transient studies cover different time-scales. For instance, commutation failure occurs in an HVDC (High Voltage Direct Current) system in a millisecond; a voltage collapse may involve induction machine stalling, which happens in a couple of seconds; transformer tap change with a time frame of tens of seconds; Complex interactions are possible among these behaviors. With this complexity, the time domain simulation becomes the only fully comprehensive way to do a proper analysis of multi-scale power system transients [50].

Before the advent of high speed digital processing technologies, such as DSPs (Digital Signal Processors), the only practical and accurate way to execute time domain simulation was based on the analog simulator, or TNA (Transient Network Analyzer). A typical analog simulator is composed of scaled down real power system devices, such as miniature generators and miniature transformers. At the same time, the necessary interconnection among those devices is the exact replica of a real system [5]; therefore, the entire system, including the scaled down devices and the interconnections between them, is able to simulate the subject of study in real- time. However, as the digital time domain tools became more versatile and accurate, the role of analog simulators was gradually superseded by their digital descendants.

Digital time domain simulation tools have been extensively used in power system studies since the late 1960s. In many areas, digital time domain simulations have become the industry standard tools. In digital time domain simulations, the nonlinear differential equations describing the behavior of power system devices are solved by numerical integration techniques. By choosing appropriate models, algorithms and time-step sizes, power system behaviors can be well predicted, studied and replicated.

There are two major types of digital time domain simulation tools in power system studies. They have been developed separately since the 1960s. One is the electromechanical transient simulation, or 'Transient Stability Analysis (TSA)'. This type of simulation tool is mainly used for predicting whether a large power system can regain an acceptable equilibrium point (stable and secure considering rotor angle) after being subjected to major disturbances [51]. In such studies, the dynamic behaviors of large inertia power system components and the dynamic interactions among them, which cause large energy flow variations, are of main concern. In such methods, the major part of the electrical network

can be represented as a quasi-steady state phasor model, with the large inertia components being represented with time domain differential equations.

The other type is the electromagnetic transient simulation referred to as EMT simulation. It focuses on emulating the detailed behaviors of the components of the power system, such as voltage spikes, current surges, voltage and current waveform distortions and harmonics. As power electronic devices are widely applied in modern power systems, simulating the detailed semiconductor switching transients has become one of the major applications of EMT simulation tools [52].

These types of simulations will be discussed in more detail hereafter.

2.2 Network analyzers and analog simulator

Transient Network Analyzer (TNA), an analog model of a power system, is a tool that had been used for many years in order to analyze power system behavior in real-time. The TNA is a legacy tool for the analysis of power system phenomena. The use of TNA predates digital EMT-type simulation programs. The TNA implements an analog model of a power system using small scale system components such as generators, transformers with response proportional to those of actual sizes [53, 54]. Transmission lines are generally modeled by an appropriate number of pi sections, circuit breakers by electrical switches; as well, transformers and reactors are modeled by small scale magnetic circuits adjusted to have electromagnetic characteristics similar to those of their full scale counterparts in the real world. The individual component models were configured and interconnected to form the system model for study [55]. An electrical system is studied by applying the actual disturbances and measuring the resulting currents and voltages in the simulator [56]. As the system is built with real, although scaled down components, the TNA is automatically a real-time implementation.

However, the scales of power system networks are so large and the structures and components are so complicated, that to build and maintain even a scaled down physical model of a power system with appropriate accuracy is prohibitively difficult and expensive. In addition, the poor replicability of the simulation results due to the aging of the discrete

components as well as the difficulty in matching exact component values as used in previous experiments has lead to reduced use of this simulation technology in practice [57, 58].

2.3 Transient stability program

An important study requirement in power systems is that of assessing transient stability [51]. The concern in this case is to determine whether generators in the network all run at the same average frequency and whether relative rotor angles of all the generator rotors settle quickly to their constant steady state values. Transient stability tools typically model the electric network using a simplified phasor representation and concentrate on accurate dynamic representation of the mechanical quantities such as rotor angle and speed. Modern TSA (Transient Stability Analysis) simulation tools such as PSS/e [59], TSAT [60] (Transient Security Assessment Tool from Powertech^{tm}) and the BPA stability programs are widely used in the power industry by power system operators and by planners in particular. These tools have become the de facto industry standard for studying power system transient stability problems. They have been employed for voltage stability studies, small signal stability studies and frequency stability studies as well. The phenomena from D to F in Figure 2.3-1, especially phenomena D and E, are the main focus of these tools. Practical power systems with tens of thousands of bus-bars and hundreds of generators can readily be handled by such tools. For example, one of the largest power system models ever created was made for the NERC (North American Electric Reliability Corporation), representing eastern North America interconnected system. This model contains more than 40000 ac bus-bars, more than 5000 generators and 15 HVDC links, and the model keeps growing. A simulation of 15 seconds with this extremely large model on TSAT can be finished in around 10 minutes, using a 2 GHz PENTIUM 4 PC. Some engineering practices have been established and widely accepted in regard to the application of these type of tools [61]. For instance, the transients of the network are not considered. The voltage and current transients of the network are assumed to decay instantaneously. As a result, the network can be represented by algebraic equations instead of by differential equations. The frequency of the system, which is the subject of analysis using those tools, is assumed to be close to the nominal frequency (i.e. fundamental frequency). The network is represented by an

admittance matrix, which only preserves the characteristics of the network at the fundamental frequency.

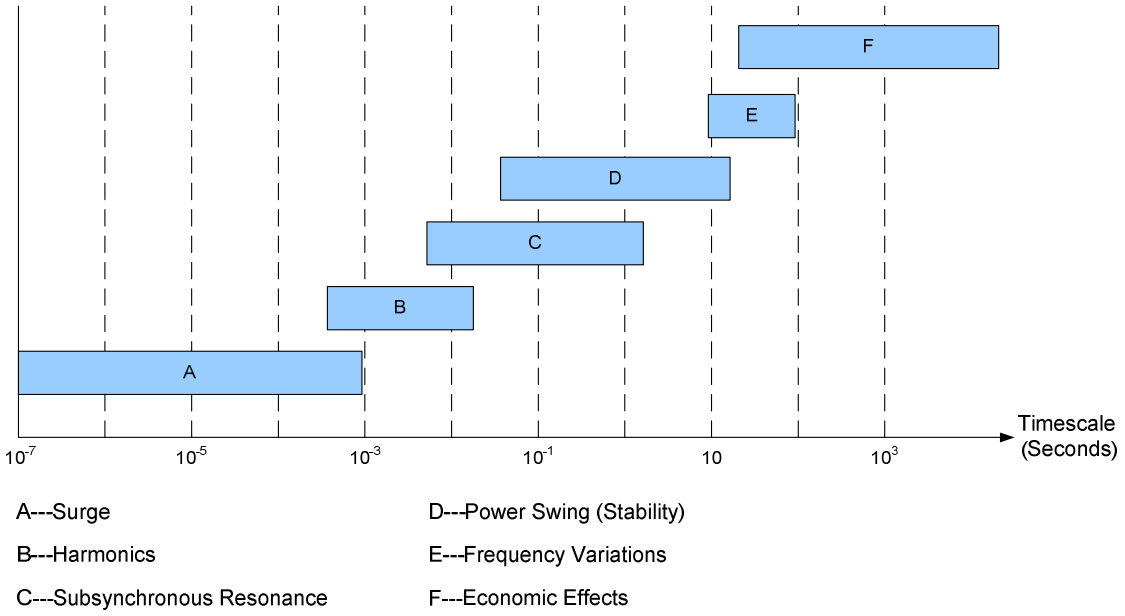

Figure 2.3-1 Time Frames of Various Power System Phenomena

2.4 Eigen-value analysis based program

One power system analysis technique that is closely related to the Eigen-value analysis is small signal stability analysis [62]. In general, power system stability is defined as the system's ability, under a given initial operating condition, to regain a state of operating equilibrium after being subjected to physical disturbances. In particular, small-signal stability analysis involves studying the stability of a power system under small disturbances. If the disturbances permit linearization of the dynamic equations of the power system for the purpose of analysis, then they are classified as small disturbances. In other words, the small disturbance would initiate system reaction in which the operating points of the system do not experience large excursion from their pre-disturbance steady state. Small changes in load or power generation, change in the voltage reference of the excitation system as well as tripping of transmission lines carrying a small load are examples of small disturbances. A small-signal study can be undertaken to identify and solve many types of problems in power

systems, such as rotor angle stability, sub-synchronous resonance, controller interaction, and stability of the power system due to power electronic devices.

For the purpose of small-signal stability analysis, the system can be linearized around an operating point in order to yield a set of linear differential equations, as follows:

$$\dot{x}(t) = A\,x(t) \qquad (2.1)$$

where x(t) is a state vector and A is called the state transition matrix. If the size of the state vector is n, then the size of the A matrix is n by n. The Eigen analysis of the matrix A produces n Eigenvalues and their associated Eigenvectors. The Eigenvalues and the Eigenvectors together determine the dynamics of the analysis subject. Therefore, the dynamic characteristics of the analysis subject can be understood further by proper application of the small signal stability analysis technique.

Voltage stability study is another power system analysis technique where the Eigen-value analysis is used as a analysis tool. Voltage stability refers to the ability of a power system to maintain steady voltage at all buses in the system after being subjected to a disturbance from a given initial operating condition. It depends on the ability to maintain/restore equilibrium between load demand and load supply in the power system. For large disturbances, the only method for voltage stability analysis is to perform a time domain simulation with all relevant models. However, if the size of the disturbances is small enough, a linearized model developed in a similar way to the small signal stability analysis can be utilized. This type of study is referred to as 'small-disturbance or small-signal voltage stability' in [61]. This type of study becomes relevant when the load in the subject power system changes gradually or the parameters of the system change smoothly.

2.5 Electro-Magnetic Transient (EMT) program

For decades, Electro-Magnetic Transient (EMT) programs have been the industry standard for studying the detailed transient behaviors of power system equipment, especially power electronic equipment, and small power networks. Of all the power system analysis tools, the

EMT programs incorporate the greatest detail, including electromagnetic as well as electromechanical phenomena. The application areas for EMT simulation tools include insulation co-ordinations, study of over-voltages due to switching surges, power electronic transient performance, sub-synchronous resonances and ferro-resonance phenomena. The primary focus of the EMT programs is the areas A to C in Figure 2.3-1 [54], although the models in the programs can reproduce and simulate all phenomena in areas D to F as well. EMT simulation techniques model every phase of the power system network in an independent way. In other words, the EMT simulation does not assume the existence of three phase system as is done in transient stability programs. Thus, all unbalanced components and behaviors can be modeled and simulated. Arbitrary number of phases can be included in the simulation. Indeed, the modeling capabilities of the most popular EMT programs such as EMTP (ATP) [63, 64], EMTDC, and NETOMAC [65, 66] are able to represent power systems accurately. Digital simulation software relies on mathematical models to represent the individual power system components, whereby the user is able to interconnect these models to form the overall power system model for the study.

One of the most common methods for representing power systems mathematically in electromagnetic transient power system simulation software is the EMT-type simulation algorithm (i.e. Dommel's Algorithm [6]). In the EMT-type simulation algorithm, the trapezoidal rule of integration is used to convert integral equations, resulting from nodal analysis of the power system, into algebraic equations. The conversion process based on the trapezoidal integration is stability-preserving for linear systems [49]. That stability means that a large time step will not cause numerical instability (i.e. numerical blow out) during the integration, though the results might lose accuracy. Applying the trapezoidal integration rule, the differential equation of a linear component (e.g., inductor, capacitor) is represented by a constant conductance in parallel with a history current term. The value of the history current term is determined only by past (history) information. Figure 2.5-1 demonstrates the trapezoidal rule application to an inductor [6]. The set of equations describing the entire system is built by using the standard nodal approach [67].

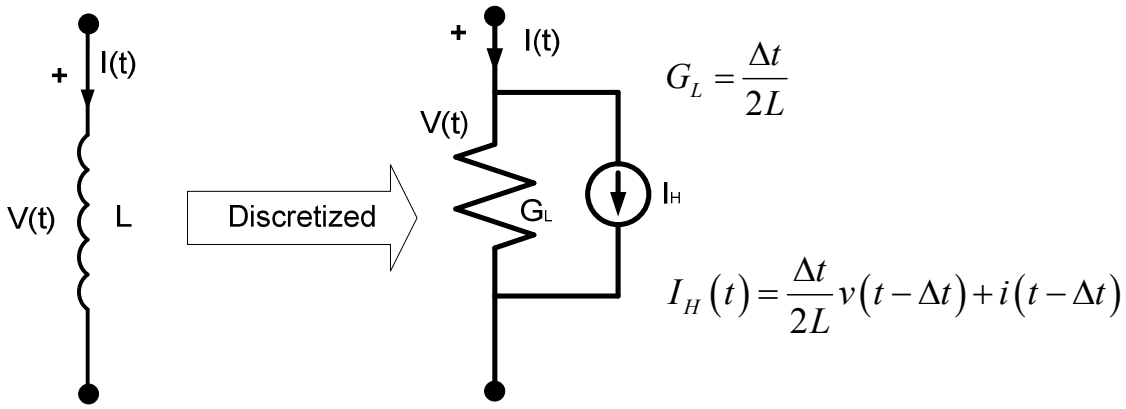

Figure 2.5-1 Inductor and its discretized representation (valid for one time step)

The application of the trapezoidal integration rule requires that the solution be computed only at discrete instants in time. The time between computed instants is known as the time step and is denoted by Δt. Typically, for most power system transient phenomena, the time step can be chosen to be in the order of 10 to 75 microseconds. All of the equations representing power system models must be computed in each time step. Large or complex power system simulations usually require many seconds to finish all the calculations within a time step in non real-time simulation programs.

3 NON-LINEAR OPTIMIZATION METHODS

3.1 Introduction

Electromagnetic transient simulation tools, such as PSCAD and RTDS, were utilized traditionally for reproducing the system behavior for a given static set of system parameters. In modern system designs, it becomes necessary to repeat several simulations with differing parameter values in order to optimize the system design or performance. Hence, there has been considerable recent work [46, 47, 68-73] in using non-linear optimization (NLO) algorithms to control instances of EMT runs, in which the parameters are automatically adjusted with a view to obtaining the best result. This section presents a brief description of such non-linear optimization algorithms. In all these methods, a desired 'performance' is quantified using an objective function 'f'. The function is normally selected to be positive semi-definite with the absolute satisfaction of the desired objective signified by 'f()=0' [47].

The optimization methods devised for the solution of multi- input variables problems can be categorized into four broad groups based on the type of information supplied by the user [40]:

i) Direct search methods, employing only the objective function value;

ii) Gradient methods, requiring accurate values of the first derivatives of the objective function;

iii) Second order methods, which, in addition to the first derivatives, also employ the second derivative of the objective function;

iv) Heuristics-based methods, requiring only the objective function value.

If a mathematical description of a system requiring optimized parameters is well defined and a mathematically closed objective function is available, the optimization through iterative process of the OE-EMT application need not be pursued. The outcome of the optimization algorithm and the associated analysis (e.g. the sensitivity analysis) can be obtained using a simple analytical approach such as setting the gradient of the objective

function to zero. On the other hand, the mathematical description of a system given in closed form may be unattainable in certain circumstances. If the optimization target system becomes too large or too complex, creating the necessary mathematical descriptive equation becomes impractical. Under such circumstances, one must choose an alternative approach. The alternative is utilization of non-linear optimization methods that do not require the closed form mathematical description. Because a closed form solution is unavailable, the objective function value must be obtained from an EMT simulation run. If the alternative choice for the optimization involves the use of heuristics-based optimization algorithms, a large number of objective function evaluations, and hence a large number of simulations, is anticipated. The large number of iterations requires long simulation execution time before a certain termination condition is met and the final optimization result becomes available. This aspect of the optimization process can be expedited by the use of real-time simulation environments, such as RTDS, due to their inherent higher calculation speed compared with off-line simulation tools. In this thesis, the down-hill non-linear Simplex algorithm (i.e., non-linear Nelder-Mead Simplex algorithm) from the direct search method category as well, the Genetic algorithm (GA) and the Particle Swarm Optimization (PSO) algorithm were selected and evaluated in conjunction with the real-time simulation environment in order to incorporate the OE-EMT method with the real-time simulation environment.

3.2 Down-hill Simplex algorithm of Nelder and Mead

The down-hill Simplex optimization algorithm of Nelder and Mead [37] is a geometry-based, non-linear optimization algorithm, independent of any explicit mathematical description of a system. This method facilitates searching for an optimum by the repetitive re-shaping of an object called a “Simplex”. In *N*-dimensional space, a Simplex is a geometric object made up of N+1 vertices. When all geometrical distances between any two vertices are equal, the Simplex is called a regular Simplex. In two and three-dimensional spaces, the Simplex will be a triangle and a tetrahedron, respectively. To understand the idea underlying the Simplex optimization method, consider a two dimensional Simplex (i.e., triangle), as presented in Figure 3.2-1 [46]:

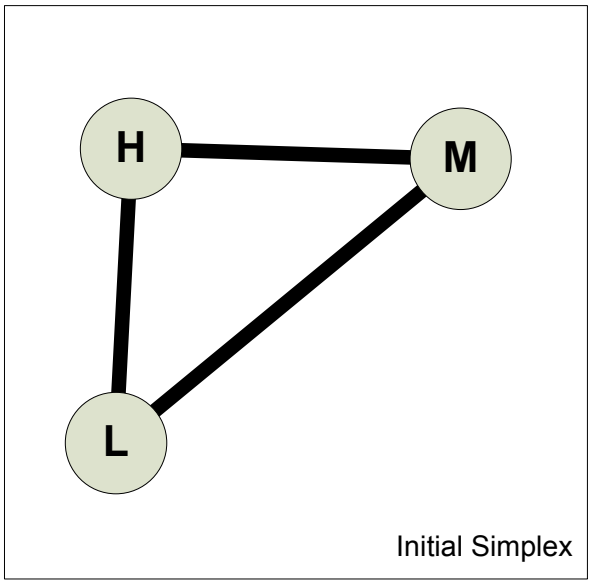

Figure 3.2-1 Two dimensional Simplex

The objective function is evaluated (i.e., by simulation in the case of optimization enabled EMT) at each of the vertices of the Simplex (triangle). The Simplex nodes are labeled high (H) to low (L) according to their corresponding objective function evaluation values. 'High' (H) label is for the node with the highest (i.e., worst) objective function evaluation output, while 'Low' (L) is for the node with the lowest output and 'Middle' (M) is in between them (H and L). The purpose of the operation is to move the simplex away from the vertex with the highest value (i.e., worst value), towards the smaller value. To do so, the high point (H) is reflected through the centroid of the face of the Simplex right across from it in order to generate a vertex with a hopefully lower objective function evaluation value than the current highest vertex in object function evaluation. This operation is called reflection and it is represented in Figure 3.2-2.

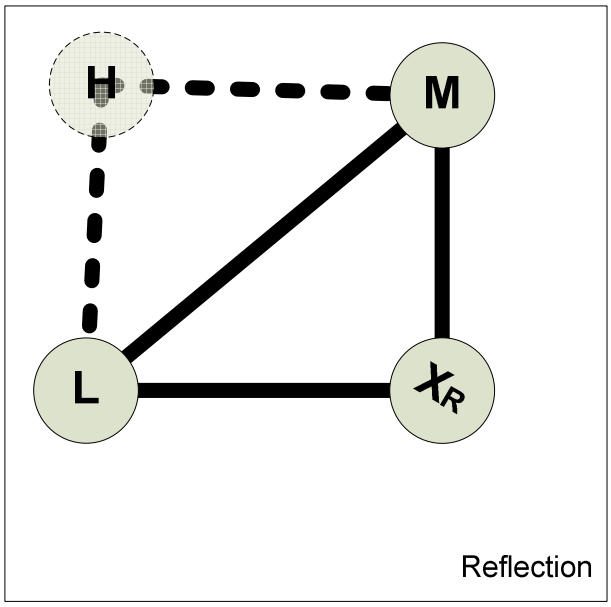

Figure 3.2-2 Reflection

If the newly generated vertex produces a lower (i.e., better) objective function evaluation value than the current lowest point (i.e. with a lower objective function evaluation in a minimization problem), the exploration in the same direction appears favorable. Then, an even larger step is taken in the same direction. This operation is called expansion and it is depicted in Figure 3.2-3.

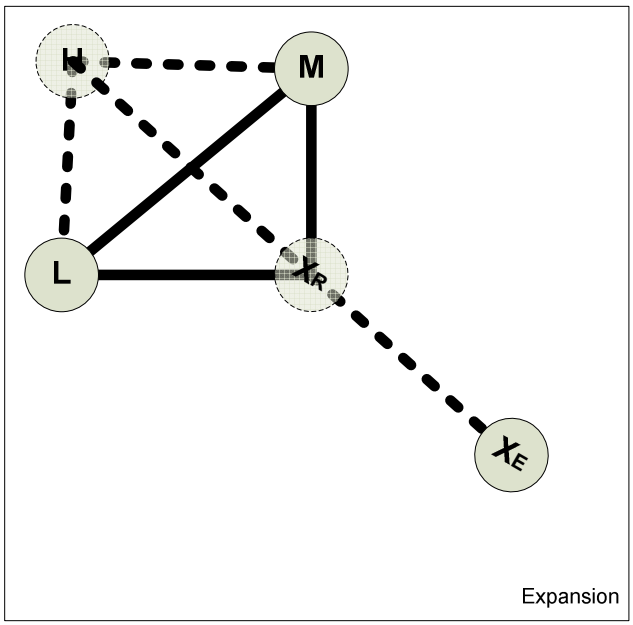

Figure 3.2-3 Expansion (when $f(X_R)$ is lower than f(L))

If the reflected point results in a higher (i.e., worse) objective function evaluation value than the middle vertex (L), but still lower than highest vertex (H) in evaluation, a contraction is ordered.

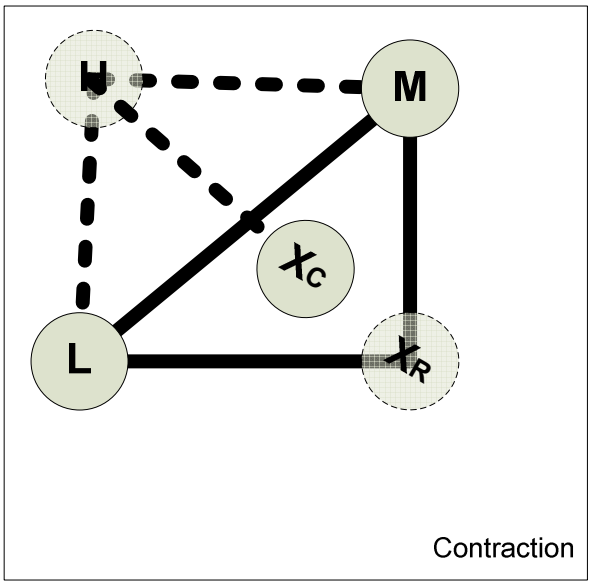

Figure 3.2-4 Contraction (when $f(X_R)$ is higher than f(M) and lower than f(M))

Another type of contraction results when the first trial of the contraction fails to generate a better point than the current worst vertex (i.e., the vertex with highest objective function evaluation value). In that case, a contraction without reflection is performed. In other words, the further contraction from the previously contracted vertex is tried with the purpose of improving the objective function evaluation results. Figure 3.2-5, illustrates this contraction operation.

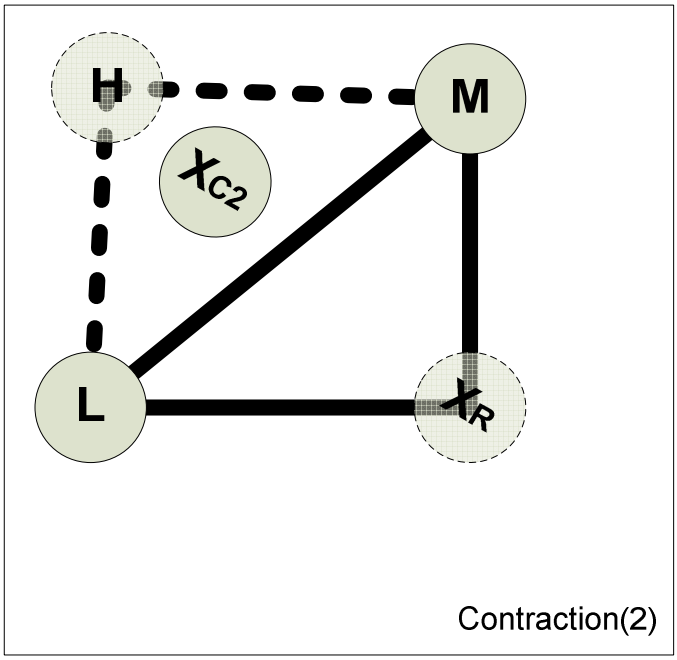

Figure 3.2-5 Contraction without reflection

Successive operations of reflection, expansion and contraction will cause the original Simplex to move towards points with lower (i.e., more desirable) objective function evaluation results. As an analogy, the Simplex is seen to 'tumble down hill' on the contour plot of the objective function in parameter space. In this method, the data pertinent to a Simplex should always be available to the optimization algorithm (i.e., N+1 point (vertex) coordinates plus their corresponding objective function evaluations). The most attractive feature of the Simplex method is the simplicity of its underlying idea, which has made the method easy to use, yet powerful and reliable when applied to optimization problems. It maintains the some sense of steepest descent without requiring the calculation of derivatives. Indeed, moving to the next (improved) point normally requires just one function evaluation.

The Simplex algorithm always results in improved performance, but cannot be guaranteed to reach the global optimum. Depending on the initial Simplex, it might move towards and converge to local optima. In such cases, the algorithm should be restarted with a new side length. Doing so will cause this Simplex method to search for other optima as well [47]. Other means of mitigating the limitation were sought after as well in [74, 75]. Unlike the heuristics-based optimization algorithms, which will be introduced later, this Simplex algorithm offers almost no opportunity to apply the parallelism in the process. This characteristic comes from the fact that all the necessary evaluations of objective functions occur in a sequential manner. This limitation of the algorithm is compensated for by the relatively rapid convergence speed of the search process.

3.3 Genetic Algorithms

Genetic Algorithms (GAs), as well as other optimization methods such as Ant Colony, Particles Swarm and Simulated Annealing optimization are considered to be components of what is referred to as Evolutionary Computing. These optimization algorithms are designed to mimic behaviors of simple biological functions or physical processes. Four well-known paradigms currently exist in the genetics-inspired branch of evolutionary computing [42]: genetic algorithms, evolutionary programming, evolution strategies, and genetic programming. Genetic algorithms are a class of evolutionary computing algorithm that begin with a population of randomly generated candidates that "evolve" towards a solution by applying "genetic" operators, modeled on genetic processes occurring in nature [76]. The cell division (or mutation) in living specimens, which is known to result in healthier (i.e., optimized) future generations under evolutionary pressure, is an appropriate analogy. In order to achieve the same effect as in the evolution process of the nature, one constructs virtual chromosomes, which embed the signature of the individual, as occurs in the real world counterpart [77]. The main application of GAs in power systems is the solution of optimization problems involving integers, in which the parameters to be optimized assume integer (i.e. fixed point) values.

One power system area where GAs are extensively utilized is in economic dispatch [78-80]. In such cases, the nature of the problem can be intuitively captured by the element of

the GA, a chromosome. Similar to the down-hill Simplex method, the core concept in GAs is also based on intuitive interpretations rather than on formal mathematics. The underlying idea, originally derived from nature, evolved from observing how the evolution of biological creatures derives from their constituent DNA and chromosomes. Thus, a simple analogy can be made with a mathematical problem comprised of many parameters. Each parameter assumes the place of a chromosome in the mathematical analogy of a real chemical sequence. The Genetic algorithms (GAs) are, in fact, attempting to capture mathematically the process of evolution in nature. The fundamental concept of genetic algorithms is depicted in the following flow chart in Figure 3.3-1:

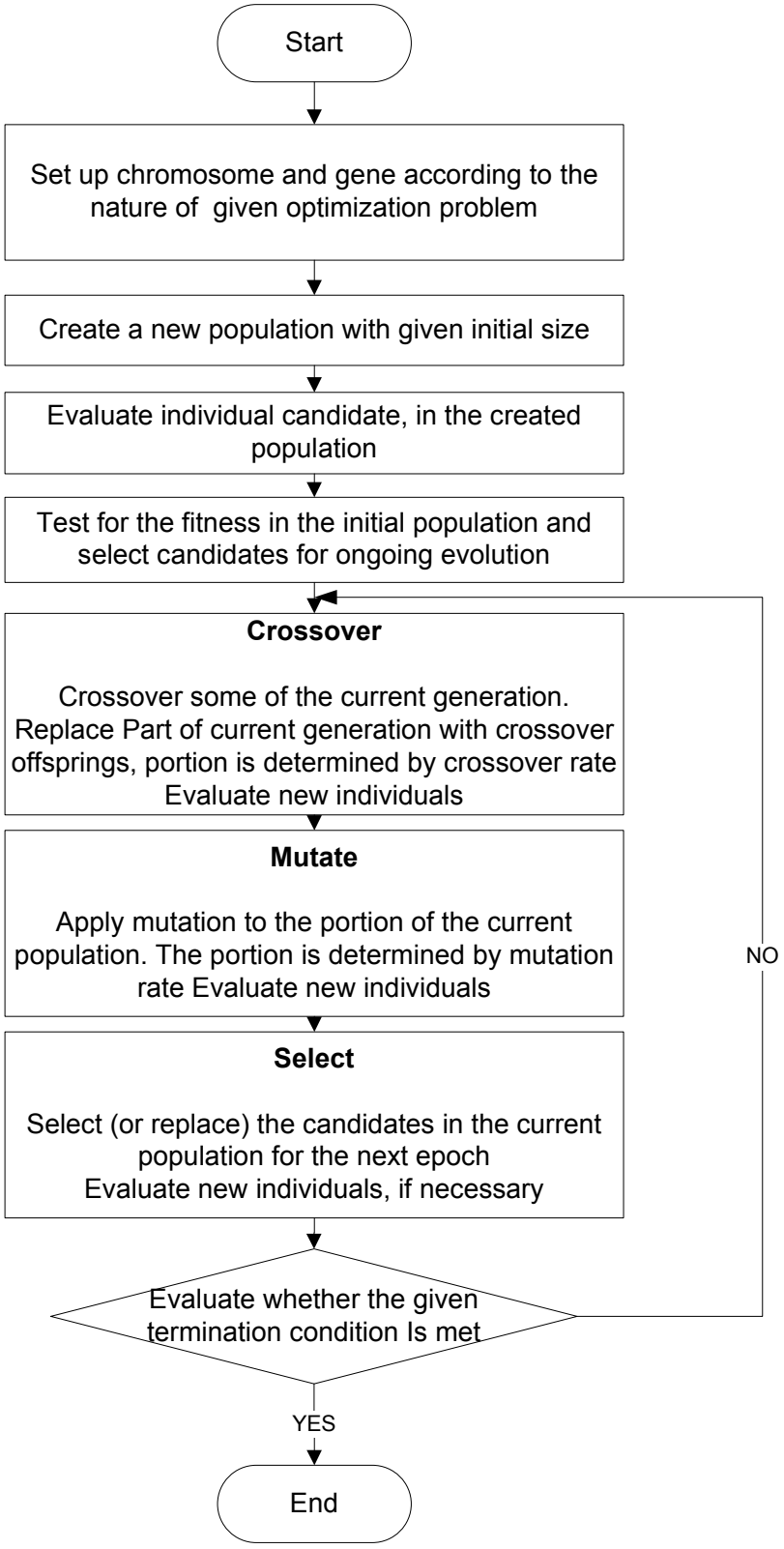

Figure 3.3-1 Flow chart of genetic algorithms process

GAs are useful for avoiding convergence to local optima. However, setting appropriate convergence criteria or terminating condition is not straightforward in the practical aspect of this algorithm. Furthermore, without a properly coordinated crossover rate and mutation rate, the performance of the optimization algorithm tends to degrade as the algorithm proceeds and the number of generations increases, ultimately resulting in a substantial increase in objective evaluations. In order to overcome the degradation of the results, GAs are frequently used in conjunction with other optimization algorithm(s) [81-83].

An implementation of GAs for a particular problem must include the following five elements [76]:

(i) A genetic representation of solutions to the problem

(ii) A system for creating an initial population of solutions

(iii) An evaluation function that substitute for the environment (or nature), rating solutions in terms of their "fitness"

(iv) Genetic operators that alter the composition of offspring during reproduction

(v) Values for the parameters that the algorithms use (e.g., population size, probabilities of applying genetic operators)

My implementation of the genetic algorithm is derived from an open source heuristics-based optimization framework named 'AForge' [84].

In my implementation, each of the aforementioned five components was designed in the following manner:

(i) As the parameters to be optimized are usually floating point numbers, the genetic representation (i.e., chromosome) design is planned for the purpose of preserving the details of the floating point number representation [85]. Each parameter in a single chromosome occupies a bit field. Hence, a floating point number parameter occupies a 64-bit word in a chromosome. For example, if the optimization problem has two variables x_1 and x_2 (i.e., floating point numbers), a chromosome would have two 64-bit length words. Each word would represent the

corresponding parameter in double precision. Figure 3.3-2 illustrates this composition of such a chromosome. The EMT simulation finally evaluates the OF, using 64-bit pieces of this chromosome as the trial parameters.

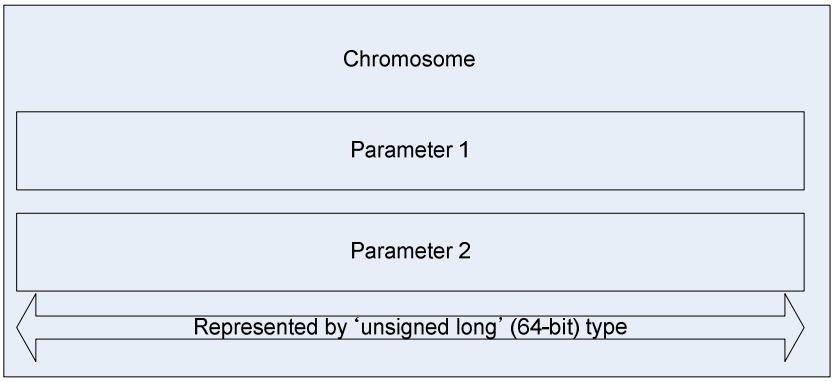

Figure 3.3-2 Chromosome design

(ii) The initial population (e.g., 100 individual chromosomes) is created randomly within the predefined solution boundary.

(iii) The objective function, which is based on EMT simulation, evaluates the fitness of the individual candidate in a population by conducting a simulation run using 64-bit pieces of these as input parameters. The evaluation results, (i.e., the fitness value), are passed to the 'parent selection' algorithm that selects parents for the next generation individuals.

(iv) The GA implementation has three genetic operators: crossover, mutation and selection [86].

The crossover operator creates offspring in the next generation from the selected parents. Figure 3.3-3 presents the implementation of the crossover operator in my GA implementation.

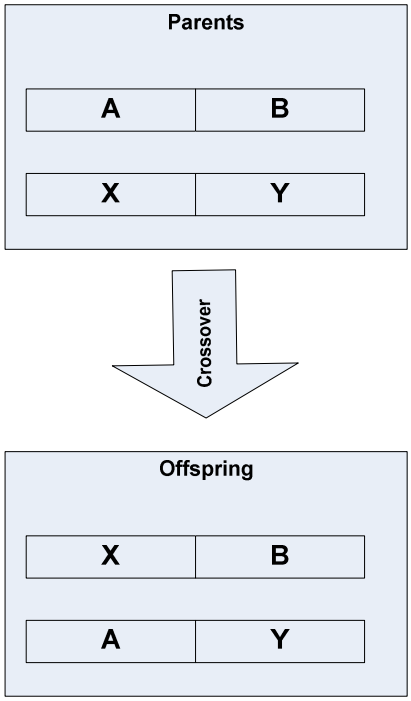

Figure 3.3-3 Operation of crossover operator

The mutation operator introduces randomness in a certain number of chromosomes when the next generation is created. The randomness is introduced into chromosomes in the population by imposing some changes.

The selection operator chooses the more desirable candidate for the parents of the next generation. The three popular selection algorithms [87] implemented and compared were as follows:

a. The elite selection algorithm

The candidates in the current population are sorted in the order of their fitness values. In other words, the fittest candidate will be the first and the least fit candidate will be the last in a row. Then, according to the random selection rate parameters, the worst part of the population is cut off and replaced with new candidates. For example, if the current population size is 50 and the random selection rate is 0.1, then the worst 5 (i.e., 50 * 0.1) will be cut off and replaced with new members.

b. Rank selection algorithm

The possibility of being chosen for the next generation is proportional to the rank of the candidate in the current population. A simple example is presented hereafter: if there are 5 candidates with distinct fitness values, then the fittest candidate should have 5 times more chance than the least fit candidate to be selected as a member of the next generation. This probability relationship translates into as selection probability of 33.33% for the fittest candidate. The following calculation presents the probability assignment process:

Number of candidates in the current population: 5
The chance assigned to the worst (5^{th} individual): 1
The chance assigned to the next worst (4^{th} individual): 2
The chance assigned to the next worst (3^{rd} individual): 3
The chance assigned to the next worst (2^{nd} individual): 4
The chance assigned to the next worst or the best (1^{st} individual): 5
Total amount of chance for being selected: 15 = 1 + 2 + 3 + 4 + 5

At the end, the selection probability assigned to each of those 5 candidates is represented in the following list:

1^{st} (fittest): 33.33 % = 5 / 15
2^{nd} : 26.7 % = 4 / 15
3^{rd} : 20.0 % = 3 / 15
4^{th} : 13.3 % = 2 / 15
5^{th} (worst) : 6.7 % = 1 / 15

c. Roulette selection algorithm.

This selection scheme is similar to a roulette wheel in a casino. The area on the roulette wheel is assigned to a candidate according to its fitness value. Thus, if a candidate occupies a larger area in the wheel, the candidate stands a better chance of being chosen as a member

of the next generation. The following calculation example presents how to calculate the corresponding area for a candidate [88]. Assume that the current population includes 5 individuals. For the sake of ease in the explanation, the optimization objective in this example is maximization, not minimization. Each of the individuals has the following fitness values:

1^{st} (fittest) : 8.48
2^{nd} : 6.82
3^{rd} : 3.08
4^{th} : 2.57
5^{th} (worst) : 1.11
The fitness value total is 22.05.

With this total value and each fitness value, we can calculate the individual's area on the roulette wheel as the following:

1^{st} (fittest) : 8.48/22.05 = 38%
2^{nd} : 6.82/22.05 = 31%
3^{rd} : 3.08/22.05 = 14%
4^{th} : 2.57/22.05 = 12%
5^{th} (worst) : 1.11/22.05 = 5%

As with the real roulette game in a casino, the larger an area an individual occupies on the wheel, the greater is the chance for the individual to be selected.

(v) The parameters associated with my genetic algorithm implementation are the following four:

The first parameter associated with the implementation is the crossover rate. This parameter determines how large a portion of population would be replaced with the offspring individuals resulting from the application of the cross over operator.

The second parameter is mutation rate. This parameter determines the rate of the mutation operator application. For example, if this rate is 0.1, then 10% of the current population would be subject to the mutation.

The third parameter is random selection rate. This parameter only applies to one of the selection algorithms: elite selection. The elite selection algorithm will replace part of population with new members according to this parameter. For example, if this parameter is 0.05, then the worst 5% of the population would be replaced with new members. The fourth parameter is the choice of selection algorithm under the selection operator.

3.3.1 Steps in GA implementation – a simple example

A simple example presented herein explains the details of the algorithm implementation further. Let us assume that the minimum value of a simple two variable function is sought after by execution of the genetic algorithm implementation. The aforementioned five elements of the algorithm need to be designed and implemented in order to apply the algorithm to the problem (function minimization).

Chromosome design

The genetic representation of the problem can be rephrased as the necessary design of the chromosome. Because the two input variables of the function are in floating point format, the chromosome design must be capable of accommodating the characteristics of the number format. In most computer systems, the double precision number (64-bit length in total) representation is employed as the binary representation of the floating point number. Thus, expressing the full extent of double precision floating point number requires the chromosome design to be composed of two 64-bit long integers. The 64-bit integer has the same length as the double precision floating point number. Hence, it allows the preservation of the floating point number precision. One benefit derived from employing integer format is its affinity to some of the algorithm operators. For instance, the crossover operation implementation becomes more straightforward if the object of the operator is in integer format, instead of in a more complicated raw floating format. This advantage in the operator

implementation has been one of the reasons of the algorithm's popularity in the integer value based problems.

Objective function: environment

The environment element, which would test the fitness of the individual, is the objective function. In this example of a simple two variable function, the function that takes in two input variable is the objective function. The chromosome, which is composed of two 64-bit long integers, would be translated into two double precision floating point numbers. Then, the resulting floating point numbers are given to the objective evaluation function as inputs. The evaluation result from the objective function is associated with the individual, and is labeled as the fitness of the candidate.

Initial population

The next element in the algorithm implementation is the initial population creation. A two-step approach is utilized in the algorithm implementation. At the beginning of the algorithm execution, the initial population, which is larger in size than the on-going population is created. Each individual in the initial population is evaluated (fitness test) using the objective function. According to the test results, the smaller sized on-going population is assembled.

Steps in a iteration

The genetic operators in this implementation are the following three operators:

The first is the crossover operator. The operator makes two offspring candidates from two parent candidates. The new individuals are tested and associated with the corresponding fitness values.

After this cross over operation is complete, a mutation operator (the second operator) is applied to the current population. The operator introduces some error into the chromosome of the candidate in the population. For example, the operator will mutate the one 64bit long

integer value of 0xcc8176543f42044e into 0x8c8176543f42044e. If one looks at the first 4 bits of the number, the previous '1100' became '0100' due to the mutation. The mutated individuals are tested and associated with the corresponding fitness value.

The third operator is the selection operator. If a portion of population is replaced with new individuals, those new individuals are tested and associated with the corresponding fitness value. This selection step concludes an evolution step. As presented in the algorithm flowchart, Figure 3.3-1, each generation experiences the genetic algorithm operators and as a result, the next generation is produced.

This process is called as epoch in the algorithm implementation. After successive evolution through many epochs, the algorithm is expected to reach the solution, a minimum value of the two-variable function in this example.

3.4 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based stochastic optimization technique originally developed by Dr. Eberhart and Dr. Kennedy in 1995, and inspired by the social behavior of birds flocking or fish schooling [89, 90]. The PSO algorithm has attracted recent attention from power system researchers [91-96]. The algorithm offers a fresh prospect of overcoming the limitations in the other heuristic based optimization algorithms. Gaing [96] found that the PSO algorithm presented superior performance to Genetic Algorithms when those two algorithms were applied and compared in economic dispatch problem solution. The PSO algorithm shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GAs). Both the PSO and the GAs are initialized with a population of random solutions and search for optima by updating generations or candidate population. However, unlike the GAs, the PSO has no evolution operators such as crossover and mutation.

PSO algorithm

In PSO, the potential solutions, called particles, "fly" through the problem space by following the current optimal particles in a manner similar to a swarm of bees seeking a food source. Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (i.e., fitness) achieved so far. This value is called *pbest*. Another best value tracked by the particle swarm optimizer is the best value obtained so far by any particle in the neighbors of the particle. This location is called *lbest*. When a particle assumes all members the population to be its topological neighbors the best value is a global best and is called *gbest*. The particle swarm optimization concept consists of changing the velocity of (e.g., accelerating) each particle toward its *pbest* and *lbest* locations at each evaluation of the population. Acceleration is weighted by a random term, with separates random numbers being generated for acceleration toward the *pbest* and *lbest* locations. If the extent of the neighborhood becomes the entire population, the *lbest* becomes identical to the *gbest*. In a N-dimensional search space, the location vector $X_i = [x_{i1}, x_{i2}, \cdots, x_{iN}]$ and the velocity vector $V_i = [v_{i1}, v_{i2}, \cdots, v_{iN}]$ are the two vectors associated with each particle (or candidate) i in the population. Each particle updates its velocity and position according to equations (3.1) and (3.2) in the PSO implementation which was constructed as part of the presented research.

$$v_i^{k+1} = v_i^k + c_1(pbest_i^k - x_i^k) + c_2(lbest_i^k - x_i^k) \tag{3.1}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{3.2}$$

where,

c_1 and c_2 are two positive constants (weighting factors)

$pbest_i^k$ is the best position particle i achieved based on its own experience

$lbest_i^k$ is the best position particle based on the neighbor's overall experience

k is the iteration index

The first weighting factor in equation (3.1), 'c_1', is the factor that determines how much of each individual's experience is reflected in the optimization process. Likewise, the second weighting factor, 'c_2', is the factor that determines how much of the group experience can affect the individual's speed change in the optimization process. Thus, the first is called 'stochastic individuality' and the second is called 'stochastic sociality'. Figure 3.4-1 exemplifies the effect of velocity vectors associated with an individual particle. The particle is under a simplified circumstance, where those parameters such as 'c' and 'r' are all assumed as unity (1.0).

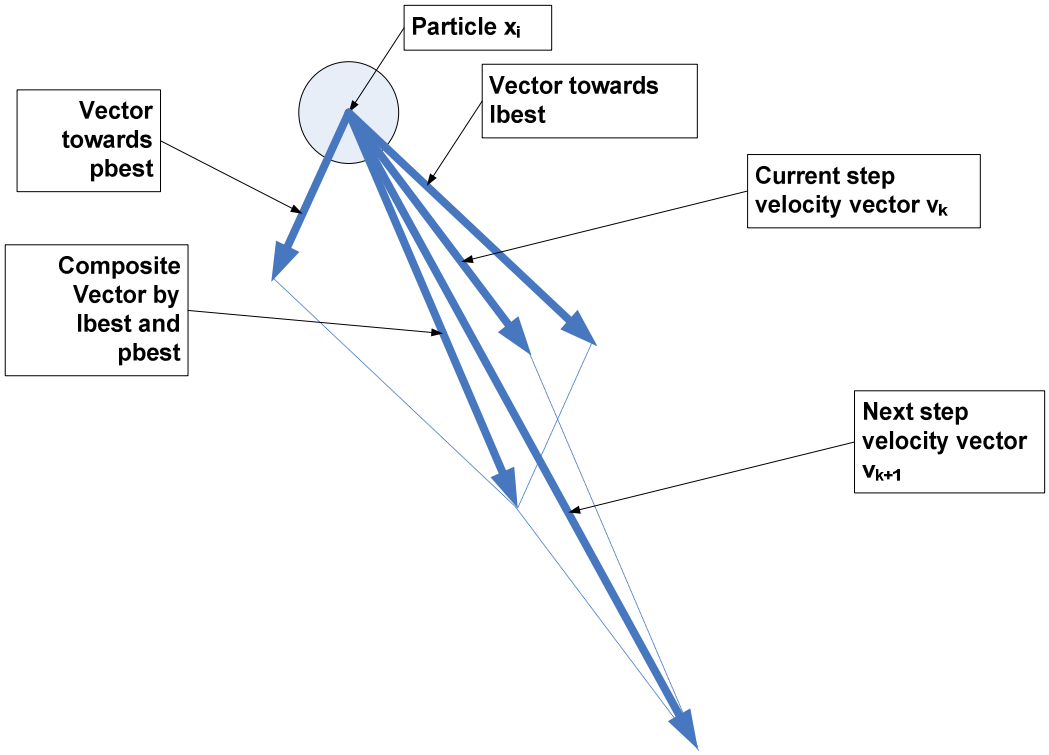

Figure 3.4-1 An individual particle with associated velocity vectors

The implementation was made in the C# language [97] and it was based on a public domain PSO library written in Java language [98, 99].

PSO algorithm: an analogy

The algorithm can be explained in a more illustrative way with a simple analogy. The swarm is assumed to be composed of bees. Each bee desires to move in the direction that it has personally determined to be best (*pbest*), but its neighbors tell it to move in the direction that is collectively determined as best (*lbest*). The bee, therefore, chooses a direction that is a linear combination of the two directions. The weighting given to any particular direction has a random factor, as described in equation (3.1).

PSO algorithm sequence

The flowchart in Figure 3.4-2 presents the process of the PSO algorithm execution.

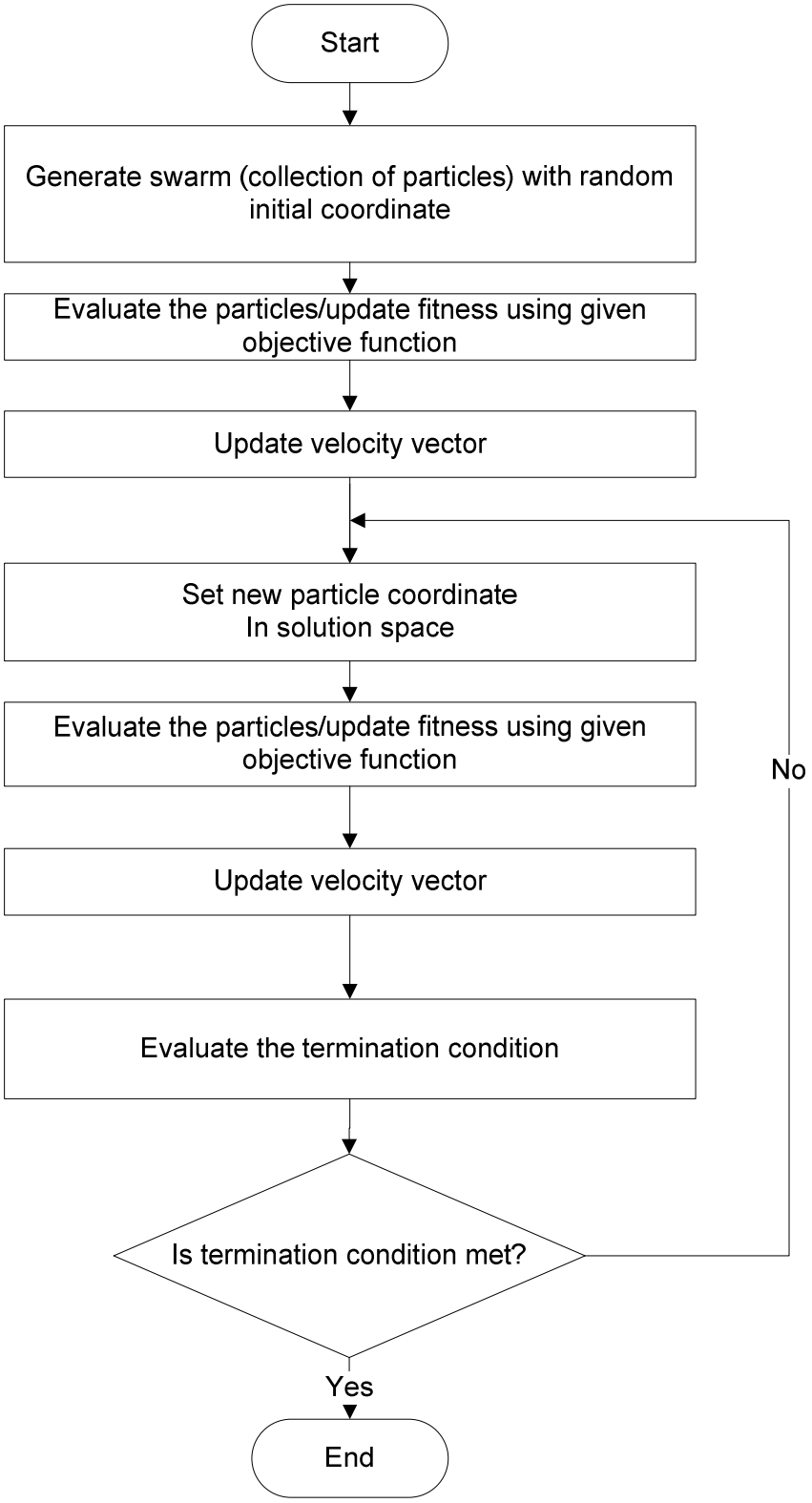

Figure 3.4-2 Flow chart of Particle Swarm Optimization

Neighborhood topologies

One factor of the PSO implementation, which may have a substantial impact on the performance of the algorithm, is the neighborhood topology [100]. Each individual in the population is affected by some other particles in the same population during the optimization process. The neighborhood topology in the algorithm implementation defines the way of identifying those 'some other particles' in the population. In the specific implementation which is part of this research, three different neighborhood topological structures were employed.

The first structure is the 'Circle' topological structure. Figure 3.4-3 illustrates the 'Circle' topological structure with a small number of particles (8 particles) and a neighborhood size of 2.

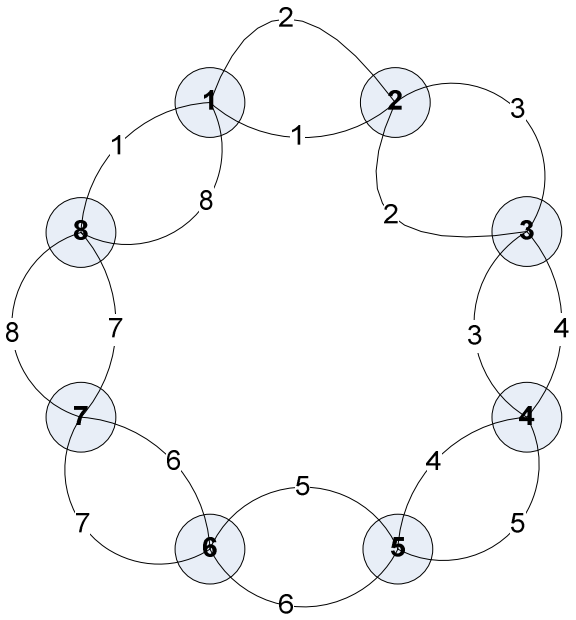

Figure 3.4-3 'Circle' topological structure

In this 'Circle' topological structure, the immediate neighbors construct the neighborhood for a certain candidate. In Figure 3.4-3, the particle number '1' joins the same neighborhood as the particle number '8' and particle number '2'. Thus, the *lbest* becomes the best value achieved by particle number '8', '1' and '2', so far.

The second structure is a ‘Star’ topological structure. Figure 3.4-4 illustrates the ‘Star’ topological structure with a small number of particles (8 particles).

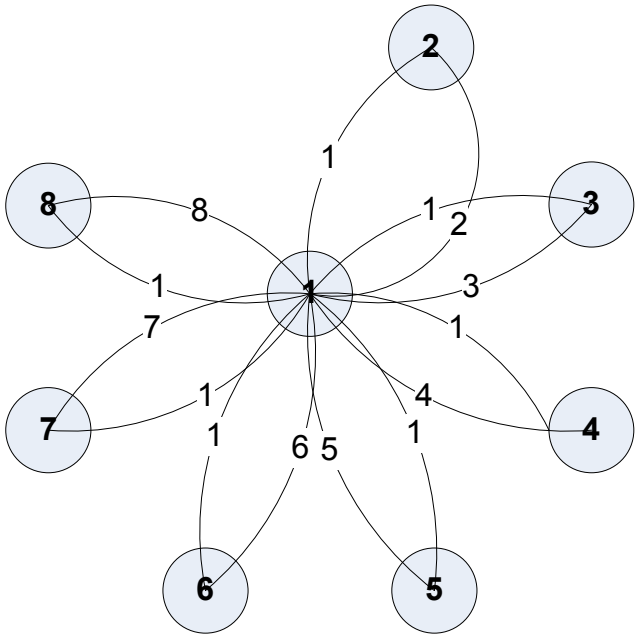

Figure 3.4-4 ‘Star’ topological structure

Only one particle is connected to the rest of the particles. All the information regarding the ongoing optimization process goes through the central individual, i.e., the particle number ‘1’ in the neighborhood presented in Figure 3.4-4. This ‘Star’ configuration is the social configuration most used in business, government and military organizations [100].

The last structure is the ‘Global’ topological structure. The entire population is treated as the individual particle’s neighborhood in this neighboring topology. Thus, the *gbest* value of the entire population achieved so far becomes the *lbest* value of an individual particle.

3.5 Applicability of optimization methods

During the research, it was noticed that the downhill Simplex algorithm is generally acceptable in terms of both convergence speed and the quality of the final outcome. This advantage is more pronounced in the research, where the number of optimization variables is relatively small. In addition to this advantage, this algorithm does not require derivative

information, thus allows easier implementation. However, the algorithm does not guarantee that the final outcome from convergence is truly optimum, namely global optimum.

Therefore, two additional optimization algorithms were evaluated. The first one is Genetic Algorithm (GA) and the second one is Particle Swarm Optimization (PSO) algorithm. The characteristics of these algorithms were explained in 3.3 and 3.4, respectively.

It is likely to see certain algorithms, including the evolutionary computing based optimization algorithms, would be suitable for certain types of problems. For example, it would be more promising when GA can be applied to combinatorial type problems. Unfortunately for the controller tuning problems described in this thesis, these heuristics-based optimization algorithms turned out to be too slow. However, they appear to be more suited to combinatorial optimization problems [79, 101]. One example of such a problem is optimization of breaker position (on or off) in a large distribution system in order to minimize transmission loss. In this case, over 700 switches had to be assigned on/off status to minimize loss [102]. In was the intention of the author to investigate such problems in the EMT arena, but he was unable to find a suitable problem more suited for GA and PSO, given the time and resources. In short, the choice of optimization algorithm generally depends on the nature of the problem which it tries to solve. Due to the restriction in both time and resources, this issue is left for future research.

4 METHOD FOR IMPLEMENTING OPTIMIZATION ENABLED ELECTROMAGNETIC TRANSIENT SIMULATION ON A REAL-TIME SIMULATOR

4.1 Introduction

The development and implementation of an OE-EMT method opened a new possibility of utilizing a simulation platform as a means of optimization. Various off-line simulation software packages such as PSCAD are now offering this capability as in-built function. In the case of PSCAD version X4, four different optimization algorithms (i.e., Golden section, Simplex, Hooke-Jeeves and Genetic algorithm) are available as options with a model named 'Optimum run'. Recall that, as briefly introduced in chapter 1, a close investigation of the optimization capability with off-line simulation software revealed two shortcomings:

Shortcoming of the off-line technique 1: Need for modeling a real system

First, in order to proceed with the optimization, the subject of the optimization must exist as a model in a simulation case. Further, the representation of the subject in the simulation case must ensure the highest level of fidelity. If these conditions are not met, the optimization results obtained from the iterative execution of the prepared simulation case cannot be fully trusted.

The simulation case tries to replicate the behavior of a real system. The purpose of optimization is to draw guidelines and obtain practical results, which can be applied to the real system. However, a certain level of compromise is inevitable when the representation of a real system is created in a simulation environment. For example, stray inductance and capacitance around the control circuit must be accounted for in order to achieve a high fidelity model, but the simulation scope expected in EMT-type simulation software often excludes this detail level of the simulation subject. This phenomena, stray inductance or capacitance, falls into the arena of different simulation techniques, using tools such as SPICE or an EMI analysis type simulation package. In theory, the same level of details can

be addressed in EMT-type simulation packages as well. In particular, most modern EMT-type off-line simulation software allows co-simulation with other simulation software packages [103, 104]; thus, any arbitrary level of details can be associated with the main simulation. However, the simulation speed and corresponding total execution time for the optimization process would be extremely large.

In addition to the issue of model fidelity, one more issues come with the requirement for off-line simulation. The key issue is regarding the extent and availability of necessary information for modeling. In order to represent a real system in simulation, every detail of the real system must be depicted in the simulation case. Otherwise, the level of compromise can be higher, rendering the optimization results less reliable. However, usually one practical obstacle, the unavailability of the necessary information, blocks the effort to close the gap between the simulation case and its real world counterpart. This unavailability can be attributed to two sources: (i) the information is no longer easily available, as the system becomes old; losing the necessary information and (ii) unwillingness of manufactures precludes obtaining the necessary details. If the necessary information becomes unavailable because of these factors, a certain level of compromise must be introduced into the process of simulation case building, otherwise the simulation case cannot be completed and no further opportunity for optimization can be anticipated. Consequently, the quality of optimization based on the compromised simulation case degrades and the optimization result cannot be fully trusted as a reliable candidate for real world application.

Solution to shortcoming 1: No modeling required

Real-time application of OE-EMT method can offer the possibility of overcoming this issue. In a real-time simulation environment, no proprietary information or retrieval of missing information in the process of simulation building is necessary. Because the simulation platform (i.e., real-time simulator) offers capability to be interfaced with real, external, physical devices such as controllers [105] or even power level devices [106, 107], the subject for the optimization (the external device) can directly participate in the simulation by itself. In this case, no further modification or alteration is necessary. Detailed explanation on the developed strategy in this solution follows in chapter 4.3.1.

Shortcoming of the off-line technique 2: Lengthy simulation time

Another shortcoming of off-line technique is that the time required before the optimization process can successfully be completed can be lengthy. As a simulation case becomes closer to a real world system, inevitably, the amount of necessary calculation for the simulation becomes greater.

Off-line simulation tools offer the capability of interfacing with external analysis software. As well, the tools additionally offer another capability to model custom devices using conventional programing languages, thus enabling users to write their own models. These capabilities enhance the fidelity of simulation, thus reducing the gap between the simulation world and the real world. However, the other aspect of this enhancement is the expansion of simulation time. When large numbers of simulation iterations are required for the optimization purpose, this would render the optimization process impractical.

The complexity of the simulation case itself can also be involved. In particular, the necessary number of calculations drastically increases once power electronics devices such as voltage source converters become part of the necessary simulation case for the optimization. As the switching frequency of a power electronics device in the simulation case increases, using a shorter time step for the simulation becomes necessary, resulting in longer execution times. Certain techniques such as interpolation [16] underneath the simulation execution can mitigate this problem (i.e., the switching frequency and simulation time step), but the techniques cannot completely eliminate the computational burden associated with the higher switching frequency. Another source of complexity in power electronics simulation comes from the topology of the switching devices in the simulation case. As more and more switching devices become involved in the simulation in a system such as a MMC (Multi Module Converter) based HVDC system, the necessary number of calculations increases because of the frequent admittance matrix inversion caused by the switching actions of each switching device in simulation. This situation can be analogous to the one with higher switching frequency and shorter simulation time step, mentioned previously. A good example of the increase in the necessary number of calculations can be found in [48].

Most off-line EMT-type simulation tools were created more than three decades ago. The legacy of gradual development during that time period still narrows the further enhancement of the simulation platform regarding utilization of multiple execution units in modern microprocessors. As result, when a detailed representation of a real world system becomes necessary or the simulation case includes complex and calculation demanding subsystems such as voltage source converters that impose heavy calculation burden, the practical resource restrictions such as execution time would prevent such trials. The single execution unit utilization of off-line software entails another issue in speeding up the optimization process. The inability of off-line programs in utilizing the opportunity for parallel processing becomes more evident in the case of heuristics-based optimizations. Most heuristics-based optimization algorithms assume a certain size of candidate population. The individuals in the population are isolated from one another in terms of association among them. In other words, the individuals in a population can be evaluated independently for the fitness of an objective. This aspect of the algorithms opens the possibility of parallelizing the evaluation process among the available execution units. However, because of lack of such a capability in the simulation engine of off-line software, the opportunity to speed up the evaluation of individuals in the candidate population is lost. All the necessary evaluations would be done in a serial manner. As result, substantially longer execution time is required for those heuristics-based optimization algorithms with off-line EMT type simulation software. This shortcoming of the off-line software makes those promising heuristics-based optimization algorithms less attractive.

Solution to shortcoming 2: Parallel processing enabled

If the required computational burden in simulation can be shared by multiple processing units, the limitation associated with the amount of calculation and corresponding simulation execution time can be alleviated substantially. In fact, this possibility was one of the fundamental motivations for the development of real-time simulation environments [10, 23]. In [108], the similar issue of large number of simulation iterations required for the purpose of Monte-Carlo style statistical analysis was addressed by a real-time simulation environment based on COTS (Commercial Off-The-Shelf) hardware. Real-time simulation

environments, such as RTDS, tapped the capability of parallel processing from the very beginning of development [5, 23, 109]. In this research, the possibility of utilizing parallel processing in a real-time simulation environment was explored in order to overcome the shortcomings of the off-line EMT simulation software. Chapter 4.3 provides a more detailed explanation of this approach.

4.2 The OE-EMT implementation in off-line EMT-Type Simulation Tools

The implementation details in off-line tools are as follows: The process can be separated into two parts: An outer loop and an inner loop.

- The outer loop, which is determined by the NLO algorithm implementation, generates trial points in the optimization space;
- The inner loop, the simulation program, assigns objective function (OF) evaluation values to those trial points with a low OF value indicating better conformance to the desired performance objective.

The user of such an environment (i.e., combination of simulation program such as PSCAD and the OE-EMT method) specifies the objective function to encapsulate the desired objective of the optimization process.

Figure 4.2-1 illustrates this process. The NLO algorithm, discussed above, is presented as an outer loop in Figure 4.2-1. This NLO algorithm receives the OF value obtained by EMT simulation, i.e., the inner loop in Figure 4.2-1. The outer loop also evaluates the exit control criteria. The exit criteria determine whether the optimization has converged. Once the convergence is determined, the optimization iteration stops. The final result of the optimization can be obtained from the last trial point before the algorithm is terminated.

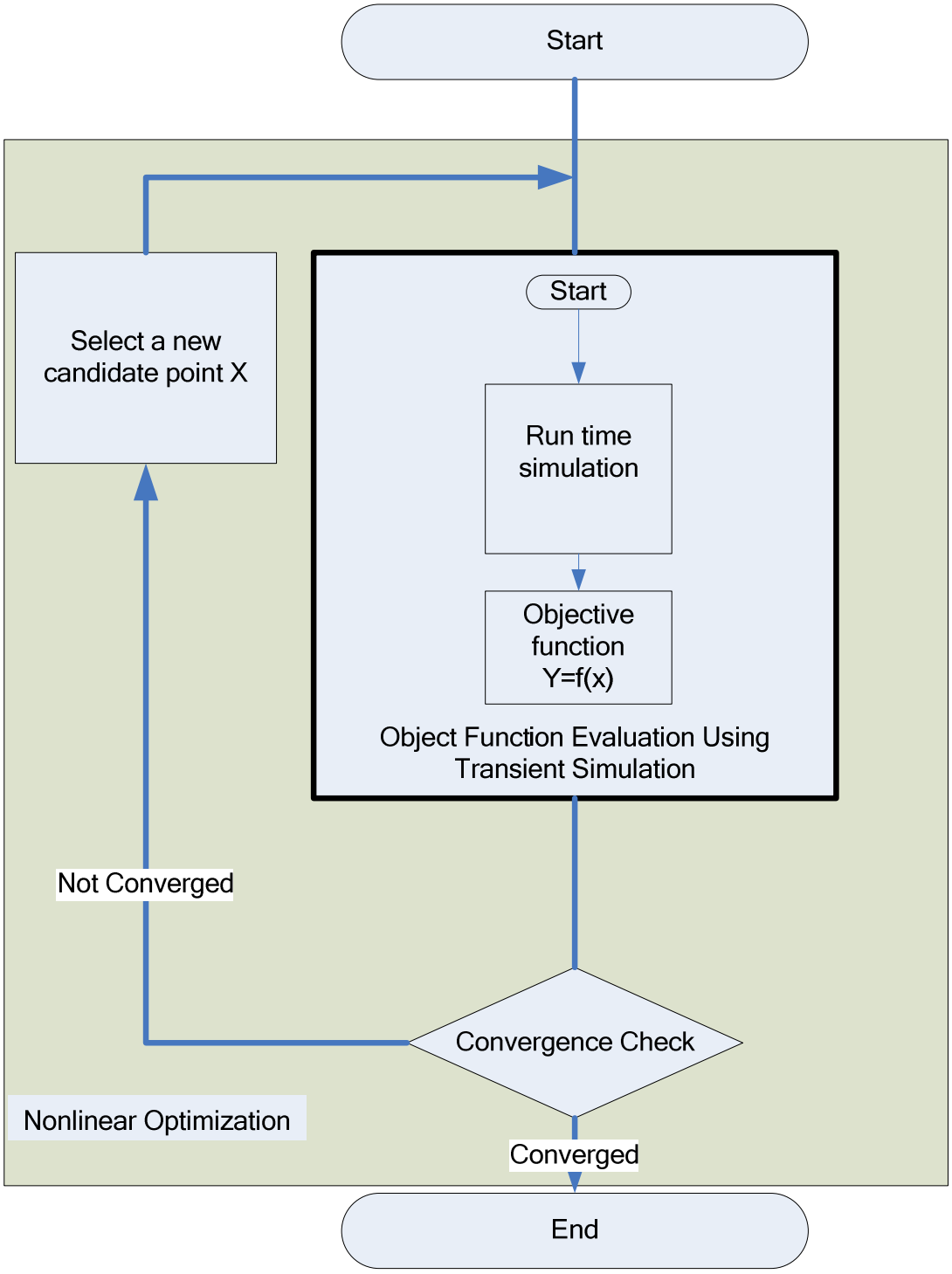

Figure 4.2-1 Optimization in an off-line EMT program

Example case: DC-DC converter parameter optimization

The operation of the off-line OE-EMT method is demonstrated by using a simple DC-DC converter (Cuk converter) simulation case. The simulation case and the associated NLO algorithm was constructed and executed in PSCAD. A more detailed description of the simulation case and the outcome of the off-line OE-EMT method application to the case can be found in [69].

The power circuit in the simulation case is presented in Figure 4.2-2.

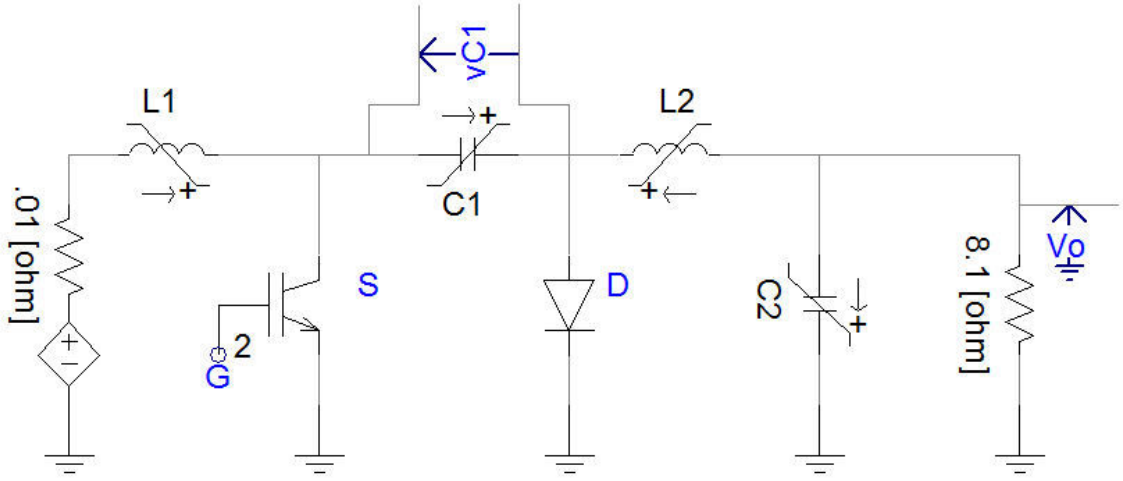

Figure 4.2-2 A DC-DC converter (Cuk converter) circuit

Two switching elements are included in the circuit above. One is a switch, denoted as 'S', and the other one is a diode, denoted as 'D'. When the switch is ON, the diode turns OFF. When the switch is OFF then the diode turns ON. The output voltage can be made higher or lower than the input voltage by controlling the duty ratio of the switch in the circuit. However, the output voltage is in reverse polarity to the input voltage. The ratio between the input voltage and output voltage is shown in equation (4.1) [110]

$$\frac{V_O}{V_S} = \frac{-D}{1-D} \tag{4.1}$$

where, D is duty ratio of switch S.

In order to control a certain output quantity such as output voltage, one can use a controller. The controller is presented in Figure 4.2-3. The controller includes a feed forward path and a feedback path. The aim is to regulate the load power ('PLoad') to its reference value ('Pref'). This is achieved by controlling the duty cycle ratio D if the IGBT switch 'S' in Figure 4.2-2. The combination between feed forward control and feedback control is implemented in the controller in the simulation case, with the purpose of improving transient controller performance as well as removing steady state error. The feed forward computes the duty cycle ratio D based on equation (4.1). Any residual error is corrected via a feedback path in the controller.

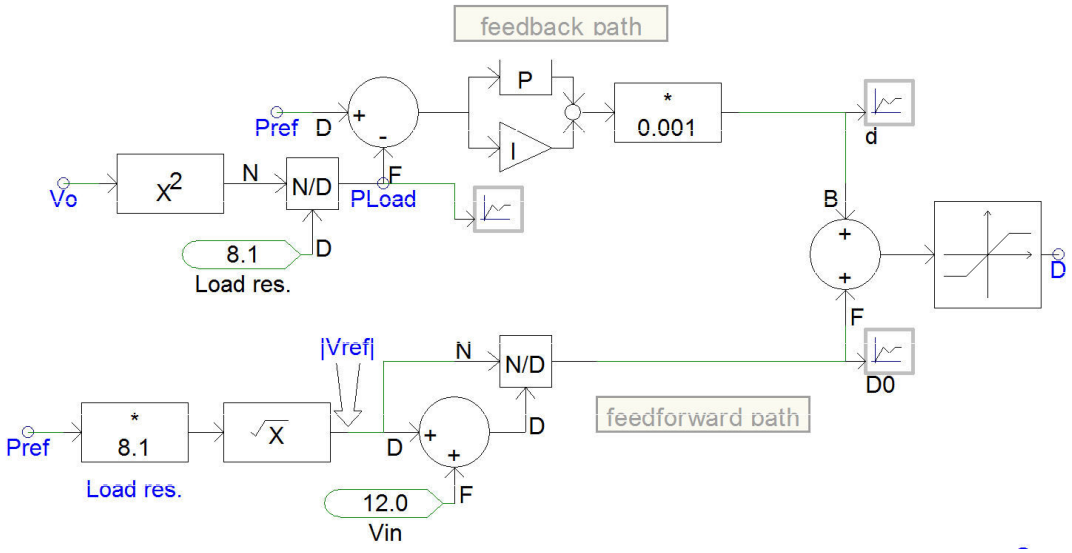

Figure 4.2-3 A duty ratio controller for Cuk converter

The parameters of the feedback PI controller (proportional gain and integral time constant) and the values of the circuit elements (L1, L2, C1 and C2) are selected by the optimization process.

The objective of this 6-variable optimization is defined as follows:

Enhance the controller's capability to trace the power order

Suppress the output current ripple in L1

Suppress the output current ripple in L2

Suppress the output voltage ripple in C1

The above optimization objectives are embedded in the objective function in the simulation case. The input of the objective function is the error between the desired output and the measurement from the simulation. Then, the error is squared and integrated (i.e., Integrated Squared Error-ISE) so as to penalize both positive and negative deviation. Also, squaring ensures that they do not cancel each other. Finally the individual ISE values are added together to produce the objective function evaluation value. A different weighting factor can be assigned to a certain objective to achieve better performance in that regard. For example, if a better performance in terms of the power output is more important than the rest of the optimization objectives, a higher weighting value can be assigned to the corresponding term in the objective function. If the transient performance of the controller requires more focus, then higher weighting value can be assigned to the transient part of the controller response.

The PSCAD implementation of the objective function is presented in Figure 4.2-4. The first path in this OF implementation penalizes the deviation of output power from the reference value. The second path penalizes the current ripple occuring in the inductor L1. Another current ripple on the inductor L2 is penalized in the third path. The last path penalizes the voltage fluctuation in the capacitor C1. The final value from this OF implementation, denoted as 'Obj' in the figure, is the weighted sum of these sub objectives.

The total OF from [69] is given as follows:

$$OF(L_1, L_2, C_1, C_2, K_P, T_i) = \int_{T_1}^{T_F} W_1(t)\left(1 - \frac{P_{Load}}{P_{ref}}\right)^2 dt + W_2 \int_{T_2}^{T_F} \left(1 - \frac{i_{L1}}{I_{L1}}\right)^2 dt + W_3 \int_{T_2}^{T_F} \left(1 - \frac{i_{L2}}{I_{L2}}\right)^2 dt + W_4 \int_{T_2}^{T_F} \left(1 - \frac{v_{C1}}{V_{C1}}\right)^2 dt \quad (4.2)$$

where, T_1(0.1 second) and T_F(1.0 second) are the step time of the power reference signal (P_{ref}), and the final simulation time, respectively. T_2(0.8 second) is selected to exclude the transient from being included in the ISE calculation.

In equation (4.2), the weighting for the inductor current ISE values (W2 and W3) and the capacitor voltage ISE (W4) were given as 50 in [69]. The weighting for the power output,

W1, is time dependent in order to place more emphasis on the transient response. W1 was given as the following equation in [69].

$$W_1(t) = 100,\ for\ t < 0.3\ (\text{second})$$
$$W_1(t) = 10,\ otherwise \tag{4.3}$$

Figure 4.2-4, presents the implementation of the OF in the PSCAD simulation case.

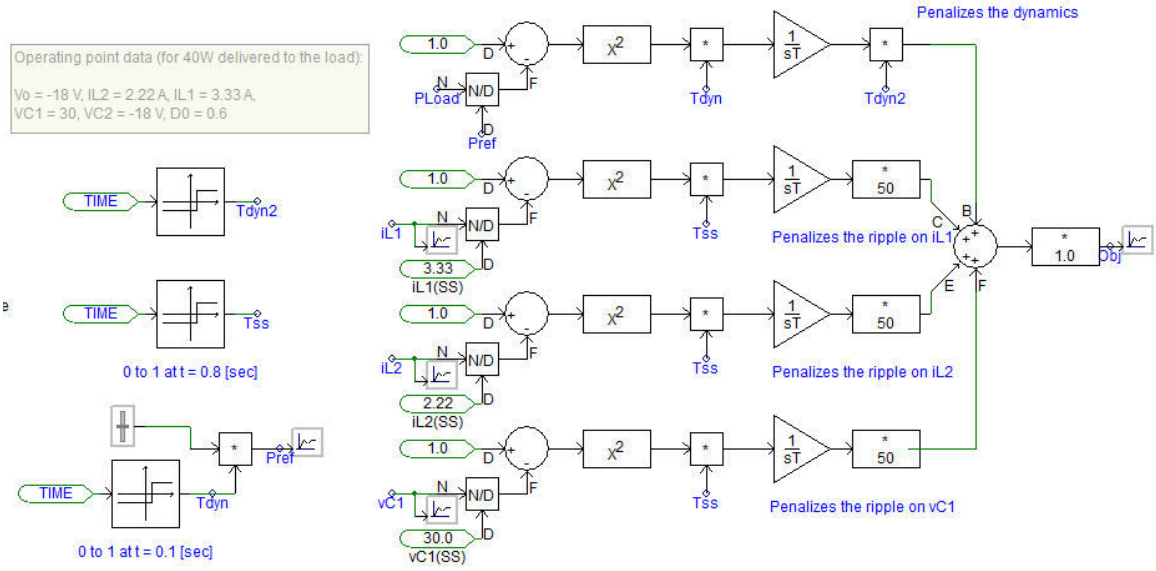

Figure 4.2-4 Objective function implementation in PSCAD

The NLO algorithm in the simulation case, the non-linear Simplex optimization algorithm of Nelder and Mead, tries to control several parameters in both the power circuit and the controller circuit to achieve the aforementioned objectives. The parameters that the optimization algorithm tries to manipulate are:

Inductance L1

Inductance L2

Capacitance C1

Capacitance C2

PI Controller proportional gain Kp

PI Controller integral time constant Ti

Hence, the solution space where the optimization algorithm traverses is 6-dimensional. The Simplex is composed of seven vertexes. Each vertex has 6 coordinates in the solution space. The NLO algorithm model with its six parameter outputs and one objective function value input is presented in Figure 4.2-5. A further processing of the output trial point is also possible in order to equalize the stride in each available moving direction in the search space. For example, as in Figure 4.2-5, the Simplex model output values which correspond to the inductances in the circuit are multiplied by a constant (1e-6) before the values are applied to the inductors in the power circuit. By this multiplication, the stride in the direction of the inductances can be equalized with the stride in the next two variables, capacitance values for the two capacitors in the power circuit.

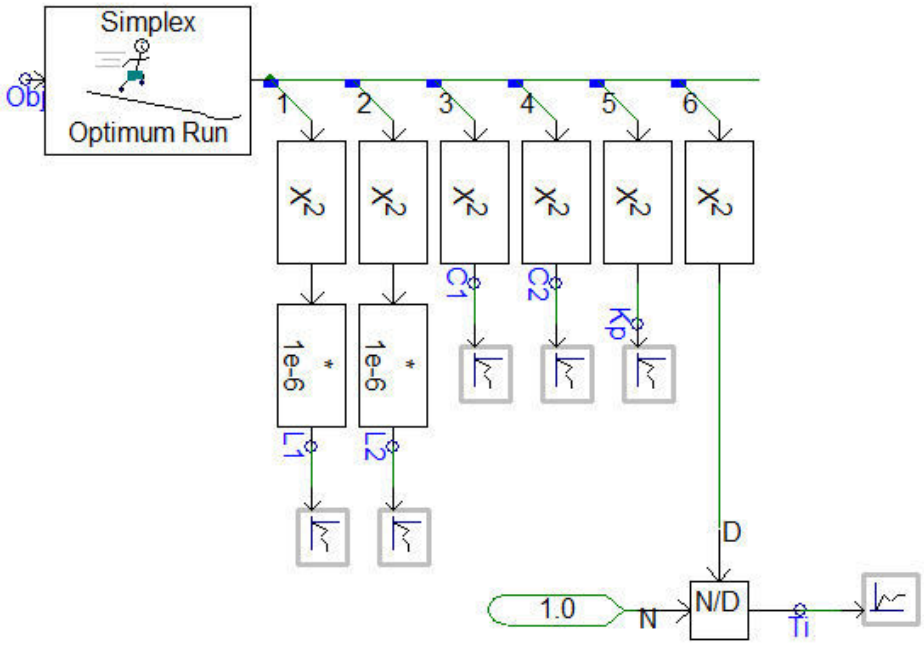

Figure 4.2-5 A NLO algorithm model in an off-line simulation

The NLO algorithm in the simulation case was able to converge under the convergence criterion value of 0.000001 after 336 iterations. The total execution time was 17 minutes 54 seconds (1074 seconds) in a PC with an INTEL core i7-2720QM processor.

Table 4.2-1, presents the initial as well as the optimized values of the parameters from the OE-EMT simulation case in PSCAD.

Initial parameters					
L1 (µH)	L2 (µH)	C1(µF)	C2(µF)	Kp	Ti
6400	4356	324	183	2.25	0.04
Optimized parameters					
L1 (µH)	L2 (µH)	C1(µF)	C2(µF)	Kp	Ti
8403	21174	188	37.6	0.078	0.028

Table 4.2-1 PSCAD Optimization result

The worst objective function evaluation output value during the iteration was 21.009 and the best value was 0.097. Waveforms in Figure 4.2-6 and Figure 4.2-7, comparison between 'Pref' value and P measurement at the output, presents the contrast between the worst optimization candidate (i.e., trial point) and the best candidate.

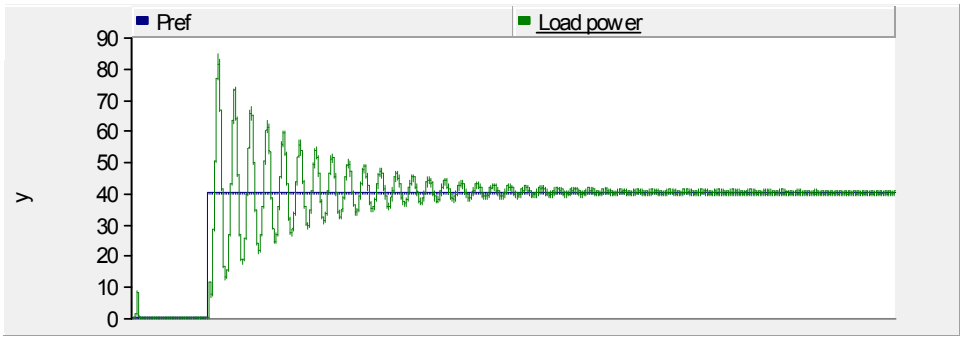

Figure 4.2-6 'Pref' and P measurement waveforms from the worst candidate

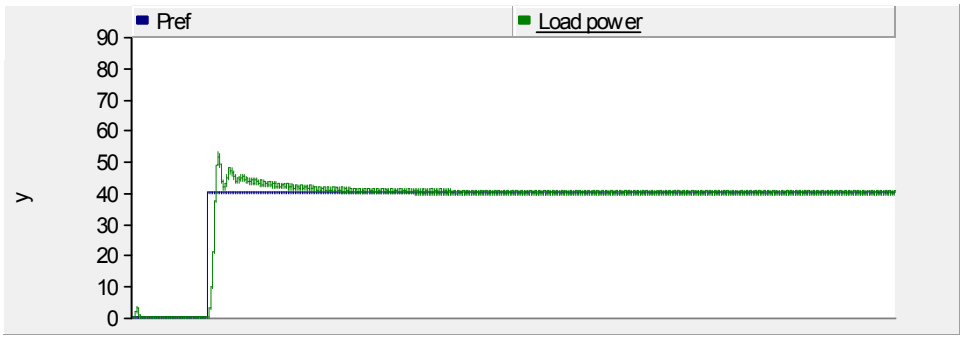

Figure 4.2-7 'Pref' and P measurement waveforms from the best candidate

Figure 4.2-8 presents the evolution of the OF evaluation value as the iteration number increases. One can notice that the OF value settles down to a range which is close to the final value long before the process was able to meet the given convergence criterion. Thus, the number of iterations could be reduced if the convergence criterion was relaxed further from the given value in this case.

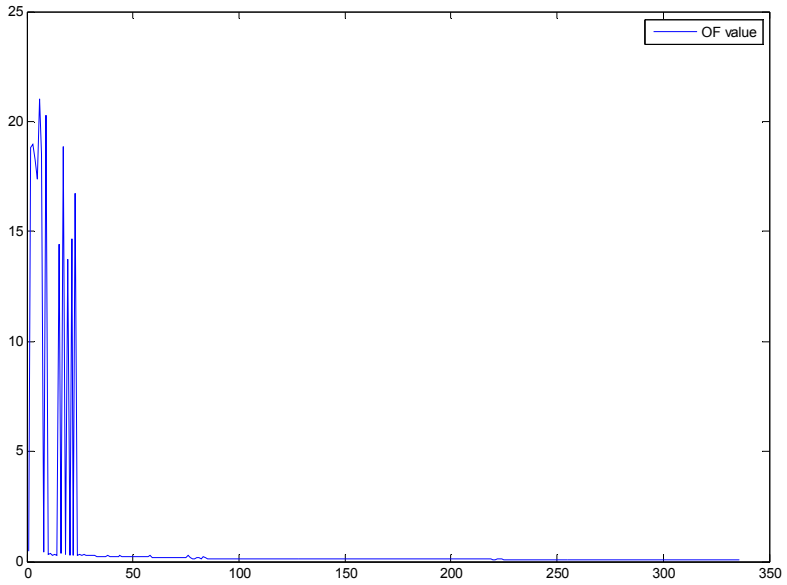

Figure 4.2-8 Trend of the OF evaluation values

4.3 Real-time implementation of OE-EMT method

The contribution of this thesis is the development of the OE-EMT approach for a real-time simulator. As briefly mentioned in Chapter 1, this new development, the extension of the technique to a real-time simulator provides several advantages over what was possible before with the off-line approach.

Benefits of real-time OE-EMT method

- The inherent high speed of real-time simulator allows faster evaluation of the objective function (OF), thereby, leading to much reduced optimization time. Figure 4.3-1 illustrates the relation between the NLO algorithm and the real-time simulator.

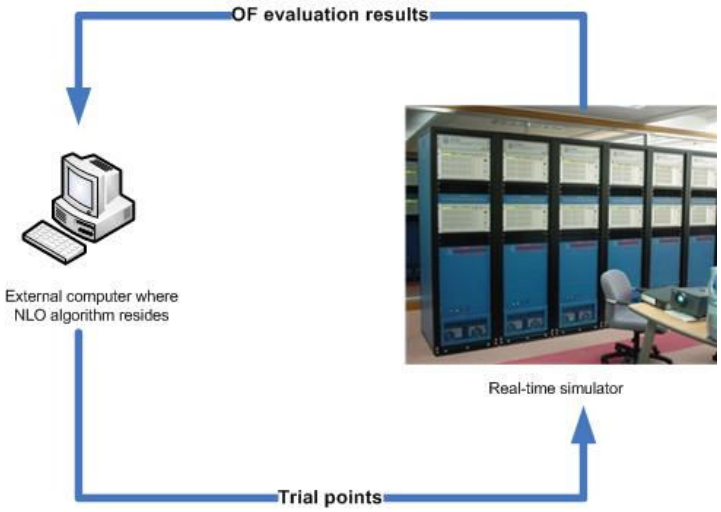

Figure 4.3-1 Faster OF evaluation using a real-time simulator

In off-line simulators, the computational burden (i.e., total simulation time) typically increases in a non-linear manner with the model's size and complexity. On the other hand, the real-time simulator can maintain real-time simulation speed by adding more processing hardware, which scales linearly with the modeled power system size [111, 112].

- As real-time simulators allow interface to actual physical hardware (controller), there is an opportunity to directly optimize the actual hardware. This opportunity provides these two sub-benefits.

(i) There is no need for an approximated model of the actual hardware. In contrast, the actual hardware would have to be modeled in an approximate way in the off-line simulators.

(ii) The parameters associated with the actual hardware can be directly manipulated by the NLO algorithms. Hence, the hardware becomes immediately available for the deployment in the field once the optimization process successfully completes.

Figure 4.3-2 presents the configuration and information exchange among the components/systems during the development. Part of the trial point information can be transmitted to the external hardware through a communication channel, such as RS-232C or Ethernet. A further discussion on this benefit follows in later sections.

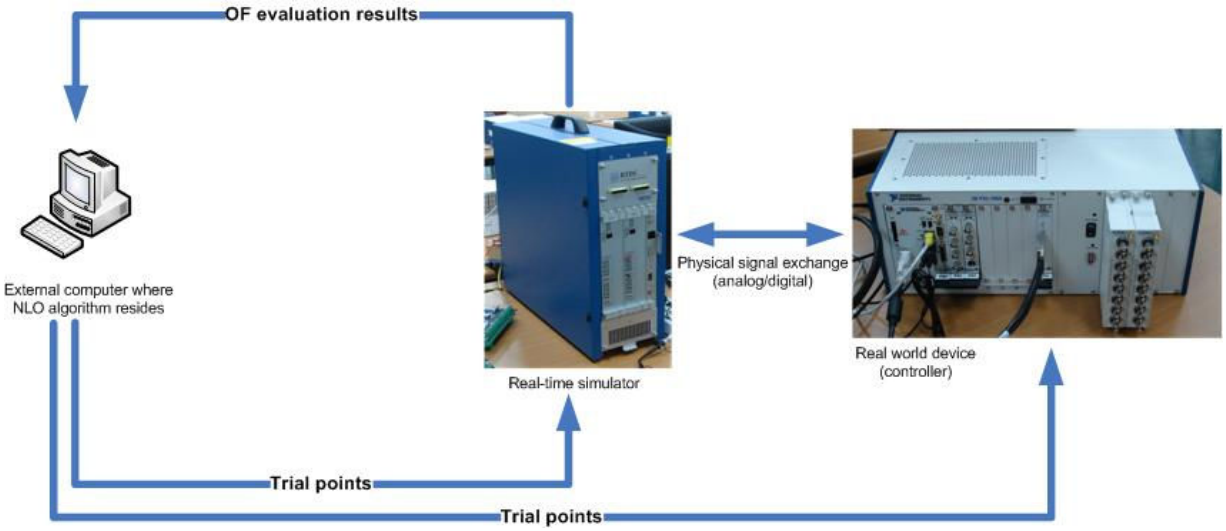

Figure 4.3-2 NLO algorithm with a real-time simulator and an external hardware

- By utilizing hardware parallelism in a real-time simulator, the OF evaluation of the optimization candidates (trial points) can be executed in parallel. Consider a simple example. A large scale real time simulator is composed of 10 processing units, but each simulation can be implemented using only 2 processing units. This provides the opportunity to run 5 different OF evaluations simultaneously, which is a significant asset for the heuristics-based optimization algorithms such as genetic algorithms and particle swarm optimization algorithms, as they require a very large number of simulation runs to find the an acceptable solution which is near optimal. Figure 4.3-3 presents this configuration. The NLO algorithm, which resides on the external computer, manages the communication between the NLO algorithm and each real-time simulator. The real-time simulators receive the optimization candidates from the NLO algorithm and evaluate the OFs. The results of the evaluations are collected by the NLO algorithm. Based on the collected results the NLO algorithms proceeds with the next step. More explanation on this benefit is presented in chapter 4.3.2.

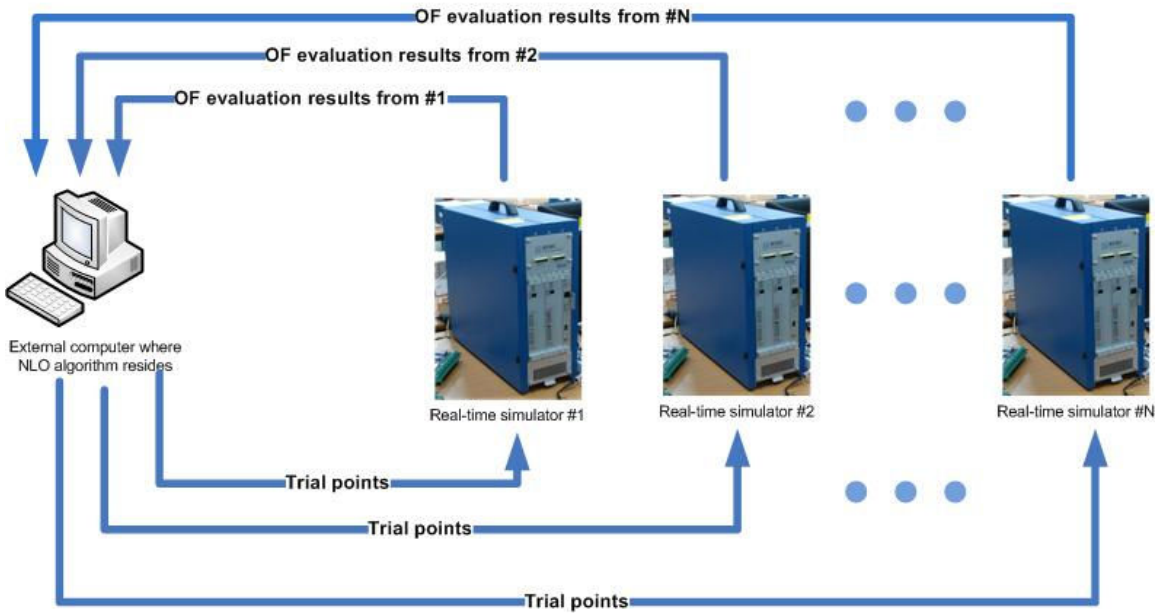

Figure 4.3-3 NLO algorithm with multiple real-time simulators

4.3.1 Direct external hardware interface

As documented in the previous section, the real-time nature of a real-time simulator allows the interfacing of the simulator with real world power or control equipment.

Due to the real-timeness of simulation and input/output capability of physical signals, a real hardware can be interconnected with the simulation environment. As the actual physical device is used, there is no need to model it in the simulation environment.

One example of this concept is presented in Figure 4.3.1-1 [113]. One of the well-known power system benchmark systems, IEEE New England 39 bus system [114], is modeled and simulated using a real-time simulator. The STACOM was physically constructed with hardware and actual controls. The rating of the real STATCOM system was 200 W. The STATCOM system was scaled up to a 200 MW system, then attached to bus number 21 in the modeled power system. The interface equipment between the real-time simulator and the physical STATCOM system consisted of A/D and D/A converters with suitable scaling and interface amplifiers.

A similar approach has been used with full rating power equipment (High temperature super conducting induction machine) using large amplifiers up to 5 MW [115].

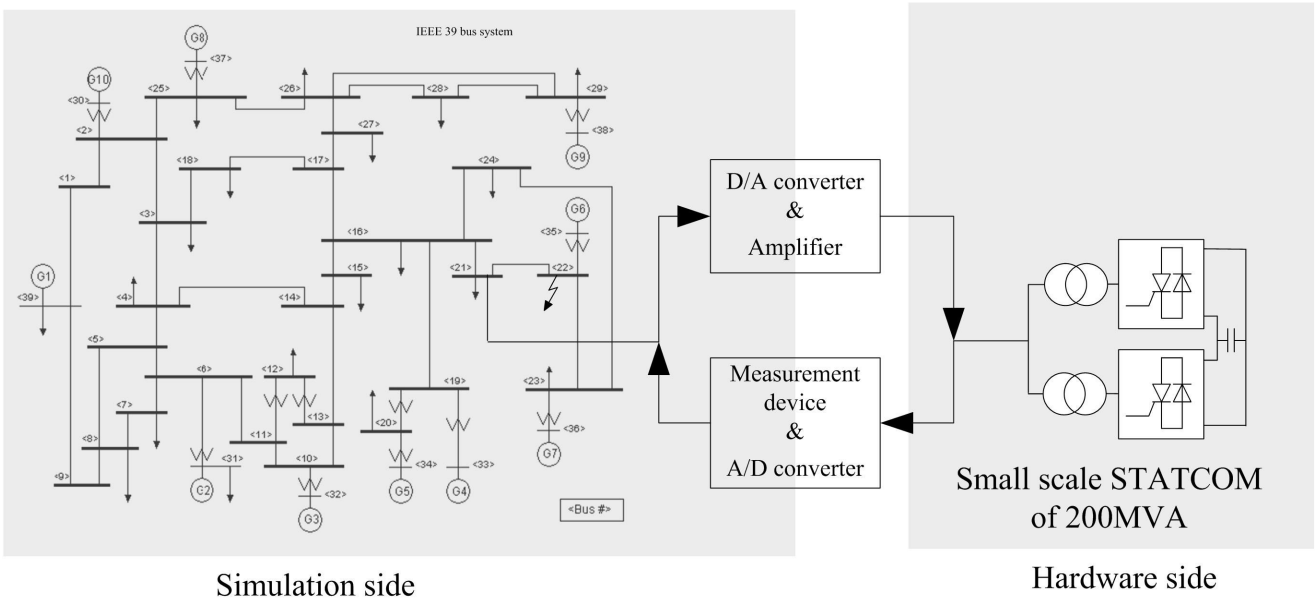

Figure 4.3.1-1 An HIL (Hardware in loop) example with a real-time simulation

Previous approach

The off-line tools need to use a software model of the device, because obviously the off-line simulator cannot directly tune the external device itself. The resulting optimized parameters, numerical values, need to be transferred to the physical device for the next step such as further verification or field deployment. The optimization result obtained by the off-line simulation software can be still suspect, because some features of the real world controllers such as parasitic capacitance effects are usually ignored when a computer model is constructed.

The first step to address this issue, the shortcoming off-line OE-EMT method associated with the gap between real world hardware and its approximate modeling, was introduced in [116].

An off-line MATLAB simulation of a modeled controller connected to a modeled power system was implemented. Using an optimization algorithm, also implemented in MATLAB, the controller was optimized.

A real-time simulator was then used to verify whether the numerical values of optimization from the off-line approach actually worked with the physical controller. A physical controller, a power system stabilizer (PSS) in this effort, was obtained and the optimized parameters from the MATLAB optimization were input to that physical controller. This controller was connected to the RTDS and the quality of the response was inspected.

Figure 4.3.1-2 describes this approach in more detail. The improvement in this effort is the introduction of a real-time simulator as a validation tool. However, the previously raised issues associated with the off-line technique such as approximation in the modeling still exist. The model has to be made in the off-line simulator, MATLAB in this case, in the same approximate manner as discussed. Then, as presented in the reference clearly, the final optimized result had to be transferred to real external hardware. Furthermore, the optimization result from the model in MATLAB does not offer any guarantee that the same response is obtained from the validation with the real-time simulator. Another deficiency found in this previous work is the lack of feedback, marked as shaded path in the flowchart. If the result obtained from the off-line simulator does not offer satisfactory results in the

verification stage, the model of the physical system must be improved. This can be laborious, and still there is no guarantee that the next optimization result will be acceptable.

As one can notice in this approach, by comparing the two flowcharts in Figure 4.3.1-2 and Figure 4.3.1-3, the fundamental cause of the aforementioned drawbacks in the work reported in [116] is the inability of the off-line approach to interface with the real world. These drawbacks can be eliminated by the approach proposed in this research.

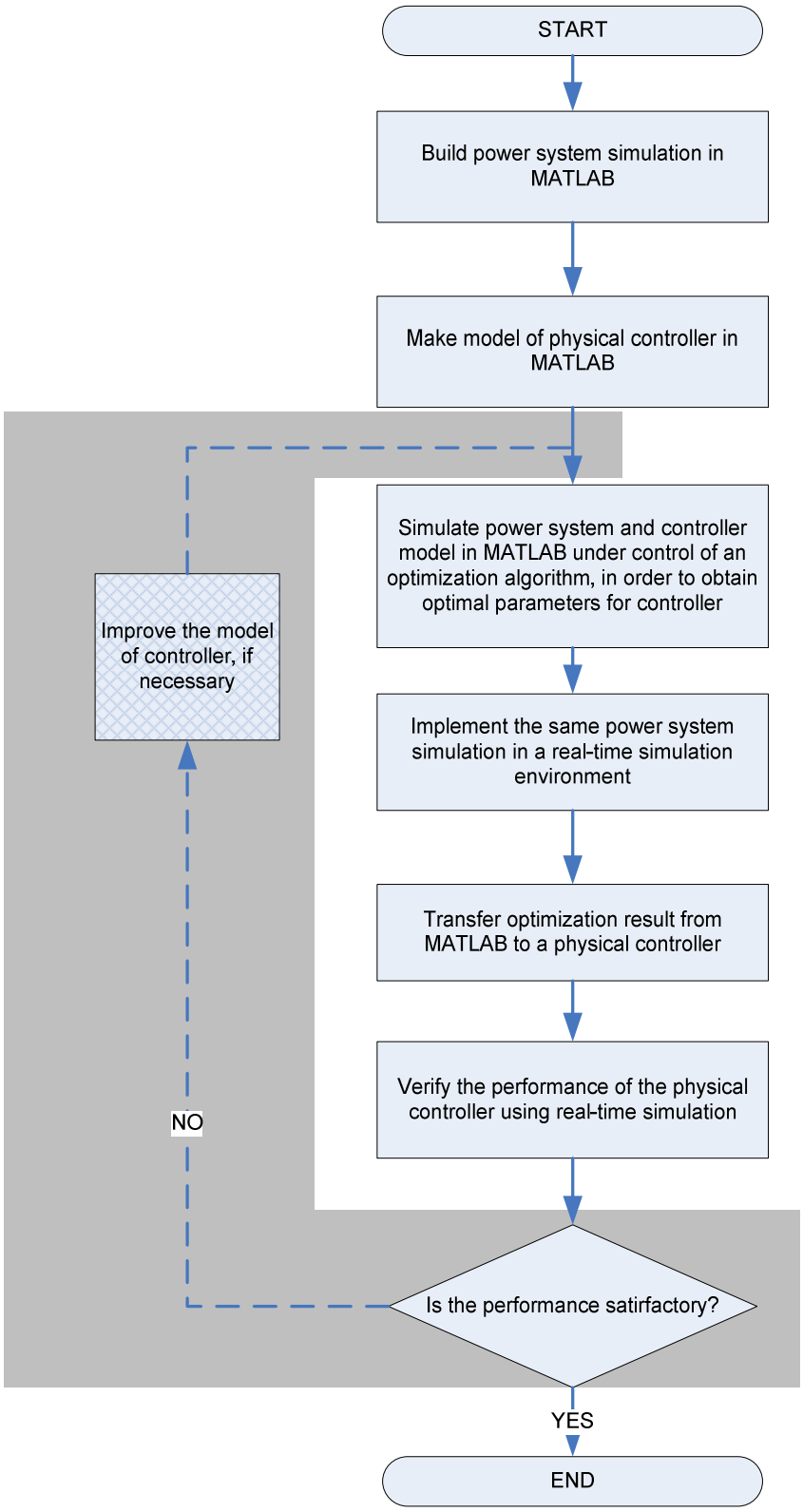

Figure 4.3.1-2 A previous approach

Proposed approach

In contrast to the aforementioned efforts regarding OE-EMT implementation with off-line simulations, the real-time optimization algorithm can directly tune a real world controller, not just a conceptual representation of it, when the optimization and simulation run together in a real-time simulation environment. A flowchart similar to one for the optimization using off-line simulation software (in Figure 4.2-1) is presented in Figure 4.3.1-3. The real-time simulator (RTDS in this particular implementation) is used as an evaluator of the optimization candidate. With the settings presented in this flowchart, as in the case of the OE-EMT application in the off-line simulation software, a wide range of optimization algorithms can be adopted as the necessary NLO as well.

As presented in Figure 4.3-2, a feature must be added to such a simulation environment to make this possible. This feature enables the NLO algorithm resident on the external computer to change parameters in the real hardware. Although commercial controllers now have the capability of external parameter setting, it is usually via a specialized custom hardware and software platform. In the examples presented in this thesis, controllers were developed in house with this feature. For the approach proposed in this thesis to become universally accepted, there is a need for manufacturers to develop standardized interfaces that would allow such parameter settings.

In the particular implementation of the in-house controller constructed in this research, an RS-232C communication capability was embedded on both sides. Before the communication initiates, both sides, i.e., the NLO algorithm implementation and the external controller, are already in agreement in terms of communication speed and the rest of the details part of which are from the RS-232C communication specification [117]. Once the optimization process begins, the candidates (trial points) are generated. At the NLO algorithm implementation, the candidate values are translated into a fixed point real number representation, Q24 format in the IQmath, virtual floating point library [118]. The controller receives these values through the communication channel. Once the received content is verified, the encoded values are decoded back to their original real values. The decoded real values are applied as the parameters to the controller. Finally, the controller interacts with

the real-time simulation with the new parameters and the performance of the optimization candidate is evaluated.

The external controller can be connected to the real-time simulation environment using its real world input/out interface capability. This interface puts the controller in a very realistic testing environment where the controller (i.e., the test subject) cannot discern whether it is connected to the real world environment or to a simulated one. By connecting the real world controller to the real-time simulation, the controller can execute its task in parallel with the simulation, exchanging the real input and output signals in synchrony with the simulation. At the same time, the candidate parameters selected by the NLO algorithm are directly input to the external hardware. By utilizing this combination, real-time interface between the external hardware and simulation and direct application of the optimization candidate onto the external hardware, the need for the conceptual model of the physical device is eliminated. The only additional information, in addition to the input and output signal specification of the external hardware, is regarding how to transfer the trial point (i.e., parameters) from the NLO algorithm implementation to the hardware.

As discussed in chapter 4.3, the proposed approach resulted in the direct optimization of the external hardware parameters, making the hardware ready for immediate field deployment.

Comment on optimization with power hardware in loop

Note that usually the hardware under test would be a controller, and hence is unlikely to be damaged physically by the simulation going into dangerous overvoltage, etc. However the approach, in theory, can be used to optimize power hardware in loop, where the possibility of physical damage is real. This will require additional thought and has not been addressed in this thesis.

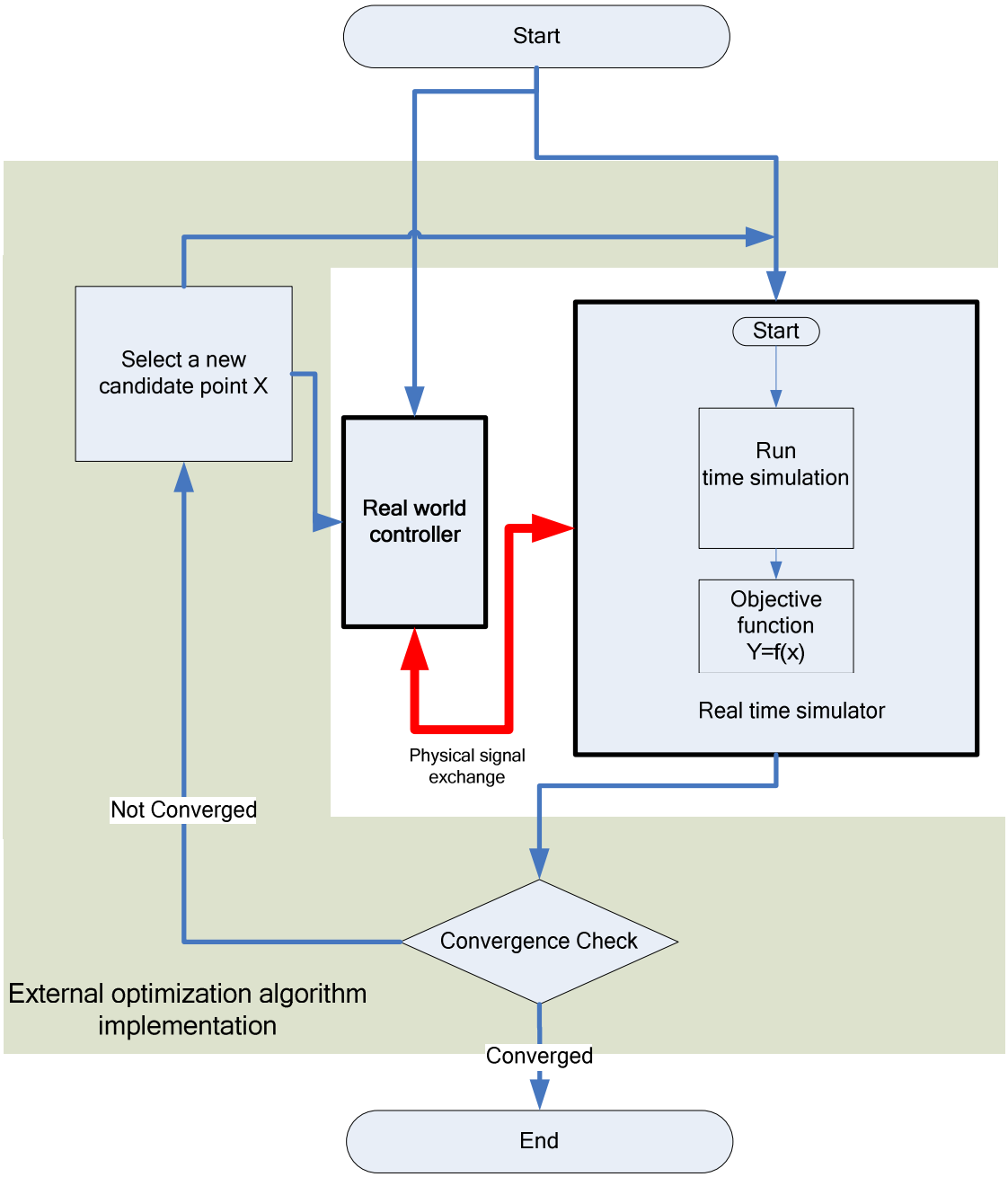

Figure 4.3.1-3 Real-time OE-EMT application with a real world controller

4.3.2 Utilizing hardware parallelism in real-time EMT-Type Simulation environment

One other way in which a real-time simulator can be used in optimization studies is to exploit its inherent parallelism. For example, if a simulation case requires 2 processing units, but 10 processing units are available, then it is possible to run 5 sets of simulations simultaneously. Certain NLO algorithms (e.g., particle swarm optimization or Newton's method [119]) can benefit by this capability.

Parallelism in computational hardware

If one investigates the parallelism in a specific hardware platform, many different layers of parallelism can be found.

The lowest level of parallelism in modern micro-processor is at the core level. Each core is composed of many identical or different execution units. In the example of Freescale MPC7448 architecture [120], 8 different execution units (ALUs and vector calculation units) can be identified. If software is written using proper facilities, which would allow the access to the execution unit level, then the software would be able to utilize the parallelism in this level, i.e., execution unit level. One example is writing in-line processor specific assembly code with full awareness of the context flow. However, most of high level programming languages require prohibitive amount of resources in terms of time and effort to do so, thus the utilization is usually beyond the programmers' access.

The next level of parallelism is the processor level. A single processor can be composed of many cores on (currently) one or two semiconductor dies in a single package. Inclusion of multiple processing cores in a single package is a relatively recent approach which most micro-processor manufactures are trying to utilize in order to boost the processor capability. The silicon implementation of this type of parallelism is usually referred to as a multi-core system.

The next level of parallelism can be at the board level, built by combining multiple processor chips (silicon). Such a system is usually referred to as multiprocessor system.

The next level of parallelism can be composed with multiple boards in conjunction with a common communication path. A popular communication path, usually referred to as 'backplane' in the industries, includes the VMEbus standard [121] and AdvancedTCA (ATCA) architecture standard [122]. By combining many multi-processor boards into a single system, such a system can provide a substantial amount of computational power. This level of parallelism can be referred to as enclosure level.

The next level of the parallelism hierarchy can be defined as cluster level with the connection of multiple enclosures in a loosely coupled way. The loose coupling between the enclosures can be a conventional communication path such as TCP/IP communication protocol [123, 124] over commodity Ethernet.

Parallelism in real-time simulator hardware

A closer investigation of the hardware/software design of real-time simulation tools also reveals multiple layers of parallelism in the construction. In the case of RTDS, two different levels of parallelism can be noticed in its hardware and software design. The first level is the scope of a single 'rack'. The design of the RTDS hardware was inherited from an industry standard VMEbus system, thus a rack means a collection of processing cards on which multiple processing units, such as DSPs (Digital Signal Processors) or conventional microprocessors, are located. Each of those processing cards is equipped with the necessary interface to the common communication backplane. The processing units in a single rack can communicate with one another via data transmission through the backplane. The hardware structure of an RTDS rack is presented in Figure 4.3.2-1. The embedded software in the system manages the entire execution of the calculation on the participating processing units. One of the system manager's vital functions is synchronizing the calculation and communication between the participating processing units in a simulation. Once a certain calculation period in a time step is complete, the management routine in the rack will initiate the communication period. During this period, the processing units in the rack will transmit and receive the necessary data from one another. When the completion of the necessary data transmission is signaled, then the managing software issues a signal which will initiate the

next phase of calculations in the processing units. This level of parallelism is associated with the corresponding 'Enclosure (Box) level'.

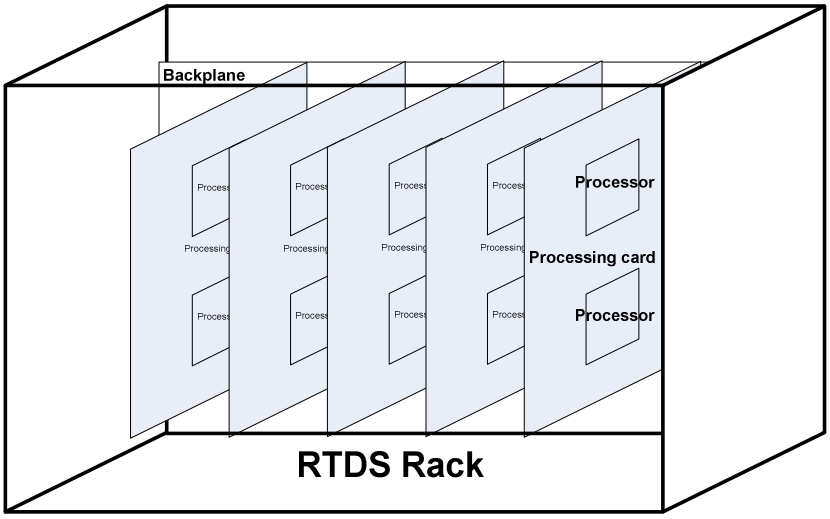

Figure 4.3.2-1 A single RTDS rack

The second level of parallelism can be found between racks. Individual racks can be interconnected together with the extension of the common communication backplane in a rack. A specially designed communication card takes the role of communication relay in an RTDS system. Using the communication path between racks, any processing unit in any rack can exchange any necessary information for the execution of the simulation with any other processing unit in the entire system. This level would correspond to 'Cluster (Collection of boxes) level' .

The third level of parallelism is in between separate RTDS systems. Multiple RTDS systems, each composed of one or more RTDS racks, can be used to allow multiple instances simulation to be executed in parallel.

User of parallelism with a real-time simulator

In case of the implementation of OE-EMT methods in off-line software such as PSCAD, the boundary between the objective function evaluator and the optimization algorithm, which governs the entire optimization process is not clearly defined. For instance, a single

simulation case includes both functions (i.e., objective function evaluation and optimization algorithm) in the current PSCAD implementation (PSCAD version 4.2.1). One disadvantage of mingling these two different functions is making the execution time on a single processing unit longer. To the contrary, in the proposed approach, the separation between the two functions is distinct. The OF evaluation (i.e., EMT simulation) is done via the real-time simulator. The NLO algorithm is resident on a distinct platform, such as a PC. One benefit expected from this separation, i.e., the separation between OF evaluator and NLO algorithm implementation, is more freedom in the NLO implementation. In consequence, a higher level language and runtime environment such as MATLAB can facilitate the NLO algorithm implementation.

Comment on using parallelism with controller hardware in loop

Strictly speaking, to use parallelism when actual controllers are present will require multiple installations of the controller. This may not always be practical, so it is more likely that the parallelism will used when the model is fully in real time simulation. One issue with multiple installations of the controller is that they may not all be precisely identical.

5 EXAMPLE STUDIES – NO HARDWARE IN LOOP

5.1 Simple studies

5.1.1 Down-hill Simplex algorithm in real-time simulation

Before applying the developed real-time OE-EMT method to complex power system problems such as power system controller tuning, the method was evaluated using a simple arithmetic function as an objective function. This procedure is described below.

Equation (5.1) is a simple two-variable function. The down-hill non-linear Simplex algorithm (i.e., non-linear Nelder-Mead Simplex algorithm) will be used to search for the minimum value. This simple two variable function is a candidate for evaluating optimization algorithms such as the down-hill Simplex optimization algorithm. This function exhibits several ‘trenches’ in which some optimization algorithms tend to settle. Moreover, the function has a contour that sometimes challenges the optimization algorithms with convergence difficulty. The geometrical characteristic of the function can be observed in Figure 5.1.1-1.

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2 + (\sin x_1)^2 x_2^2 \tag{5.1}$$

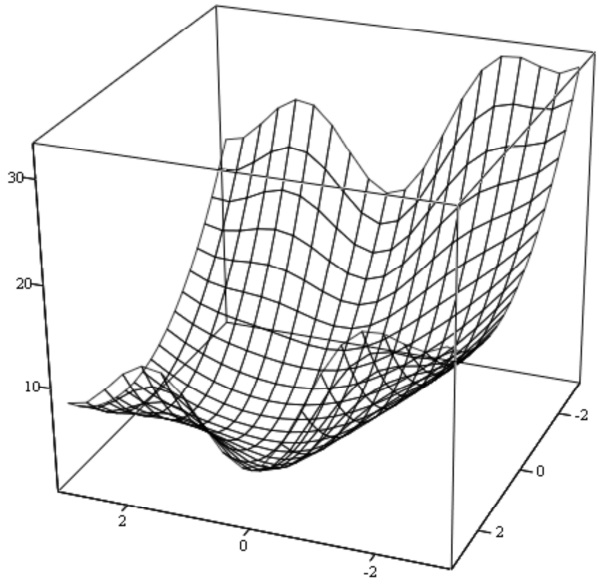

Figure 5.1.1-1 3-D plot of the target function

This problem has an analytical solution as presented below, and hence this problem is good for testing optimization algorithms because the obtained solution from an optimization algorithm can be compared with the analytical solution.

The minimum of the equation is the solution of the partial derivatives. Equations (5.2) and (5.3) present the partial derivatives of the equation.

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2(x_1 - 1) + 2(\sin x_1)(\cos x_1)x_2^2 \tag{5.2}$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = 2(x_2 - 1) + 2(\sin x_1)^2 x_2 \tag{5.3}$$

At the minimum (or optimum),

$$\frac{\partial f(x_1, x_2)}{\partial x_i} = 0 \quad (i = 1, 2) \tag{5.4}$$

Solving Equation 5.4 for x_1 and x_2 gives the following result:

$$x_1 = 0.77459472$$

$$x_2 = 0.67150265$$

The mathematical function in Equation 5.1 is used in the exploration of multiple implementation paths. There were two ways: One includes writing a program using a script language. The other is to use the building block of the real-time simulator. This is exemplified below for the special case of the down-hill Simplex algorithm, which is described in chapter 3.2.

Implementation 1: Iteration by restarting a simulation case

In this approach, the trial point evaluation was conducted by restarting the real-time simulation case. By doing this, the same initial condition for each evaluation is guaranteed. However, the overhead associated with the starting of a real-time simulation case such as the communication between the graphical user interface and the real-time simulator is added onto the total execution time of the optimization algorithm.

The Simplex algorithm can be implemented by using the script capability of the Runtime software module in RSCAD. The graphical user interface (GUI) of RTDS (i.e., RSCAD) is composed of many software modules. The 'Runtime' module is for monitoring and controlling the simulation running in the RTDS hardware. Frequently, repetitive executions of real-time simulations are required for real physical device testing with the simulator. For example, exhaustive testing of a protective relay requires hundreds of simulation runs with various system conditions. In order to automate the execution of evaluation iterations, the 'Runtime' software module provides a scripting ability. The syntax of the 'Runtime' script is a subset of the C language [125] syntax. The syntax is equipped

with high level programming language features such as flow control, thus enabling users to cover more complex iterative process automation. This ability in 'Runtime' is used as a vehicle to interconnect the external optimization algorithm with the real-time simulator, i.e., the RTDS simulator.

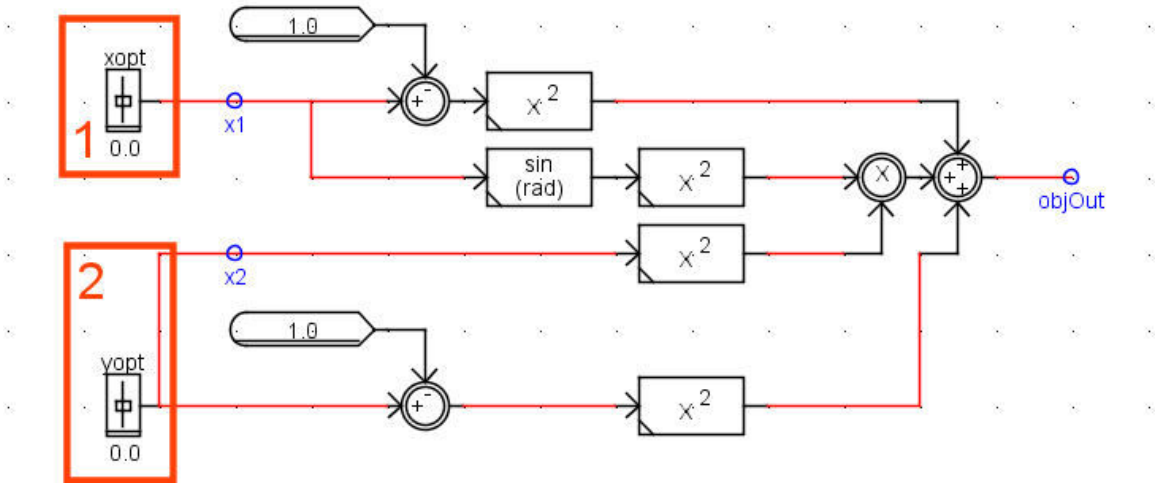

Figure 5.1.1-2 RTDS simulation case of a simple optimization application

Trial values of optimization parameters, i.e., x1 and x2 can be selected via the two 'slider's marked as '1' and '2' in Figure 5.1.1-2. Their values can be controlled using a script language in the real-time simulation environment, RTDS simulator in this experiment. In particular, the down-hill Simplex algorithm is coded using the RTDS script language. The implementation changes x1 and x2 (sliders 1 and 2 in the simulation case) according to the OF evaluation output from "objOut" value. The new trial values are applied to the slider by recompiling the case again, and this process adds a substantial amount of overhead in total simulation time.

The convergence criterion was selected so that the process would stop when the OF evaluation values from successive OF evaluations differed by a value, which is referred to as the convergence threshold. When this implementation was tested, a total of 61 iterations were executed before the convergence threshold of 0.00000001 was satisfied. The optimization process took 169 seconds before termination. Thus, each evaluation took an average of 2.77 seconds. Time to convergence, i.e., 169 seconds, is excessively long when

compared with the same experiment with different implementation approach, which will be explained later. The involvement of the communication between the script language runtime and the real-time simulator is the major reason for the long execution time.

Script in the RTDS simulator is used primarily for controlling the simulations in an automatic manner. Thus, the expressive power of the script is considerably less than conventional programming languages such as the C language; this lack of expressive power hampers the implementation task of an optimization algorithm when the algorithm itself requires intricate control of both data and execution flow.

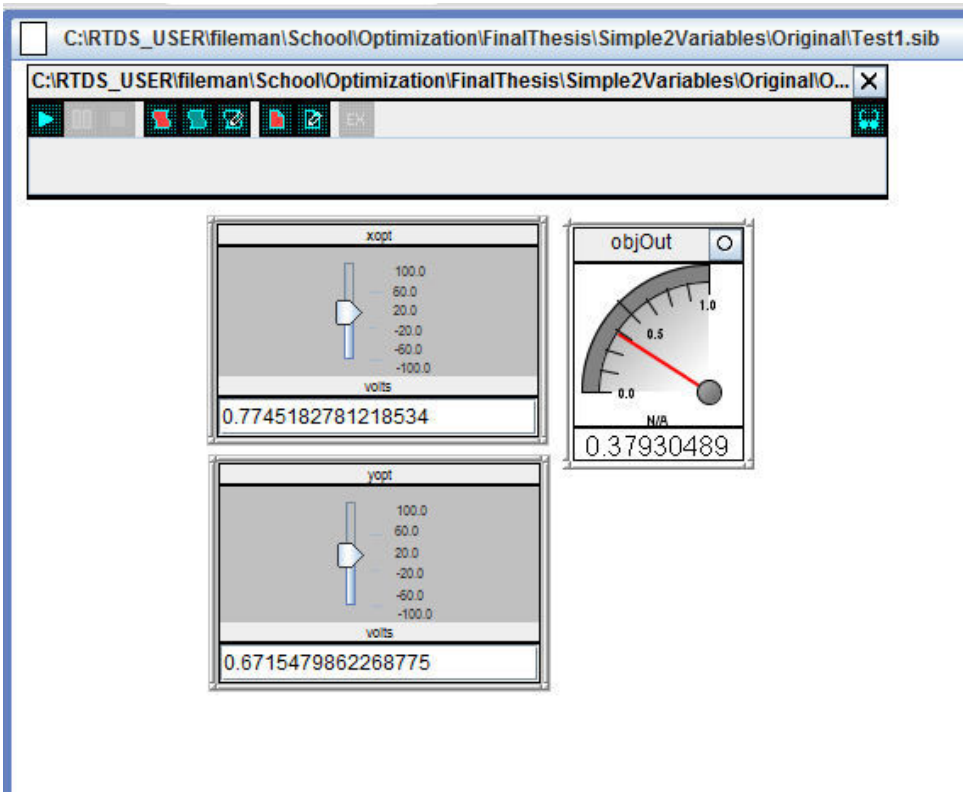

Figure 5.1.1-3 RTDS simulation result by Runtime script

The captured screen image in Figure 5.1.1-3 presents the final result from running the Simplex algorithm implementation using the Runtime script. The function minimum converged at the value of 0.3793 when x1 = 0.7745 and x2 = 0.6715. This final result from the experiment agrees well with the analytical solution. The difference between the analytical solution and the solution obtained by this experiment is less than 0.01%.

The advantage of the approach in this experiment, i.e., using RTDS script to implement the NLO algorithm offers flexibility in the algorithm implementation as well as the OF evaluation case building. It is much easier to write a certain algorithm in a simple language such as the script language than using low level language such as machine language. In addition, the simulation case, which produces the OF evaluation values, is completely separated from the NLO algorithm. Hence, several issues related to the NLO algorithm implementation such as ensuring that each EMT simulation run starts from the same initial condition are not a problem.

The disadvantage of the approach is the long execution time during the optimization process. This is because delay associated with communication between the off-line PC running the NLO algorithm and the real-time simulator is now encountered whenever new trial parameters are generated. However, the communication overhead is consistent regardless of the size or scale of the simulation case, thus it can be ignored practically when longer simulation times are required for the OF evaluation.

Implementation 2: Multiple iterations in a single simulation run

Another way to implement the same NLO algorithm in a real-time simulator is using a single simulation run. A long run is initiated, and the trial points are changed during this single run. A new parameter set is applied after the prior set results in steady state.

In order to incorporate the optimization algorithm into a simulation case, a real-time simulation model was constructed. In this implementation, the NLO algorithm runs in the real-time simulation loop, and does not require the additional communication overhead in simulation time as in the previous method. Figure 5.1.1-4 presents an RTDS simulation case prepared to locate the minimum value of the above function.

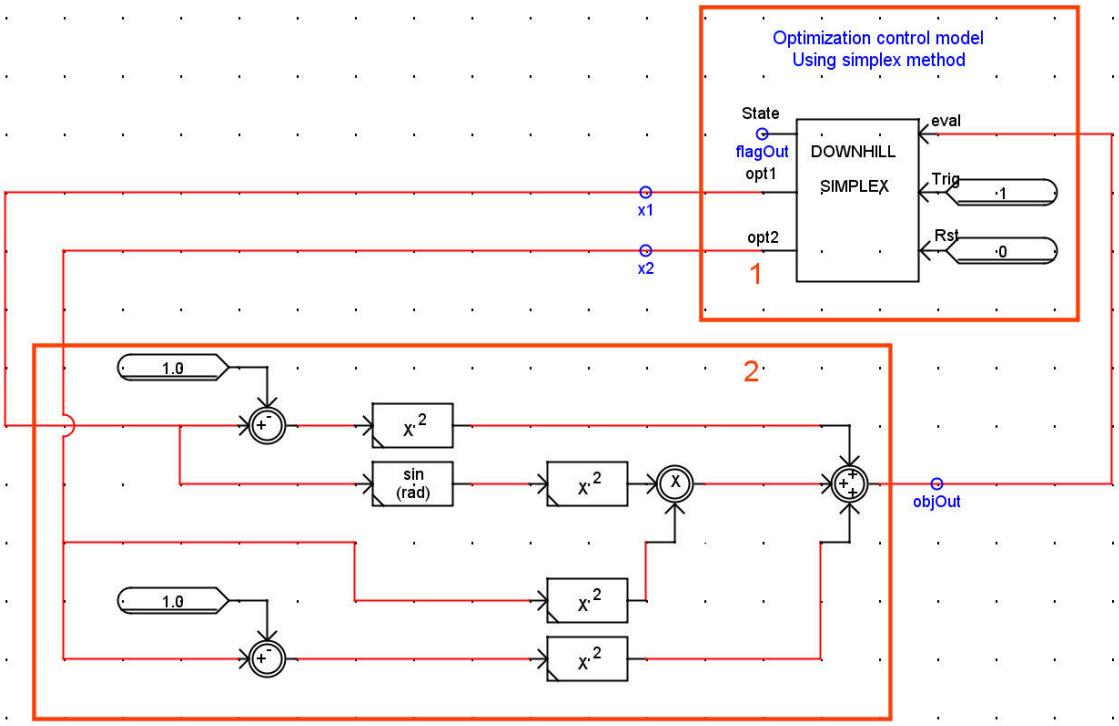

Figure 5.1.1-4 Optimization algorithm in real-time model

In Figure 5.1.1-4, the area marked '1' is the RTDS optimization model which runs the Simplex algorithm; the area marked '2' is the two-variable function presented in Equation (5.1). After compiling and running the case, the Simplex algorithm converged to the minimum value after 71 time steps when the convergence criterion value was 0.0001. Because the NLO implementation is now part of the simulation case and the model has no ability to start and stop each iteration, the relationship between the NLO implementation in area '1' and the OF evaluation in area '2' becomes more complex.

The real-time simulator in this experiment allows the assignment of an execution priority to each of the models in the simulation. Therefore, the NLO implementation in area 1 has the first priority in the execution in time step and the rest of the models, which is the implementation of Equation (5.1) has the lower order in the execution. For example, at the 33rd time step, x1's value was 0.645081 and x2's was 0.793091 from the NLO implementation. Equation (5.1) produced the corresponding output value of 0.396171. The NLO operation at this time step was 'Reflection' (See chapter 3.2). At the next time step, i.e., the 34th time step, the NLO implementation received the result, 0.396171 to determine

the next operation. Because the OF evaluation value was worse than the next worst evaluation output in the Simplex, the next operation was determined as 'Contraction' and the corresponding output values, x1 = 0.793167 and x2 = 0.644684, were produced. Figure 5.1.1-5 clarifies this process further.

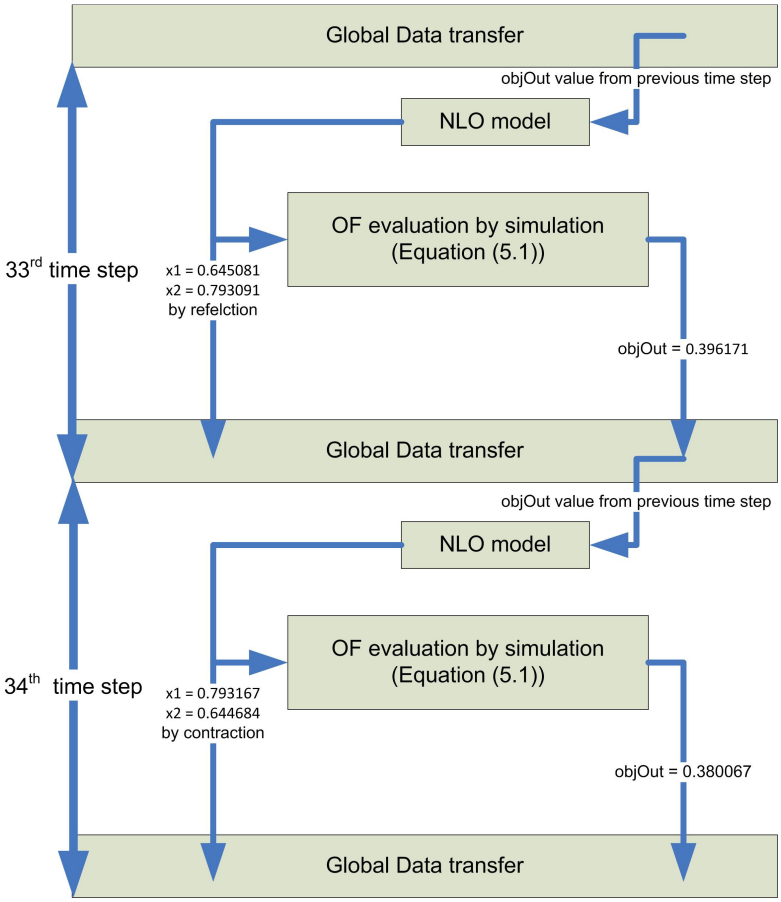

Figure 5.1.1-5 time step progress in the single simulation approach

The results, i.e., the trend in OF evaluation values, from the experiment are presented in Figure 5.1.1-6. The horizontal axis in the figure is the number of time step and the vertical axis is the OF evaluation value.

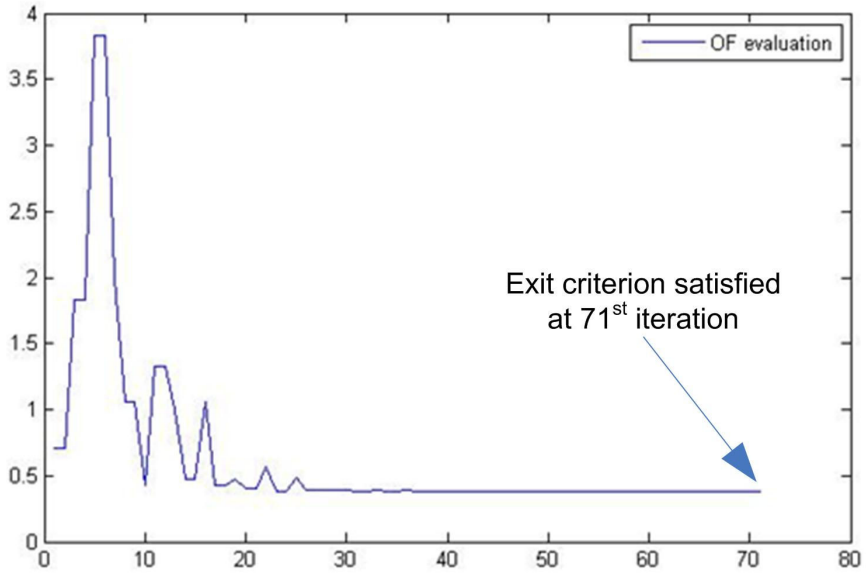

Figure 5.1.1-6 Optimization result by real-time model

The iteration successfully met the convergence criterion after 71 iterations. The first input variable labeled ‘x1’ settled at the value of 0.77357 while the second input variable labeled ‘x2’ came to the final value of 0.670918. Both inputs produced the final function output value of 0.379307. The difference between the final x1 value and the analytical solution is 0.13% and the difference between the final x2 value and the analytical solution is 0.08%. A single iteration took 10 µS. Hence, 710 µS was the total execution time for this optimization process.

The obvious advantage of the approach in this experiment, i.e., using an RTDS building block to implement the NLO algorithm is the savings in total execution time, as communication with an external NLO algorithm is not utilized. Selection of new parameters as well as conducting the OF evaluation was completed within the simulation time step, which was 10 µS. This is order of magnitude faster than the time required for the script implementation, which was 169 seconds.

The disadvantage of this approach is the complexity of the implementation. The effort required to write a building block in the assembly language of the particular processor, i.e., IBM PowerPC 750GX [126], in the real-time simulator was not trivial. The selected real-time simulator provides a much easier environment for that purpose, called CBUILDER, but

still this approach cannot compete with the previous approach in terms of the amount of effort required in the implementation. In addition, another source of complexity in implementation comes from the real-time building block's ineptitude in controlling the start and stop of the simulation case. Thus, the state in the simulated system needs to be initialized at the beginning of each iteration. That initialization has to be coordinated with the NLO implementation as well. Note that as Equation (5.1) has no dynamics, convergence happens in one time step.

In a more practical problem, a much longer simulation time will be required, and because the scripting method in the previous section has a large, but fixed communication delay, it will become a smaller portion of the total time for each simulation run. Consequently, the speed up in the total execution time will not be as dramatic as presented in this experiment. Therefore, the disadvantage associated with the previous approach will diminish as each OF evaluation requires more simulation time.

Hence, this approach was discarded in the remainder of the work presented in this thesis.

5.1.2 Hardware parallelism utilization

5.1.2.1 Genetic algorithm in real-time simulation

This section describes the implementation of a genetic algorithm (GA) optimization algorithm. The same simple mathematical function, presented in Equation (5.1), is used here for the demonstration of the essential concept.

Two different approaches were attempted. The first was similar to that of the down-hill Simplex algorithm described earlier, in that one real-time simulator was used to run simulation cases one at a time, each with its own parameter set. In the second approach, several simulation runs were simultaneously spawned on several different real-time simulators. Each simulation run culminated with an OF evaluation value that signified how close the simulation results matched the desired ones. This way, the inherent parallelism in the GA process, as described in chapter 4.3.2, was exploited.

In this experiment, initially only one EMT simulator (i.e., OF evaluator) was used. Then the number of EMT simulator was expanded by adding more real-time simulators. This expansion continued until the number of simulators consumed all the available hardware. Figure 5.1.2.1-1 presents this algorithm implementation approach. In this particular experiment, the five RTDS simulators served as the 5 parallel OF evaluators.

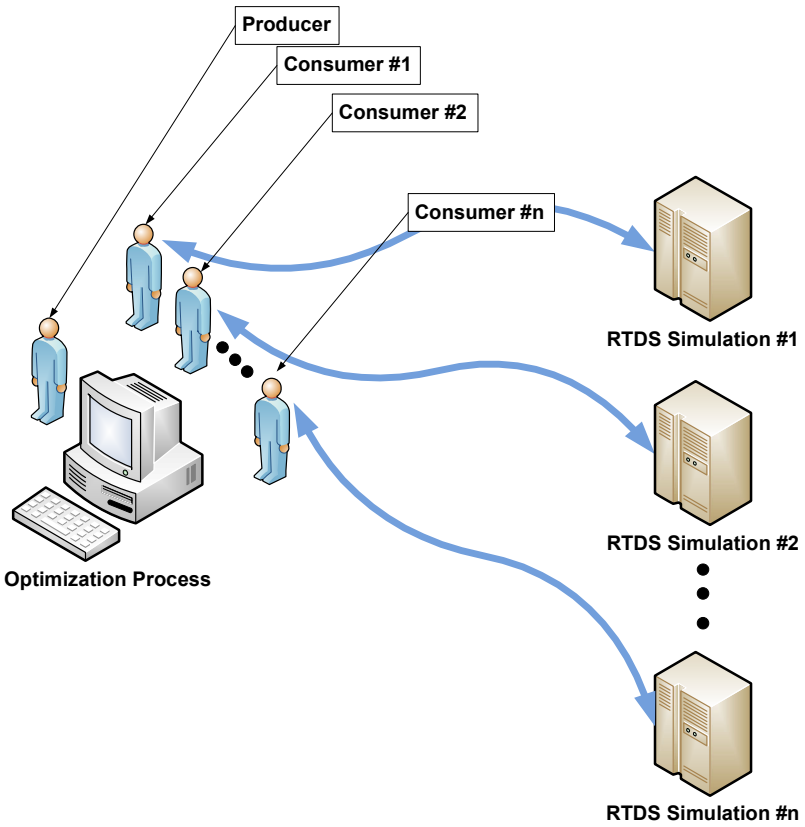

Figure 5.1.2.1-1 Relations between the optimization process and simulation executers

Table 5.1.2.1-1, presents the final function minimum values from the test results.

Number of real-time simulator	The final result (best fitness value)
1	0.379304945
2	0.379337668
3	0.379336536
4	0.379311919
5	0.379370064

Table 5.1.2.1-1 Optimization results with genetic algorithm

As presented in Table 5.1.2.1-1, essentially the same optimized OF minimum was obtained, which agrees well with the theoretical value of 0.3793048.

Table 5.1.2.1-2 presents the computational performance of the test:

Number of real-time simulator	Total Execution time (Minute)	Ratio (N simulators /single simulator)
1	539.0	1.0
2	286.5	0.532
3	249.1	0.462
4	126.1	0.234
5	107.7	0.199

Table 5.1.2.1-2 Execution statistics with genetic algorithm

Figure 5.1.2.1-2 is a graph of total execution time versus the number of simulators available for the OF evaluation. As expected, the total execution time required for optimization drops with more parallel simulators available.

The decreasing total execution time along with more simulators in Table 5.1.2.1-2 suggests that the large amount of required simulation time, which can become prohibitive in some practical applications, can be reduced substantially when more resources are utilized as the object function evaluator in the optimization process.

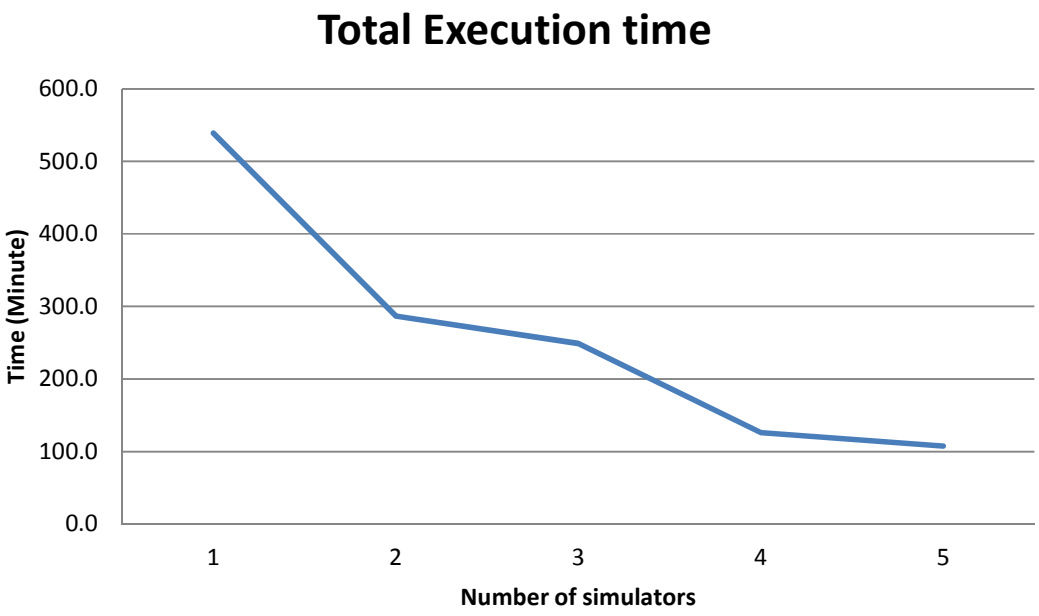

Figure 5.1.2.1-2 Total execution time versus number of simulation executers (with a Genetic algorithm)

Figure 5.1.2.1-3 presents the decreasing trend of the total execution time with the addition of more simulators. This trend is plotted as the solid line. For comparison, the ideal trend, i.e., a linear increase in computation speed with increasing number of simulators, was plotted as the dotted line. Note that a linear increase in speed corresponds to a hyperbolic curve for the reciprocal of the number of simulators. As can be observed from the figure, the ratio of the execution time required for a single simulator to total execution time for multiple simulators in Table 5.1.2.1-2 does not reflect precise inverse linear relationship with the number of simulators. This is because of the unpredictability in the number of OF evaluations each time the optimization algorithm is run, due to the random nature of Genetic Algorithm (See chapter 3.3).

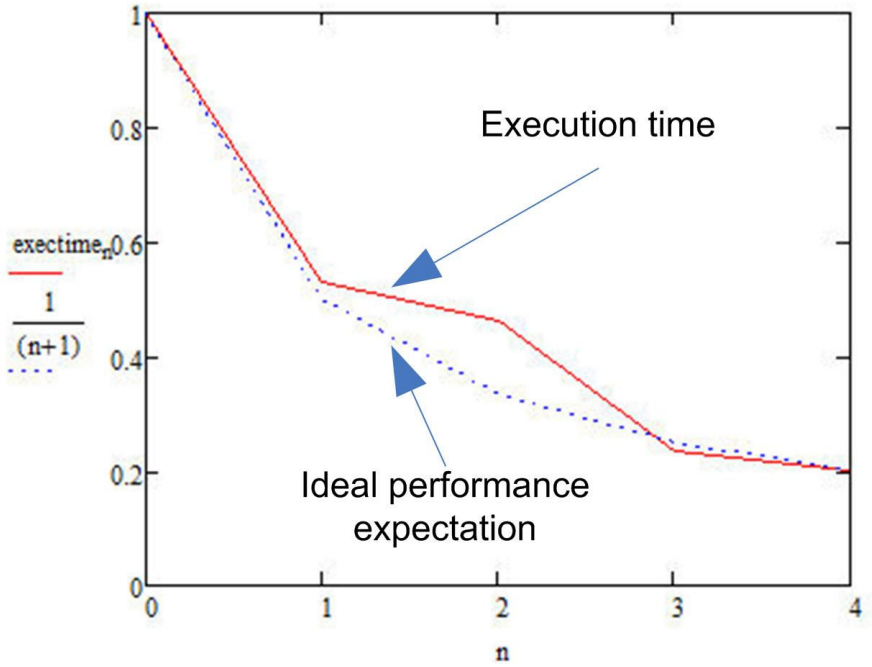

Figure 5.1.2.1-3 Trend of GA processing time with more real-time simulators

Although the example considered a simple arithmetic function, i.e., Equation (5.1), used for the demonstration purpose, the real-time simulator can run any required simulation in real-time for evaluating the OF. The real-time simulation execution statistics indicates that the complexity or the size of a simulation case will not have any effect on the total amount of time required by a certain optimization problem solving. This execution time independency is another distinctive feature available only in the real-time simulation environment.

5.1.2.2 Particle Swarm Optimization (PSO) algorithm in real-time simulation

The simple two-variable function, evaluated in 5.1.1 and 5.1.2.1 by the down-hill Simplex algorithm and the Genetic Algorithm (GA) respectively, is considered again using the Particle Swarm Optimization (PSO) algorithm in order to search for the minimum value. By

comparing different optimization algorithms with an identical target function, the behavior of each of those optimization algorithms can be evaluated and compared.

As before, the implementation was attempted on various numbers of OF evaluation platforms, ranging from 1 to 5. Table 5.1.2.2-1 shows the optimization results using PSO on different number of real-time simulators.

Number of real-time simulator	The final result (best fitness value)
1	0.37930489
2	0.37930489
3	0.37930489
4	0.37930489
5	0.37930489

Table 5.1.2.2-1 Optimization results with PSO algorithm

As presented in Table 5.1.2.2-1, all the tests with a different number of objective function evaluators (i.e., real-time digital simulators in this experiment) converged to the correct final value, close to the theoretical result (0.3793048). Furthermore, the final results from all the different trials resulted in a practically identical value. This closeness of the final output is slightly different from the experiment results obtained by the Genetic Algorithm. This observation suggests that the Particle Swarm Optimization algorithm can be more consistent with respect to the final value than the final result from the Genetic Algorithm.

Table 5.1.2.2-2 summarizes the computational performance of the optimization implementation. The total computation is faster, in general, as more real-time simulators were utilized. However, the randomness in the procedure results in different number of OF evaluations before the exit criteria was satisfied.

Number of real-time simulators	Number of Iterations	Total execution time (Minute)	Ratio (N simulators/single imulator)
1	4880	421.9	1.0
2	6600	387.6	0.919
3	4480	188.9	0.448
4	4560	185.0	0.438
5	5880	211.4	0.501

Table 5.1.2.2-2 Execution statistics with PSO algorithm

Figure 5.1.2.2-1 summarizes the relationship between total execution time and the number of real-time simulations. As was observed in the previous chapter, the benefits of using multiple simulators as object function value evaluators in parallel is clearly suggested in this plot.

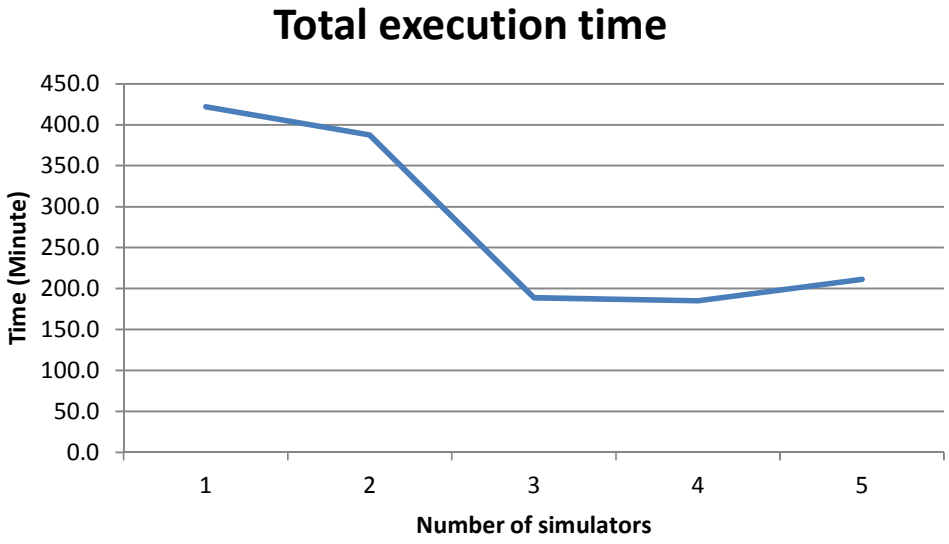

Figure 5.1.2.2-1 Total execution time versus number of real-time simulators (with PSO algorithm)

In Figure 5.1.2.2-1, one noticeable trait of the curve is that the total execution time took longer when 5 simulators were used rather than when 4 simulators were used. The reason becomes clear when the total number of iterations is compared. When the number of

simulators was 5, the total number of simulation was 5880; whereas the corresponding number of 4 simulators was 4560. More than 1000 iterations were required when 5 simulators was used than when 4 simulators were used.

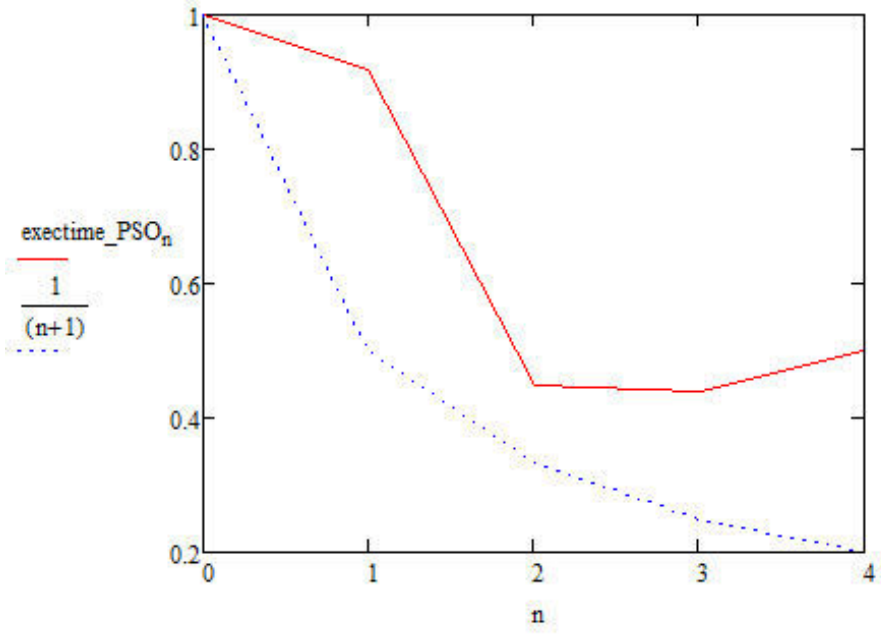

Figure 5.1.2.2-2 Trend of PSO processing time with more real-time simulators

Figure 5.1.2.2-2 presents the trend of the PSO processing time with addition of more simulators. This trend is plotted in solid line. In the same way as in Figure 5.1.2.1-3, the ideal trend, linearly decreasing trend as more simulators were added, was plotted in dotted line as well. The deviation of the trend from the ideal curve is more pronounced here than the curve in the case of GA implementation. The same unpredictability, as discussed in the previous GA case (in chapter 5.1.2.1) can be ascribed as the reason. A different result would be obtained if the experiment were repeated. Statistically it is likely (and appears to be reasonable) that the speed of the OF evaluation would increase with more simulators. Due to the limited time availability of sufficient number of simulators and due to the very large convergence time, the author was unable to conduct such statistical experimentation in this case.

5.2 Power system case studies

5.2.1 HVDC controller tuning

5.2.1.1 Down-Hill Simplex method application

An HVDC transmission system real-time simulation case with a software controller was selected in order to evaluate the developed optimization technique. Parameters for this case were similar to that of the 200 MW Blackwater back to back HVDC system [127], installed between Texas and New Mexico, in the United States. Because the system is a commercial project, no details regarding the control were available for the modeling. Hence, a generic control system was constructed based on the CIGRE HVDC benchmark case [128].

The Draft of the case is presented in Figure 5.2.1.1-1, Figure 5.2.1.1-2, Figure 5.2.1.1-3 and Figure 5.2.1.1-4. On the rectifier side, the 12 pulse monopolar converter is connected via a transformer to the ac filters and via a 100 km 230 kV transmission line to a remote source. On the inverter side, the transmission system is more complex, with a connection of 2 3-phase 180 km 345 kV transmission lines and an intermediate load. The 11th and 13th harmonics filter provide 26.5 MVar and the high pass filter provides the same value of reactive power. The rectifier and inverter controls are all based on simple PI (Proportional-Integral) regulators. The rectifier is operated in current control mode and regulates the DC current. The inverter operates in constant extinction angle (CEA) control mode and regulates the extinction angle (γ) [129].

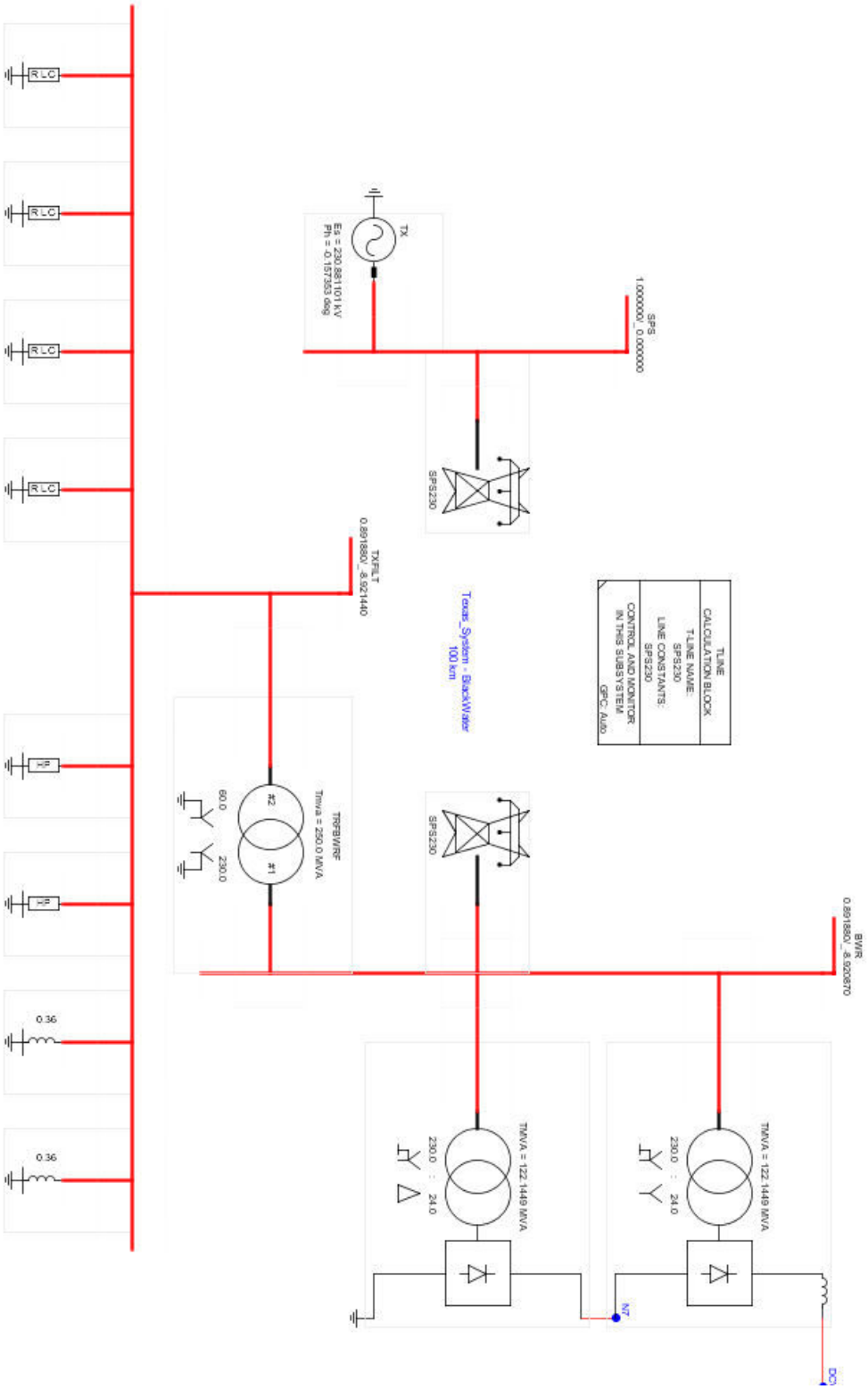

Figure 5.2.1.1-1 An HVDC transmission system: Rectifier side

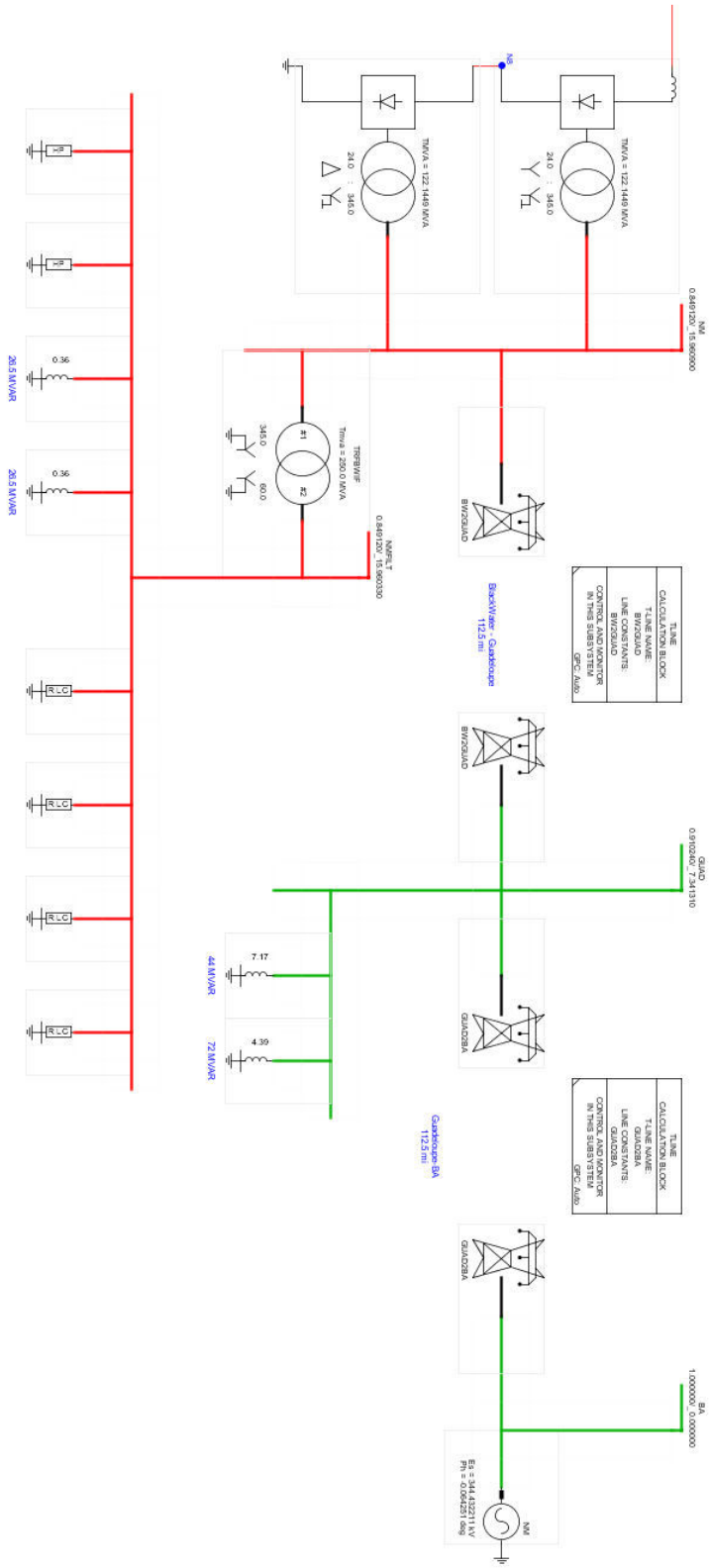

Figure 5.2.1.1-2 An HVDC transmission system: Inverter side

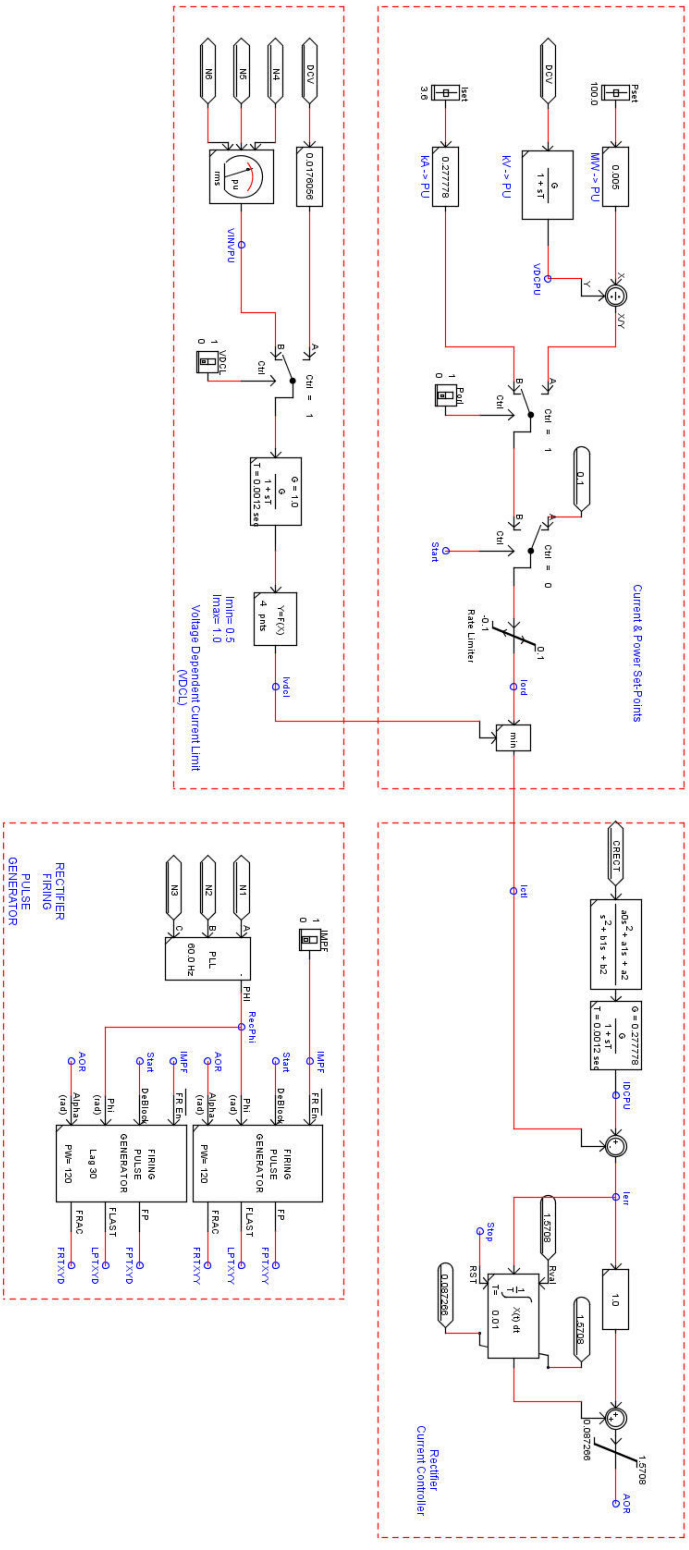

Figure 5.2.1.1-3 An HVDC transmission system: Rectifier controls

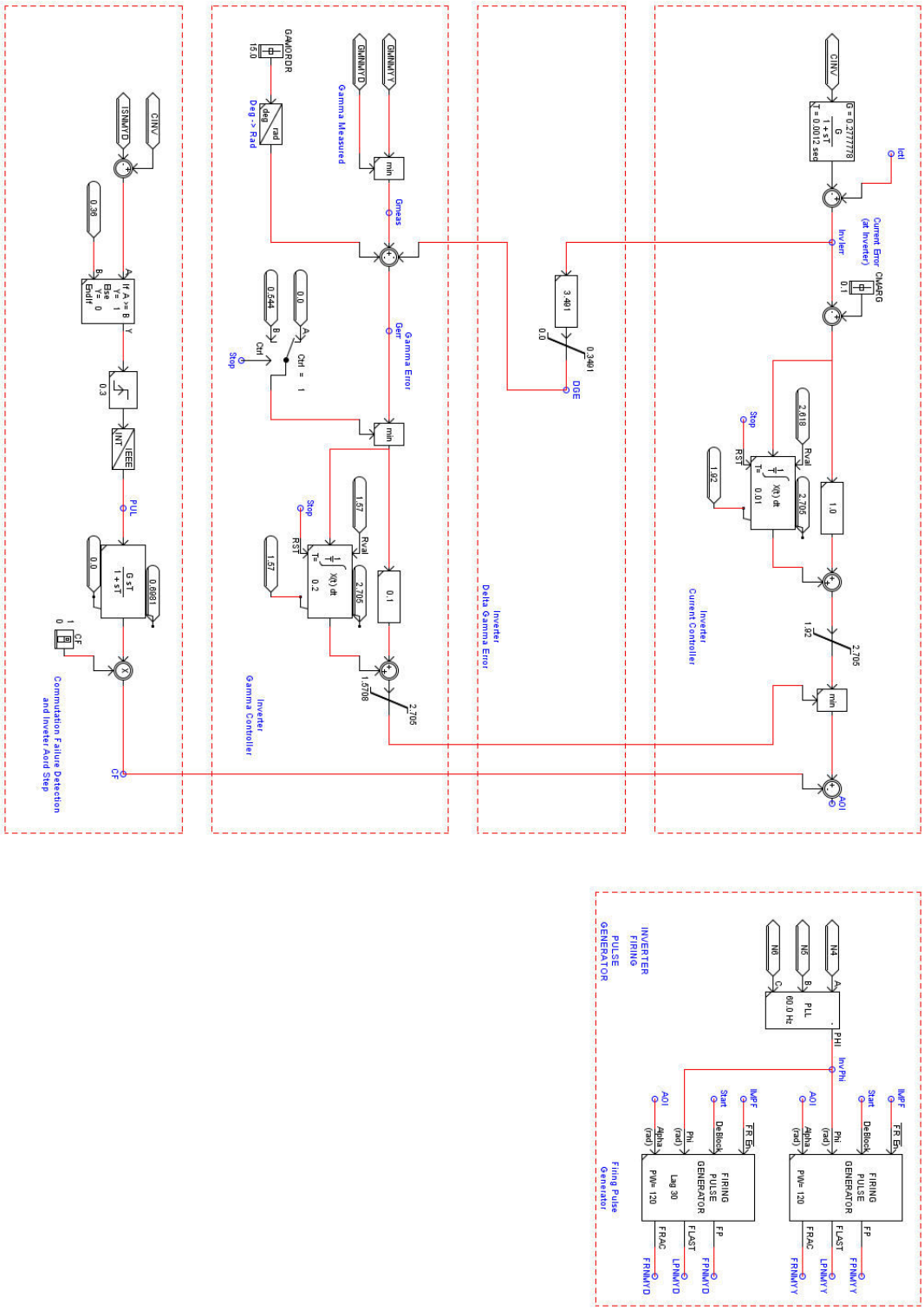

Figure 5.2.1.1-4 An HVDC transmission system: Inverter controls

The controller parameters in the rectifier current controllers, (two in total), were chosen as control variables for the optimization. The chosen parameters were as follows: proportional gain and integrator time constant for rectifier current control PI controller (Kp1, Ti1). The objective of the optimization was to attain the minimum transient and steady state deviation between the ordered and the actual DC currents. Hence, the object function for the optimization in this case can be written as:

OF = ISE (Integral Square Error)

$$= K_1 \int_0^{Simulation\ end} \left(I_{measured} - I_{order}\right)^2 dt \tag{5.5}$$

where

K_1: *Weighting factor*

$I_{measured}$: *Measured DC current*

I_{order}: *Ordered DC current*

The optimization problem in this test case can be stated as follows:

Minimize OF(Kp1, Ti1)
Subject to
Kp1 > 0, Ti1 > 0

The real-time simulator hardware used spanned two generations. Thus, there was a mix of computing devices that included 12 digital signal processors (Analog Device ADSP-21062 [130]) and 2 conventional microprocessors (IBM PowerPC 750CXe [131]). The necessary calculations for the power system models in the simulation were made on the DSPs, while the network solution and control model calculation were executed on the PowerPC microprocessors. The latest real-time hardware generation utilizes a newer PowerPC (the Freescale MPC7448 [132]) exclusively.

The pre- and post-optimization waveforms of DC current are presented in Figure 5.2.1.1-5. The desired value of DC current was 1.0 PU. Before optimization, a steady state error appeared, indicating insufficient gain in the integral regulator. The integral regulator will eventually always result in zero steady state error for a constant reference, but the action of the integrator may take a long time if the integral regulator gain is insufficient. The magnitude of oscillation in the 'Before optimization' waveform also presents high steady state oscillation in measured current. This is due to the fact that the proportional gain was too high and thus, the controller modulated the firing angle with the oscillation frequencies and thereby sustained them. After the optimization was completed and corresponding controller parameters were tuned, the steady state error and the oscillation were minimized as presented in the 'after optimization' waveform.

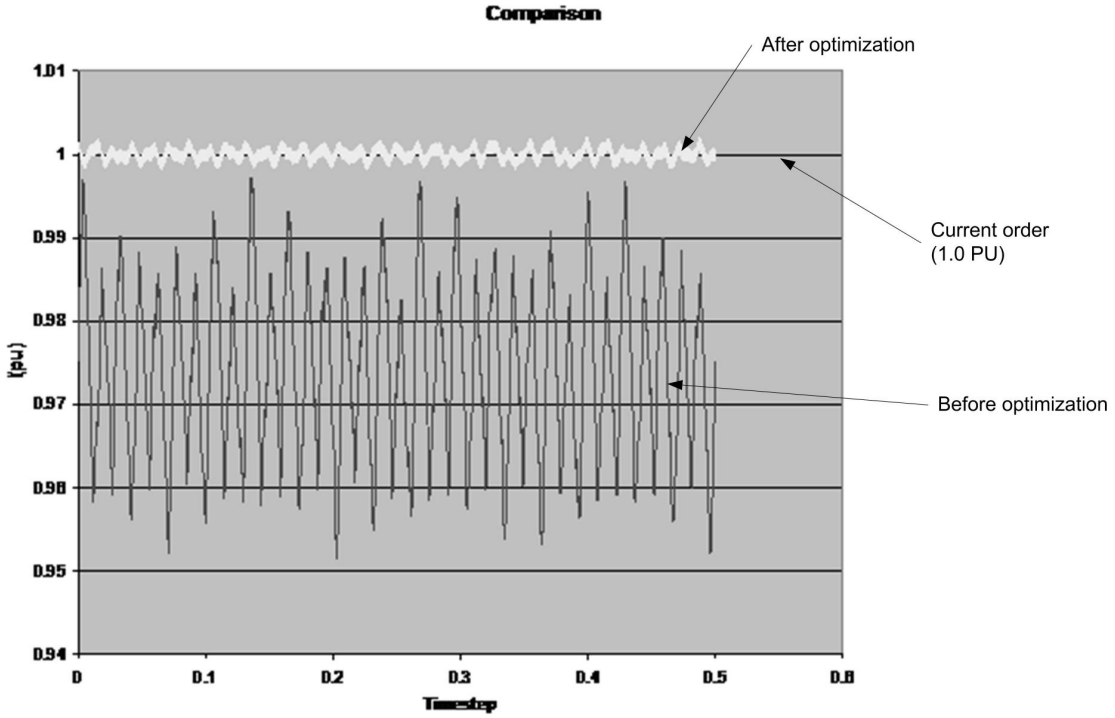

Figure 5.2.1.1-5 Steady state current waveform comparison

The down-hill non-linear Simplex algorithm (non-linear Nelder-Mead Simplex algorithm), described in chapter 3.2, was used to optimize the selected parameters in this experiment. The steady state current, which closely matches the given order with much less

fluctuation was obtained through the optimization process. The effectiveness of the OE-EMT application in this experiment is evident as presented in Figure 5.2.1.1-5.

The HVDC simulation case was revisited with a different objective. Previously, the steady state error was the only focus of the optimization. In this new experiment, the current order was changed in order to improve the performance of the controller under the transient state. The start time of the controller and ISE accumulation was 0.5 seconds after the start of simulation. The end time of the ISE accumulation was 5.5 seconds. Hence, the ISE value was integrated during a period of 5.0 seconds. The current reference was ramped from 1.0 PU to 1.1 PU at a rate of 0.5 PU per second. The ISE calculation was the same as in the previous case.

The down-hill Simplex algorithm successfully converged after 17 iterations. The convergence criterion was 0.01. Figure 5.2.1.1-6, presents the current waveform comparison between the current order, the current waveform before optimization and the current waveform after optimization. It is noted that the current waveform after optimization settled on the current order with shorter time and less overshoot.

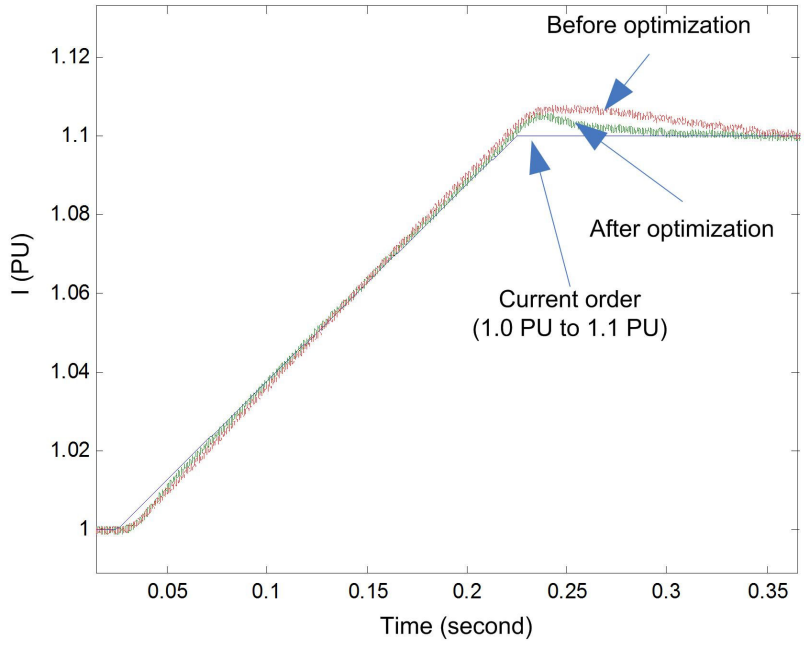

Figure 5.2.1.1-6 Transient state current waveform comparison

Comment on selection of initial condition

The rate of convergence is influenced by the coordinates of vertices in the simplex and the initial step length. Also, initiating the optimization process at a different initial simplex may lead to different local optimums. It is advisable that the user attempts to do several optimizations starting from widely different initial conditions. Some of the converged simplexes may be better than others.

5.2.1.2 Comparison of performance between a real-time simulator and an off-line simulator

The performance of the real-time simulation based optimization was compared with a similar model in off-line environment (PSCAD). As before, the down-hill Simplex optimization algorithm was utilized for the purpose of the controller parameter optimization. The same parameters as in the real-time implementation, i.e., time constants and gains in the PI controllers, were chosen as the optimization variables. The same optimization goal, i.e., minimizing the difference between the current order and the measured current, was set. Thus, the same objective function as was presented in the previous chapter was evaluated by the off-line simulation iterations.

The elapsed time for the optimization was measured and compared with that of the real-time simulation. Figure 5.2.1.2-1 presents this comparison. Each OF evaluation required simulation of a 20.5 second long simulation (real-time). In the off-line simulation, each such evaluation required 64.3 seconds whereas with the real-time simulation only 24.8 seconds were required. In this comparison, the actual computation time requirement of the real-time simulation was exactly 20.5 second because the simulation runs in real-time. The remainder of the time consumption was used by the simulation preparation tasks such as case downloading which are not related to the actual power system simulation calculation. This comparison suggests that as the number of iterations increases, the benefit of using a real-time simulation environment for the OE-EMT application becomes more evident, albeit at a great cost.

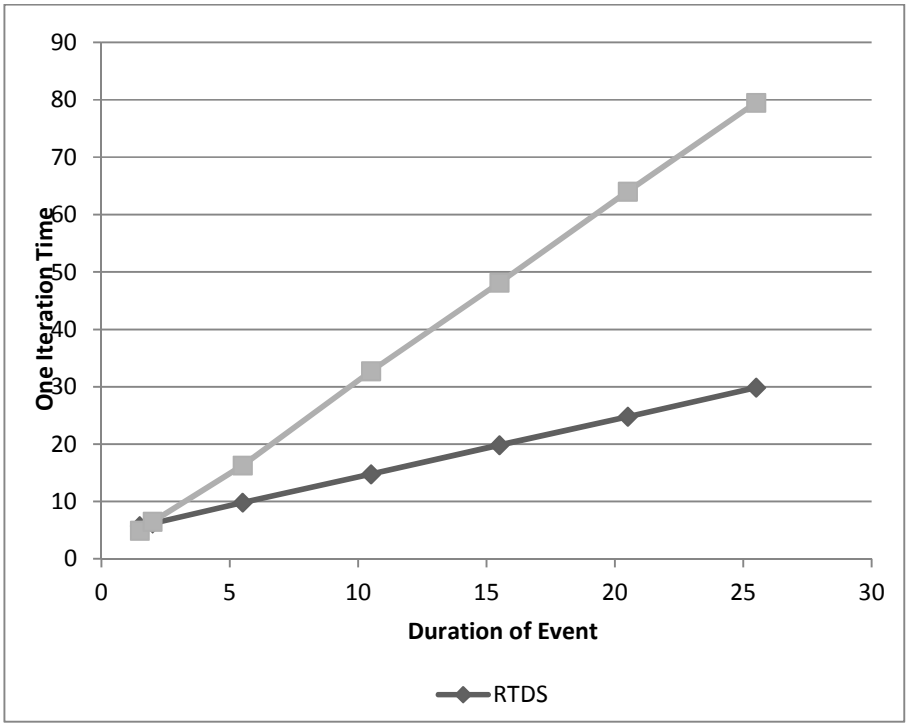

Figure 5.2.1.2-1 Comparison of execution times

5.2.1.3 Parallelism application: a Genetic Algorithm

The next step after the previous power system case study was the application of one of the heuristic-based optimization algorithms, i.e., a genetic algorithm method, to a real power system optimization problem. The system modeled in the 1st CIGRE HVDC benchmark model [128] was selected as a target simulation case. This simulation case models a ±500 kV, 1000 MW HVDC transmission connecting two AC systems with ESCR of 2.5 at the rectifier and inverter side. Data for system parameters can be found in [133, 134] The behavior of this system has been widely studied and is thus well known, and the computational power requirement of the simulation case is moderate. Figure 5.2.1.3-1 presents an outline of the RTDS simulation case.

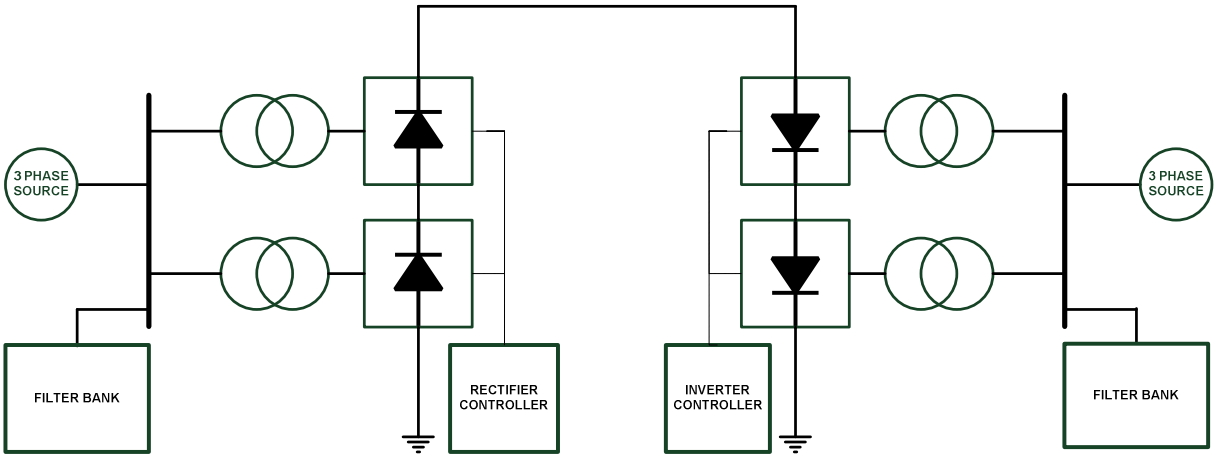

Figure 5.2.1.3-1 CIGRE HVDC benchmark simulation case: an overview

Figure 5.2.1.3-2 depicts the rectifier side of the CIGRE HVDC benchmark RTDS simulation case and Figure 5.2.1.3-3 presents the inverter side of the simulation case.

As in the previous example, the rectifier is operated in current control and the inverter side in constant extinction angle (γ) control modes. Both major controls are based on simple PI (i.e., Proportional-Integral) regulators. The details of the control block diagram can be found in Figure 5.2.1.3-4 and Figure 5.2.1.3-5.

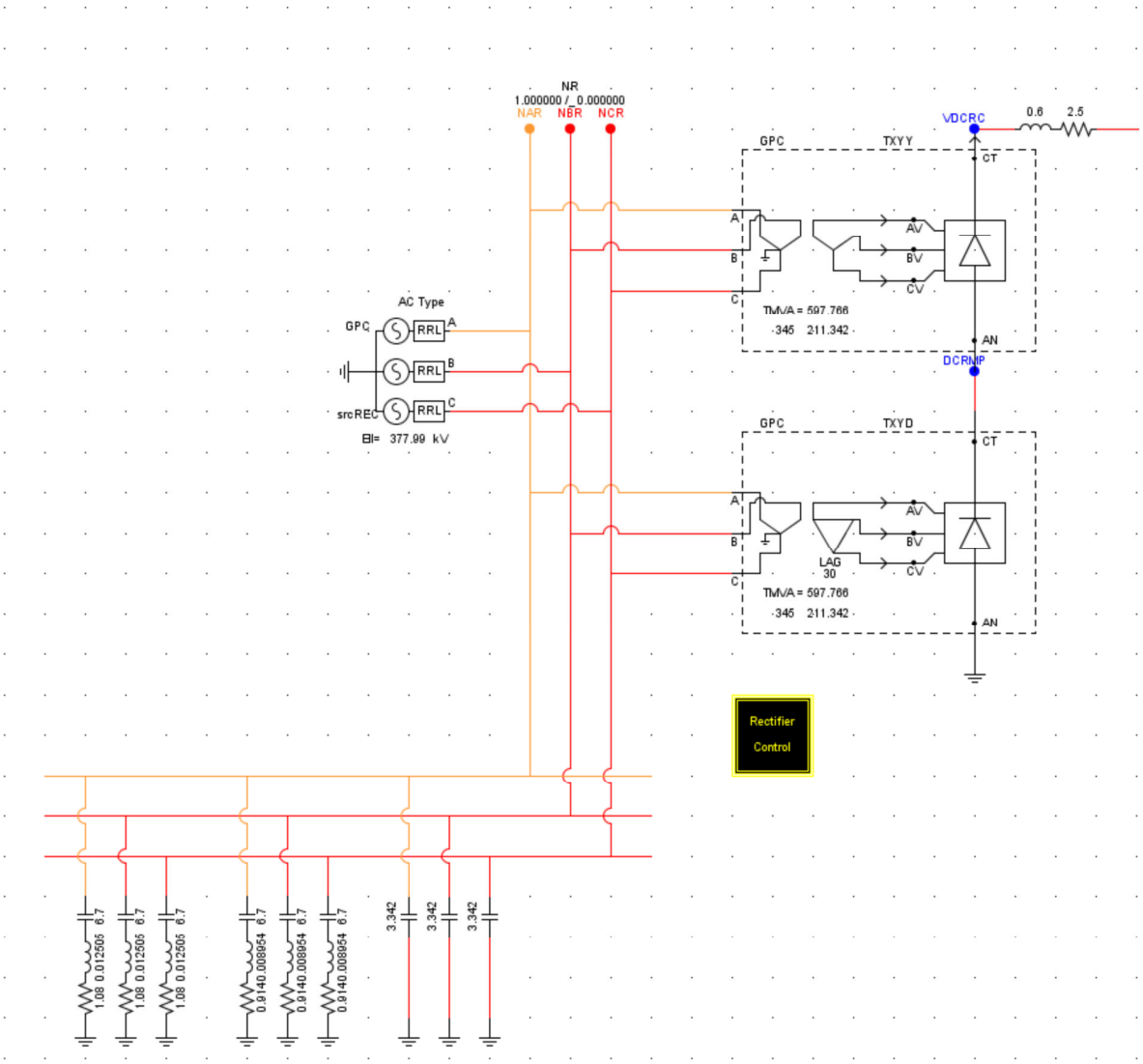

Figure 5.2.1.3-2 CIGRE HVDC benchmark system real-time simulation case: Rectifier

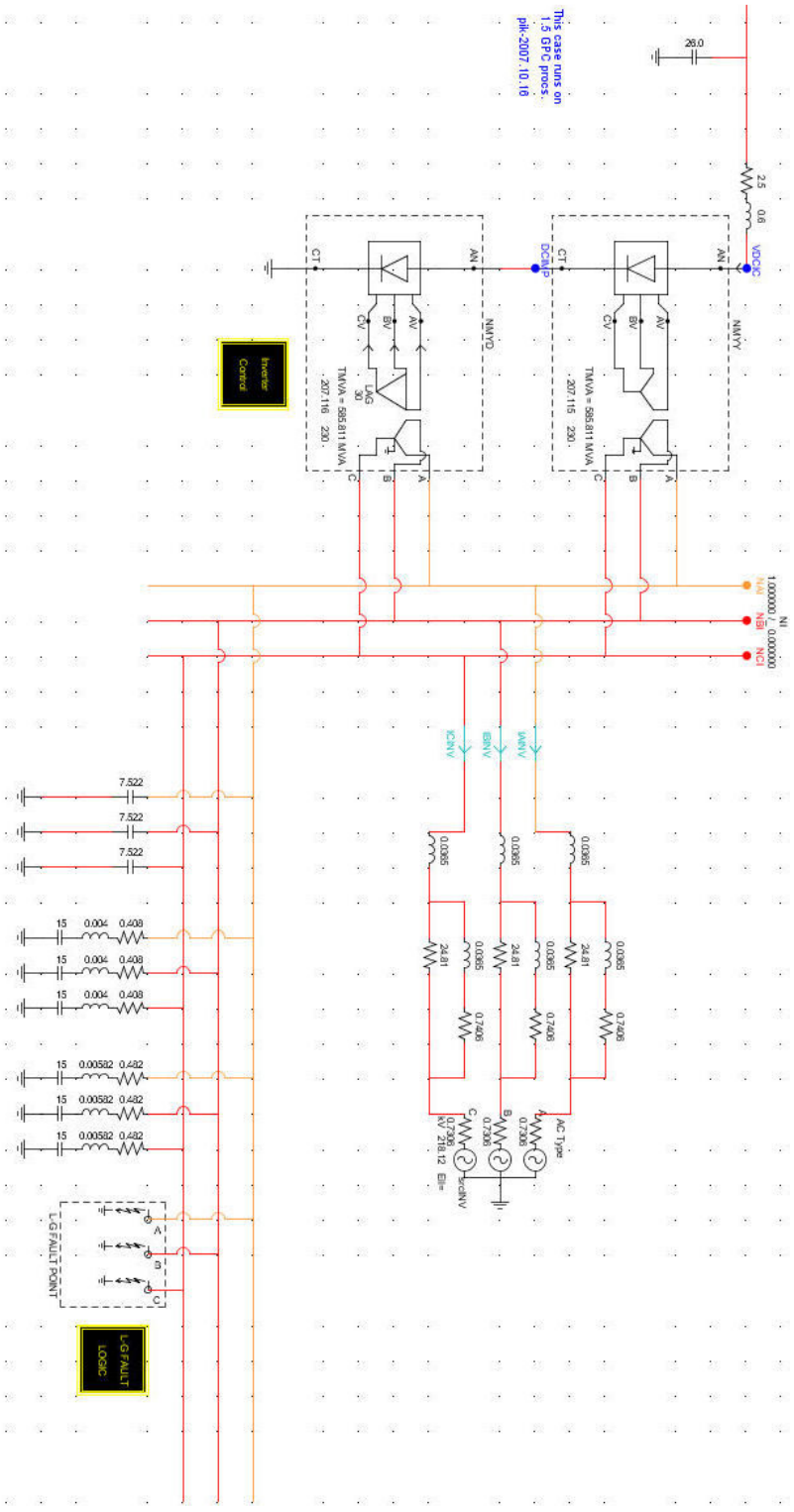

Figure 5.2.1.3-3 CIGRE HVDC benchmark system real-time simulation case: Inverter

The selection of the optimum PI regulator proportional gain and integrator time constant becomes the controller parameter tuning problem, which is solved by the optimization process (i.e., Genetic Algorithm) implementation and real-time simulation iterations. The time constant and the gain in the PI regulator of the rectifier controller are the first two controller parameters. The time constant and the gain in the similar PI regulator in the inverter CEA controller constitute the other two parameters. In total, four parameters were steered by the optimization algorithms until the algorithm located the final values which will meet the design objective(s).

The design objective of this controller parameter optimization problem is to match the DC current flow to the given order with minimum error. This imperative for the controller parameter optimization was embedded in the cost (i.e., objective) evaluation function aspect of the simulation. The control parameter search also has to be confined to a feasible region. Hence, a penalty term is added to the objective function, when the solution from the optimization algorithm makes an excursion into the non-feasible region, such as when the integral regulator time constant or proportional regulator gain becomes negative.

In particular, the object function for the optimization in this case is:

OF = ISE (Integral Square Error)

$$= K_1 \int_{T_{start}}^{T_{end}} \left(I_{measured} - I_{order} \right)^2 dt \tag{5.6}$$

where

K_1 : *Weighting factor*

$I_{measured}$: *Measured DC current*

I_{order} : *Ordered DC current*

The optimization problem in this test case can be stated as follows:

Minimize OF(Kp1, Ti1, Kp2, Ti2)

Subject to

Kp1 > 0, Ti1 > 0

Kp2 > 0, Ti2 > 0

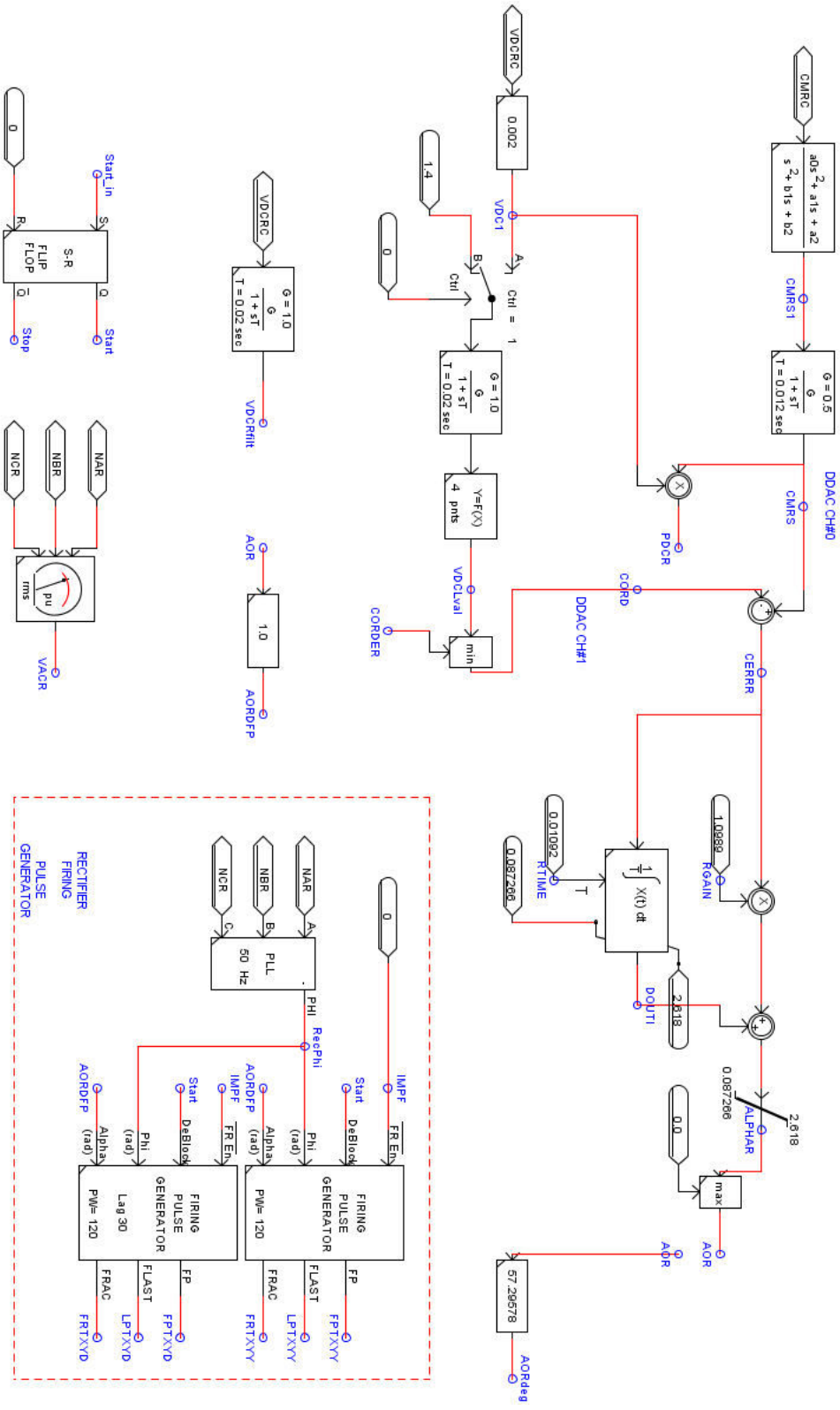

Figure 5.2.1.3-4 CIGRE HVDC benchmark system: Rectifier controller

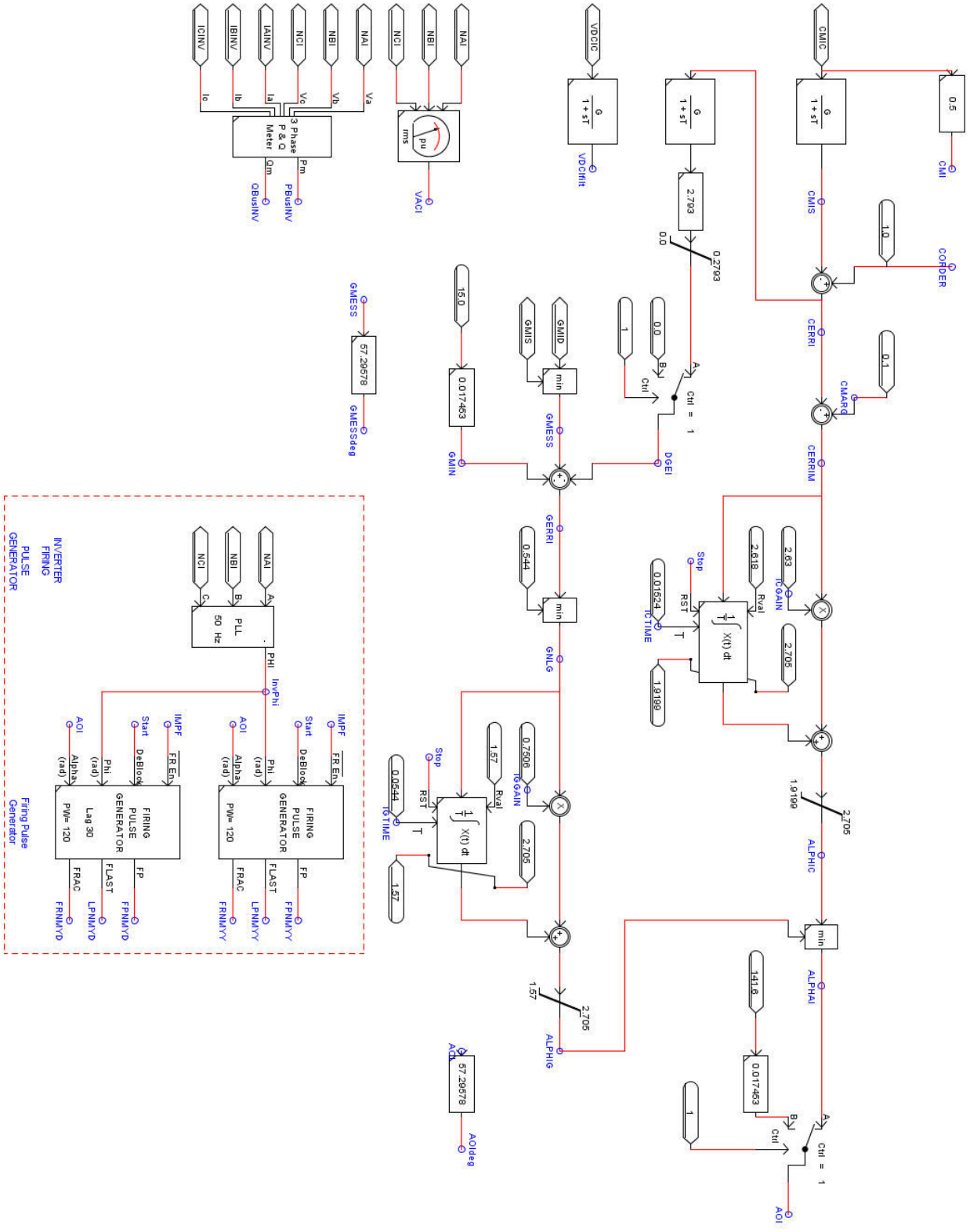

Figure 5.2.1.3-5 CIGRE HVDC benchmark system: Inverter controller

The implementation of the cost function in the real-time simulation case is presented in Figure 5.2.1.3-6.

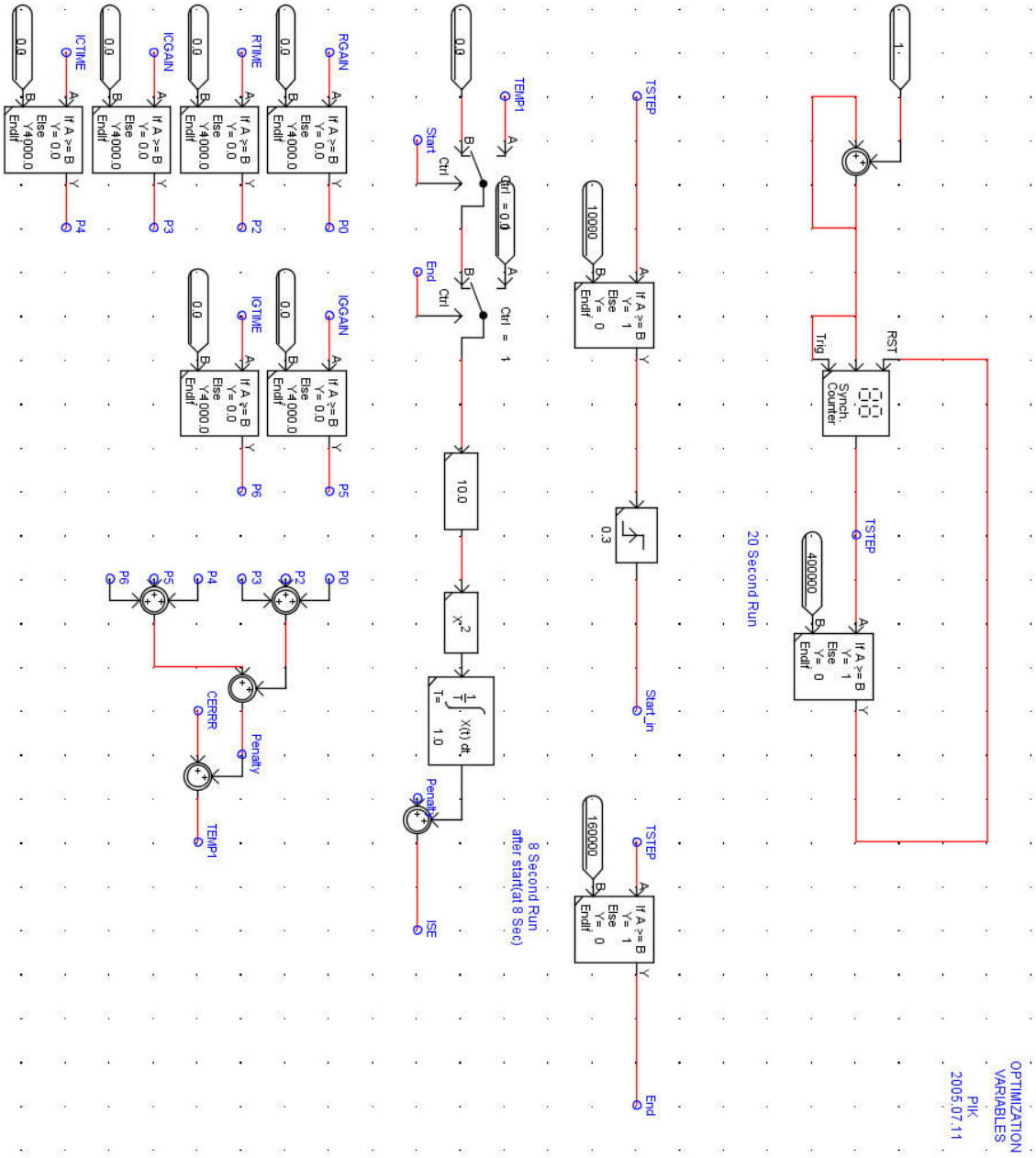

Figure 5.2.1.3-6 Cost (Objective) function implementation

The optimization experiment was performed in a manner similar to the other optimization problem solutions presented in chapter 5.1.2.1. First, the CIGRE HVDC

benchmark real-time simulation case was executed in a single real-time simulator using the GA (Genetic Algorithm) approach. The GA has built-in parallelism, which was explained in chapter 3.3. This parallelism allows each individual simulation, which is from the chromosome candidate in a population generated by the GA implementation, to be evaluated independently and simultaneously. This characteristic of the algorithm offers the possibility of considerable speed up if more real-time simulators can be deployed for the purpose of parallel execution of simulation instances. Table 5.2.1.3-1 presents the results of this experiment. The number of simulation evaluators was increased from 1 to 5. When multiple real-time simulators were engaged, simultaneous simulations (each in real-time) and optimization candidate evaluations became possible. The data in Table 5.2.1.3-1 presents a comparison of execution times, indicating a substantial savings in time with more real-time simulators, i.e., objective function evaluators. The table also provides a comparison of the time per iteration for the various cases, resulting from using a single objective function evaluator to using five evaluators.

Number of real-time simulators	Total number of OF evaluations	Total Execution time (Minutes)	Average execution time per OF evaluation (Seconds)
1	2844	1337.4	28.22
2	2205	536.2	14.59
3	4356	744.6	10.26
4	5753	722.7	7.54
5	1176	171.3	8.74

Table 5.2.1.3-1 Execution statistics – controller tuning with GA

The total elapsed time generally decreased in an inverse-linear manner as in the previous experiment, presented in chapter 5.1.2.1. This reduction in time was expected because the individual iterations required less and less time as more real-time simulation evaluators conducted the necessary simulation computation in parallel. However, due to the built-in randomness in the algorithm, some optimization sequences reached the exit criterion with a fewer number of OF evaluations.

Table 5.2.1.3-2 summarizes the evolution of that cost function evaluation output as the proposed optimization process managed the controller parameters. The solutions converge to approximately the same optimal value around 2, with widely varying worst evaluation results. The similarity of the final results suggests that the number of objective function evaluators in the GA application doesn't cause significant deviations in the final optimization result.

Number of RTDS racks	The worst result	The final result (best fitness value)
1	199.1300049	2.07061052
2	232.0108337	2.25619364
3	472.5647583	2.02886605
4	150.4537048	2.21859479
5	332.7230225	2.31824899

Table 5.2.1.3-2 Optimization results – controller tuning with GA

Figure 5.2.1.3-7 presents the relationship between the total execution time and the number of real-time simulators which participated as OF evaluators. Many heuristics-based optimization algorithms have an unpredictable nature in the total number of objective function evaluations. In the case of using three simulators, the total execution time increased from the use of two simulators, which was an unexpected result contrary to intuitive expectation. This increase was due to the increased number of evaluations when three simulators were used. The resulting number was 2205 when two simulators were used, but that number increased to 4356 with three simulators, a more than 90% increase. Thus, the increase in the number of evaluations resulted in the total execution time increase. Comparing the number of evaluators and the average execution time reveals an intriguing aspect. The average execution time increased in the case of five simulators from the previous case where four simulators were used. This increase resulted from the termination of the algorithm after a smaller number of iterations in the 5 simulators case, which caused fewer simulations to be used in calculating the average. The number of OF evaluations in the 5-simulators experiment was 1176. However, that corresponding value, i.e., the number

of OF evaluations, was 5753 when 4 evaluators were engaged. One possible reason includes the assumption that the optimization algorithm is not fully parallelized amongst the participating real-time simulators. In this particular implementation, the OF evaluation was executed on RTDS, while the graphical interface of RTDS (RSCAD) and the GA algorithm were executed on a separate PC with a single CPU. Thus, the computation at the PC side was not parallelized. However, this issue was not pursued further in this research because i) the primary purpose of the demonstration, parallelism utilization in the real-time OE-EMT method, was achieved, and ii) the necessary information, i.e., fine-resolution measurement of the timing at each steps of the algorithm execution and OF evaluation, was not obtainable.

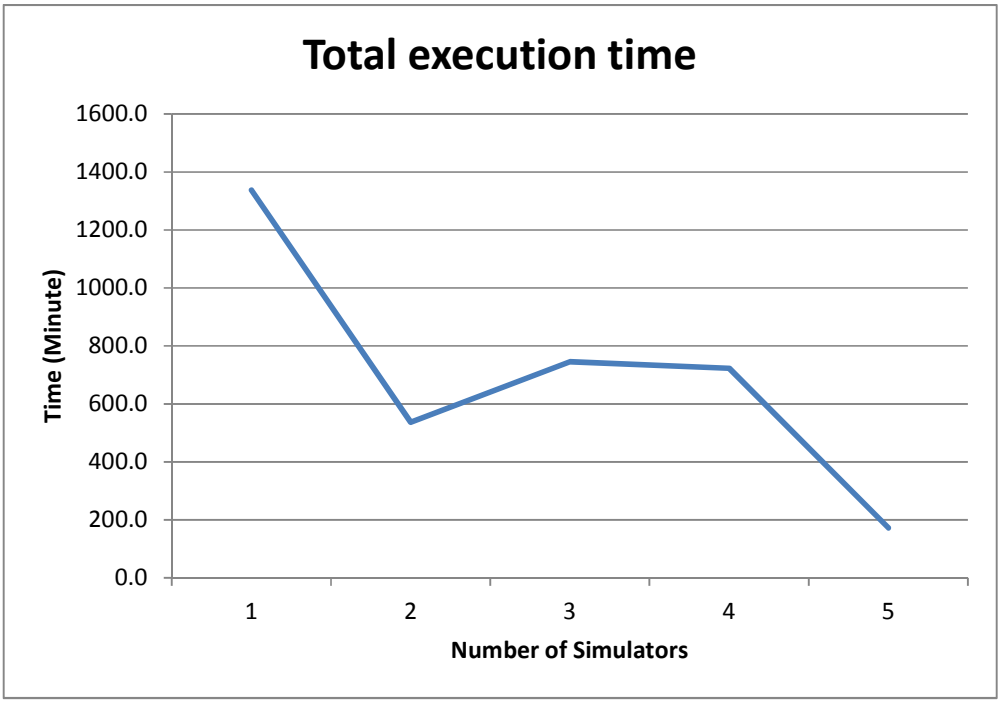

Figure 5.2.1.3-7 Total execution time versus number of simulation evaluators

6 OPTIMIZATION INCLUDING HILS (Hardware-In-Loop Simulation)

One contribution this thesis claims is a procedure to directly optimize actual control hardware using the optimization approach described in earlier chapters. In previous experiments, the controllers, which were the target of the optimization, were software models. The software model was part of the real-time simulation case; thus, the experiments were not interfaced with real world devices. In the experiments presented in this chapter, the actual real hardware controller was engaged, a process requiring interfacing between the real-time simulation and the controller and the interface between the control parameters (which are the subjects of the optimization algorithm) and the optimization algorithm implementation.

6.1 Hardware power electronics controller optimization using the real-time OE-EMT method

In order to evaluate the effectiveness of the proposed approach, an experimental setup was established. A simple DC/DC converter circuit (Buck converter) simulation case was made in a real-time simulation environment. In this experiment, RTDS with its small time step simulation capability [135] was selected as the real-time simulation environment. In the Buck converter circuit configuration, as presented in Figure 6.1-1, a small voltage drop across the storage element (i.e., an inductance in this simulation case) contributes to the decrease in the final output voltage. Thus, the voltage drop across the inductor reduces the voltage output from the Buck converter to slightly less than the expected value based on the given duty ratio. In order to compensate for the voltage drop and to make the voltage output follow precisely the duty ratio command, a simple Proportional Integral regulator (PI regulator) was inserted between the duty ratio command and the final voltage output. The regulator receives the duty ratio command as a reference and the voltage output as a feedback signal. Then, the regulator tries to control the actual duty ratio being provided to the switching element in the power circuit with the aim of matching, as closely as possible, the final output voltage with the duty ratio command. In this experiment, a nonlinear

optimization algorithm, the down-hill Simplex algorithm (Nelder-Mead Simplex algorithm), was selected as the optimization algorithm.

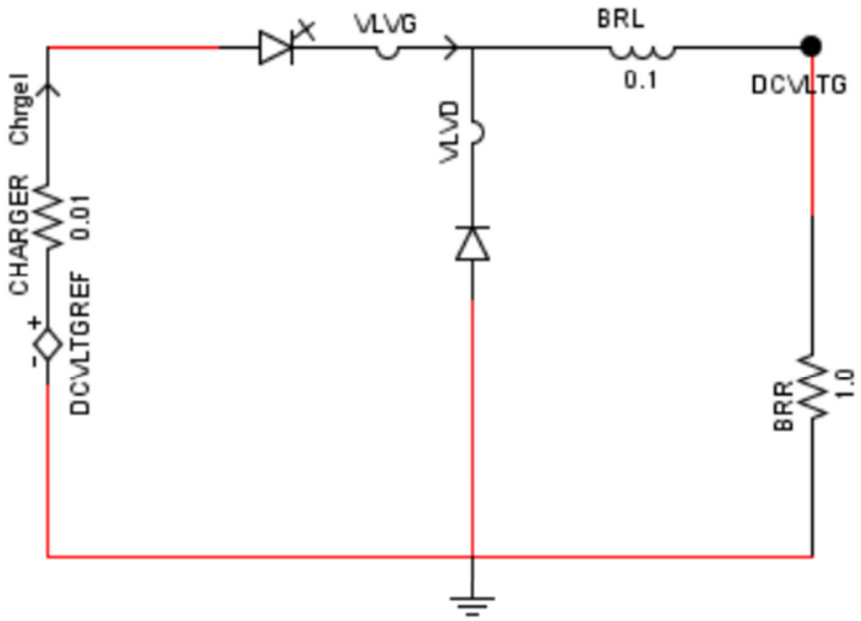

Figure 6.1-1 Buck converter simulation in real-time simulation

The objective of the optimization was set to minimize the difference between the duty ratio command and the final voltage output. An evaluation function employed to produce the figure of merit is presented in Equation (6.1):

$$ISE = \int (V_{REFERENCE} - V_{MEASURED})^2 \, dt \tag{6.1}$$

The final figure of merit (ISE in (6.1)) was utilized by the optimization algorithm in order to generate a new set of candidates for the next step. Based on the objective function evaluation results, the optimization process attempted to moderate the two regulator parameters, i.e., the proportional gain and the integrator time constant, to achieve the given optimization objective.

External hardware was employed to execute the function of the PI regulator. This regulator was implemented using a digital signal controller (DSC) [136]. The block diagram of the PI regulator implemented in the hardware is presented in Figure 6.1-2. Two analog to digital channels on the hardware were used to import the analog signals from the real-time simulation environment. One was the duty ratio command signal while the other was the final voltage output from the simulated Buck converter. Both signals were properly scaled at the analog output stage in the real-time simulation environment, in order to meet the signal input requirement of the hardware controller, 0-3.3V range. The output of the regulator was the duty ratio which was compared with the saw-tooth waveform in a PWM module in the digital signal controller. The saw tooth waveform generator and the comparator were all internal to the digital signal controller, and were configurable through software. The final output from the external hardware was the gating signal which drove the switching element in the Buck converter simulation through the digital interface of the real-time simulator.

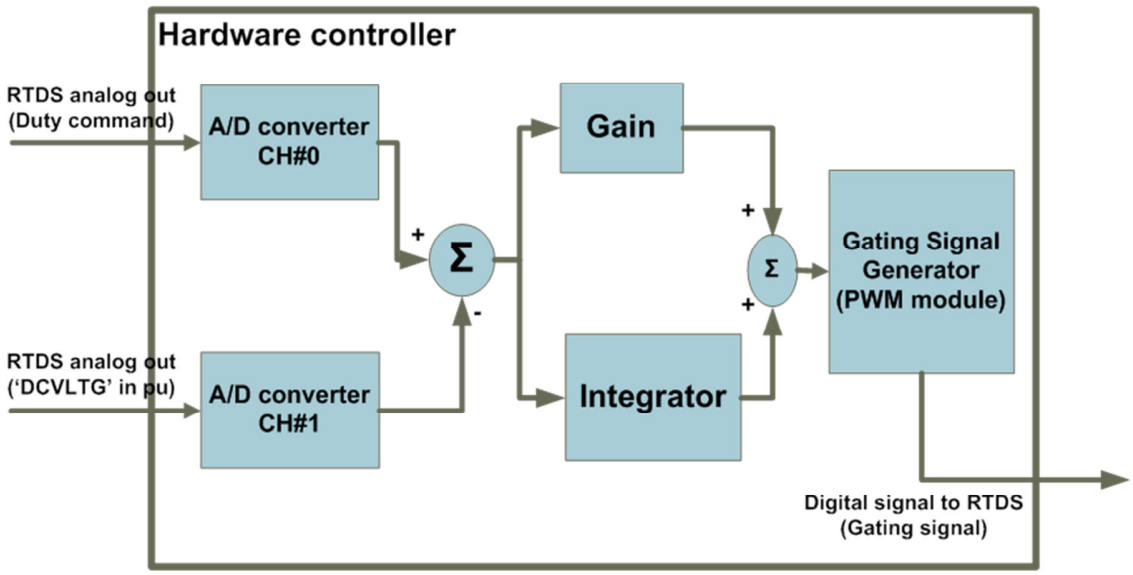

Figure 6.1-2 Block diagram of PI controller implementation

Figure 6.1-3 shows the hardware controller and the necessary signal interface between the hardware controller and the real-time simulation environment. The analog output from the real-time simulation environment is marked as 'GTAO' and the digital input is marked as 'GTDI'. In this experiment, RTDS was employed as the real-time simulator, in which the

Buck converter was simulated. The GTAO and GTDI were the physical signal interface cards, connected to the RTDS through fiber optic cables. The digitally computed simulation waveforms were transmitted to the GTAO card and transformed to analog voltages. The GTDI card processed the gating signal from the controller. The processed gating signal from the GTDI card was transmitted to the real-time simulator through a fiber optic cable using a proprietary protocol.

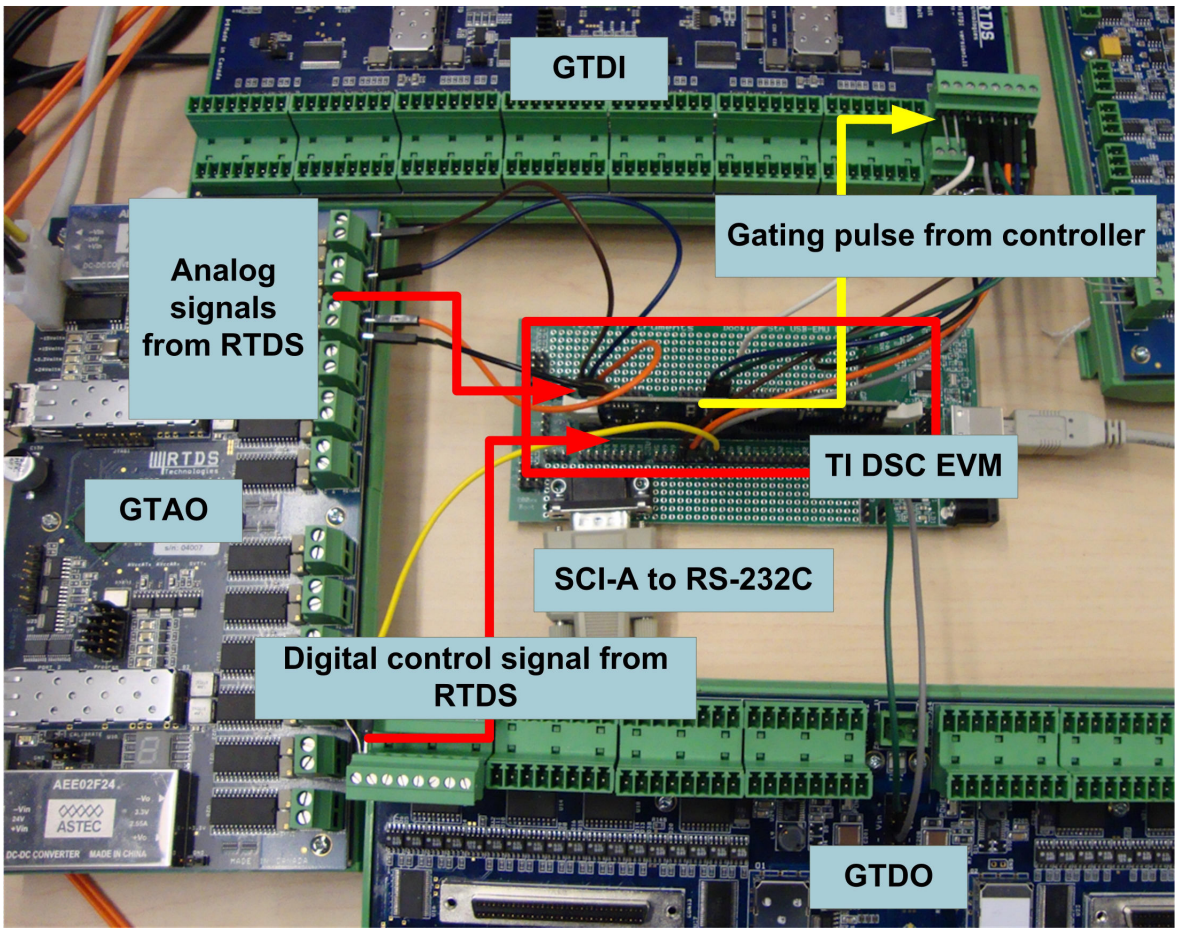

Figure 6.1-3 Experiment set with hardware controller

The selected optimization algorithm was implemented in the MATLAB® software package. The code was derived from [137]. This algorithm received the objective function evaluation results from the real-time simulation environment through TCP/IP socket communication. Based on the objective function evaluation results, a new set of candidates,

i.e., proportional gain and integrator time constant, was generated and passed to the external hardware controller through the RS-232C serial communication channel. The performance of the candidate was evaluated in the real-time simulation and the evaluation result returned to the optimization algorithm. The iteration continued on until termination criteria in the optimization algorithm was met. The configuration of the entire experiment set is presented in Figure 6.1-4.

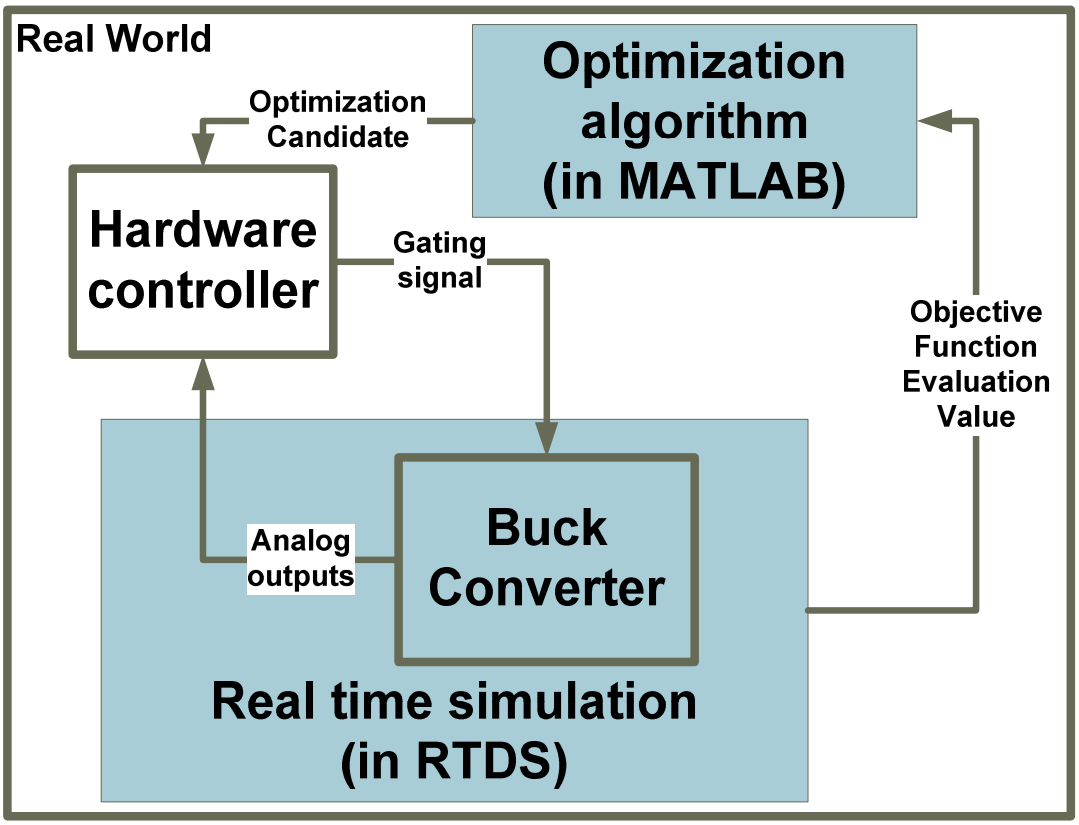

Figure 6.1-4 Experiment setup configuration

A test was designed and executed to evaluate the proposed optimization method using the experiment set which was explained previously. The selected optimization algorithm began with an initial simplex. The members of the initial simplex were as follows: (0.1, 0.1), (100.0, 0.1) and (0.1, 100.0). The optimization process terminated after 31 iterations. According to the termination condition associated with this experiment, the execution would terminate when the difference between the best objective function evaluation result and the

worst result in the simplex (with 3 vertices) was less than the given tolerance (0.2). Figure 6.1-5 presents the trend in the OF values during the iterations.

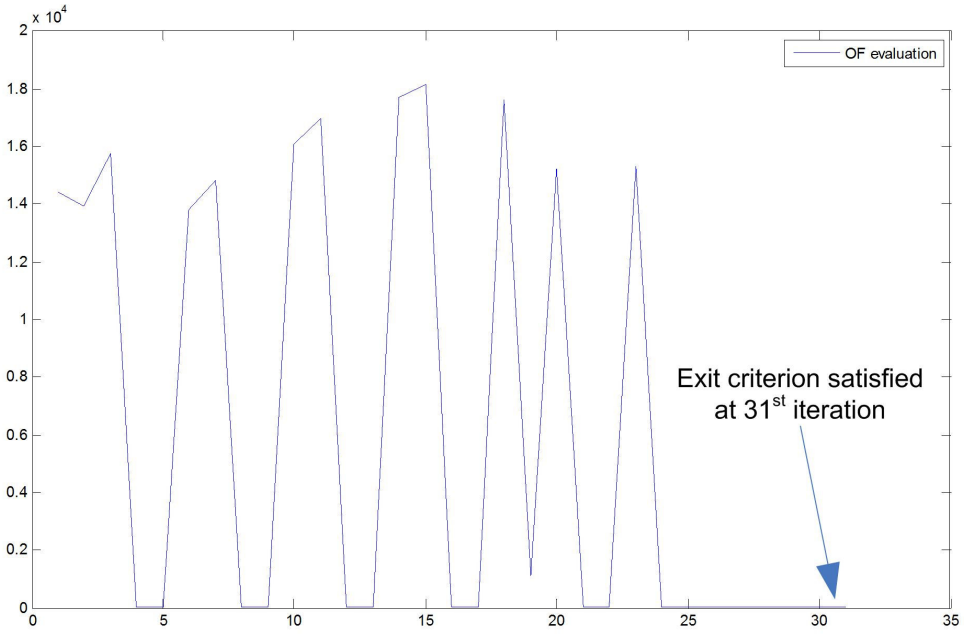

Figure 6.1-5 OF evaluation value trend

Table 6.1-1 presents the candidates in each of those iterations and the corresponding objective function evaluation results.

Iteration Number	Pgain	Iconst	ISE
1	25.075	50.05	14406.3
2	37.5625	25.075	13928.8
3	43.8062	12.5875	15766.7
4	100	0.1	4.31435
5	50.05	0.1	3.47504
6	68.7813	12.5875	13801.2
7	71.9031	6.3438	14823.4
8	50.05	0.1	3.4423
9	75.025	0.1	3.92476
10	59.4156	6.3438	16069.5
11	60.9766	3.2219	16958.6
12	50.05	0.1	3.43313
13	62.5375	0.1	3.75767
14	54.7328	3.2219	17714.7
15	55.5133	1.6609	18146.3
16	50.05	0.1	3.58073
17	56.2938	0.1	3.68648
18	52.3914	1.6609	17620.7
19	52.7816	0.8805	1099.25
20	52.9768	0.4902	15217.7
21	50.05	0.1	3.43081
22	53.1719	0.1	3.66409
23	51.4158	0.4902	15308.1
24	51.5134	0.2951	6.26823
25	51.5622	0.1976	4.533
26	51.6597	0.0024	2.87689
27	48.5378	0.0024	2.96279
28	50.0744	0.0512	3.22339
29	50.0866	0.0268	3.21241
30	50.0927	0.0146	3.12868
31	50.0957	0.0085	3.07578

Table 6.1-1 Hardware in the loop optimization test result

It can be observed from the experimental results in Table 6.1-1 that some of the candidates were not successful in controlling the PWM (i.e., Pulse Width Modulation) duty ratio of the switching element in the Buck converter, resulting in a large ISE (Integrated

Squared Error) value. When the simplex algorithm successfully converged and terminated after 31 iterations, the algorithm was able to reduce the ISE from 14406.3 to 2.8769.

Note that with a physical implementation with analog interface, there may be some noise in measurement which could skew the results. Hence, the same experiment was repeated. This time, the experiment terminated after 36 iterations. The best candidate produced during the iteration was (53.2206, 0.0024) which resulted in the ISE value of 2.7513. The best candidates in both the experiments matched one another with less than 5% difference (in both the parameter values and the ISE values), thus demonstrating that the results were consistent even when the optimization was conducted at a different time.

The performance difference between the best candidate and the worst candidate can be observed from the two waveforms in Figure 6.1-6 and Figure 6.1-7. Figure 6.1-6 is the per-unitized DC voltage output and duty ratio command waveforms plot when the best (fittest) candidate, (51.6587, 0.0024), was applied to the hardware controller as parameters. Figure 6.1-7 is the same waveforms plot as the Figure 6.1-6 when the worst candidate, (54.7328, 3.2219), was applied. When the best (fittest) parameters were applied to the controller, it was able to follow the step change in the duty ratio command immediately with little overshoot. On the contrary, the worst candidate made the controller unable to follow the duty ratio order, as observed in Figure 6.1-7.

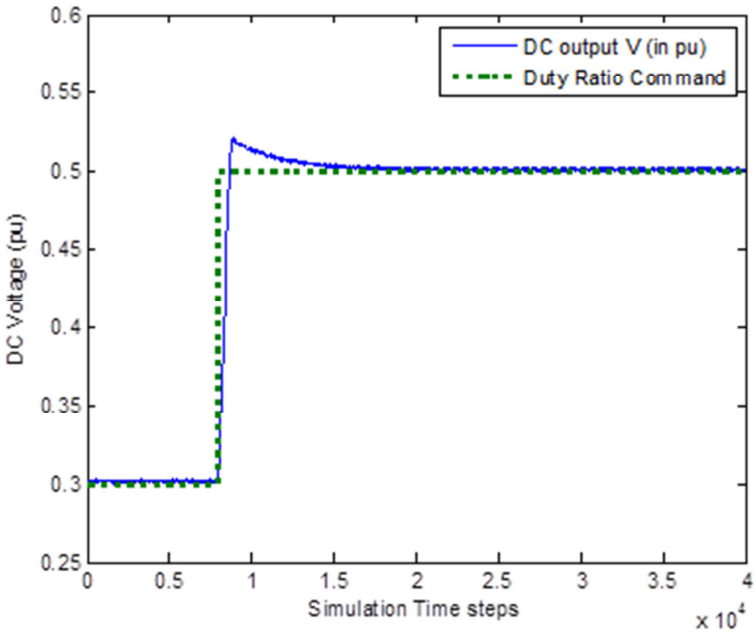

Figure 6.1-6 Controller performance with the best candidate

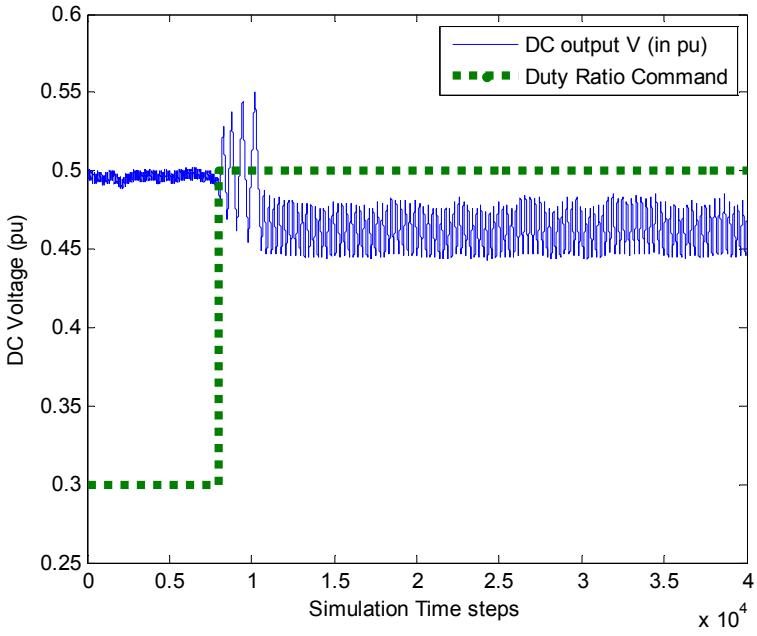

Figure 6.1-7 Controller performance with the worst candidate

6.2 A Real Hardware Controller Optimization with Particle Swarm Optimization Algorithm

The Particle Swarm Optimization (PSO) algorithm was applied to select optimum controller parameters for the same real-time simulation case as was used in chapter 5.2.1.3, i.e., the CIGRE HVDC Benchmark simulation case.

The rectifier and the inverter in the simulation case both have their own controls. Current control is the normal control mode at the rectifier side. The extinction angle control is the normal control mode at the inverter side. The controllers at each side, i.e., the current control and the extinction control, employ simple PI regulators as main regulators.

In this experiment, unlike the previous approach where the whole control was simulated, the rectifier side controller was completely replaced by an external hardware controller. The real hardware controller was implemented on a Texas Instruments TMS320C6713 DSP (Digital Signal Processor) [138]. Figure 6.2-1 presents the external hardware controller which embedded the entire rectifier side controller in the CIGRE HVDC benchmark case. The necessary signal interface between the hardware controller and the real-time simulation case was implemented using analog to digital converters and digital to analog converters.

The optimization algorithm adjusted the following 6 variables:

(i) Proportional gain of the rectifier current PI regulator
(ii) Integrator time constant of the rectifier current PI regulator
(iii) Proportional gain of the inverter current PI regulator
(iv) Integrator time constant of the inverter current PI regulator
(v) Proportional gain of the inverter extinction angle PI regulator
(vi) Integrator time constant of the inverter extinction angle PI regulator

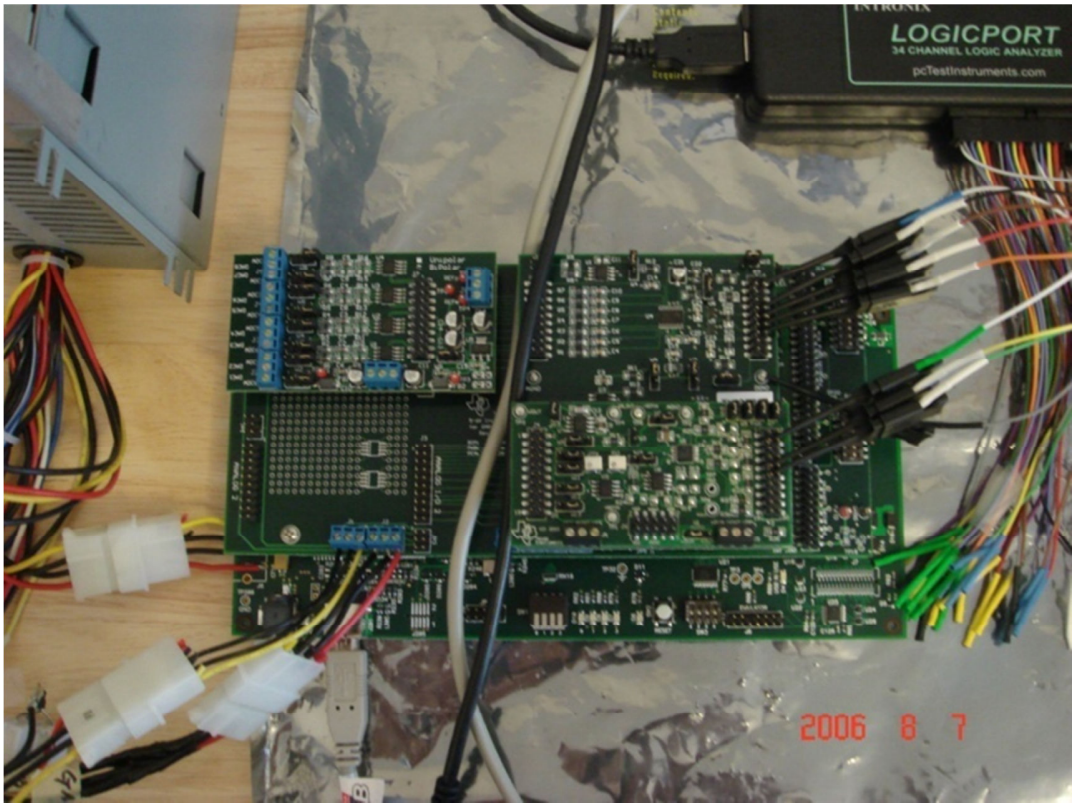

Figure 6.2-1 A real hardware controller implementation

Among these variables, the proportional gain and integral time constant of the rectifier current PI regulator resided in the real physical controller, whereas the rest of the variables were within the simulation case. The parameter in the external hardware controller was modified by means of an RS-232C communication link to the computer running the optimization program. The real hardware controller in this experiment was equipped with the necessary communication interface [139]. This configuration, i.e., the co-existence between the simulation and real world hardware in the optimization process, presents the power of the developed approach in this research. Physically existing real-world electronics can be mixed with simulated controllers seamlessly, and the entire system can be designed and optimized as a whole system.

The objective function for the optimization in this case is as follows:

OF = ISE (Integral Square Error)

$$= K_1 \int_{T_{start}}^{T_{end}} \left(I_{measured} - I_{order} \right)^2 dt$$

where

K_1 : *Weighting factor*

$I_{measured}$: *Measured DC current*

I_{order} : *Ordered DC current*

(6.2)

The optimization problem in this test case can be stated as follows:

Minimize OF(Kp1, Ti1, Kp2, Ti2, Kp3, Ti3)
Subject to
Kp1 > 0, Ti1 > 0
Kp2 > 0, Ti2 > 0
Kp3 > 0, Ti3 > 0

The total length of the experiment was over 170 hours (i.e. 7 days 2 hours) and the number of iterations was 32136. The experimental findings suggest that the developed real-time OE-EMT environment is robust and rapid enough to endure such a lengthy test schedule and the large number of objective function evaluations which may be highly impractical or prohibitive in off-line OE-EMT applications. Another finding from the experiment, this lengthy convergence time, implies that the Particle Swarm Optimization (PSO) is not an effective method for optimizing this particular problem. Thus, this experiment demonstrated that the developed platform can cater to optimization algorithms which would require a large number of objective function evaluations. The graph in Figure 6.2-2 illustrates the trend of ISE (Integrated Squared Error) values (after outliers were removed), which provides the index of the controller's performance, as recorded throughout the experiment. The vertical axis is the magnitude of the ISE value; the horizontal axis is the number of iterations.

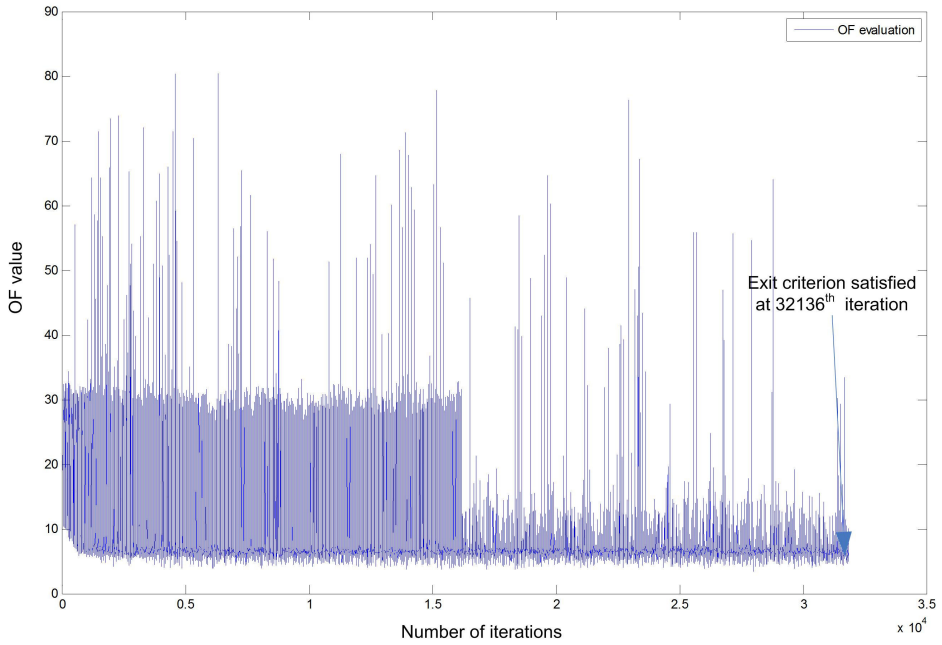

Figure 6.2-2 ISE values recorded along the iterations

The hardware controller which was used in this experiment was a DSP evaluation circuit called DSK (i.e., DSP Starter Kit) originally designed for educational and demonstrational purposes [140]. Thus, the quality of the signal interface, the analog signal in particular, of the hardware circuit is less than ideal, exposing the analog signals to external noise such as power and ground noise.

The trend in Figure 6.2-2 suggests that the proposed optimization algorithm (Particle Swarm Optimization (PSO) algorithm) performed as intended to reduce the ISE value of the optimization target by the iterative process. The beginning part of the trend curve shows that the ISE value settled with a minimum value around 8 after about two thousand iterations. Also, this graph suggests that the adequate number of iterations required by the proposed algorithm is less than four thousand. This trend, the early convergence of minimum OF evaluation values, is more evident in Figure 6.2-3. The iterations would terminate much sooner, if the termination condition had been less stringent. This redesign of the termination criterion is part of the optimization algorithm enhancement, and is retained as an ongoing research theme.

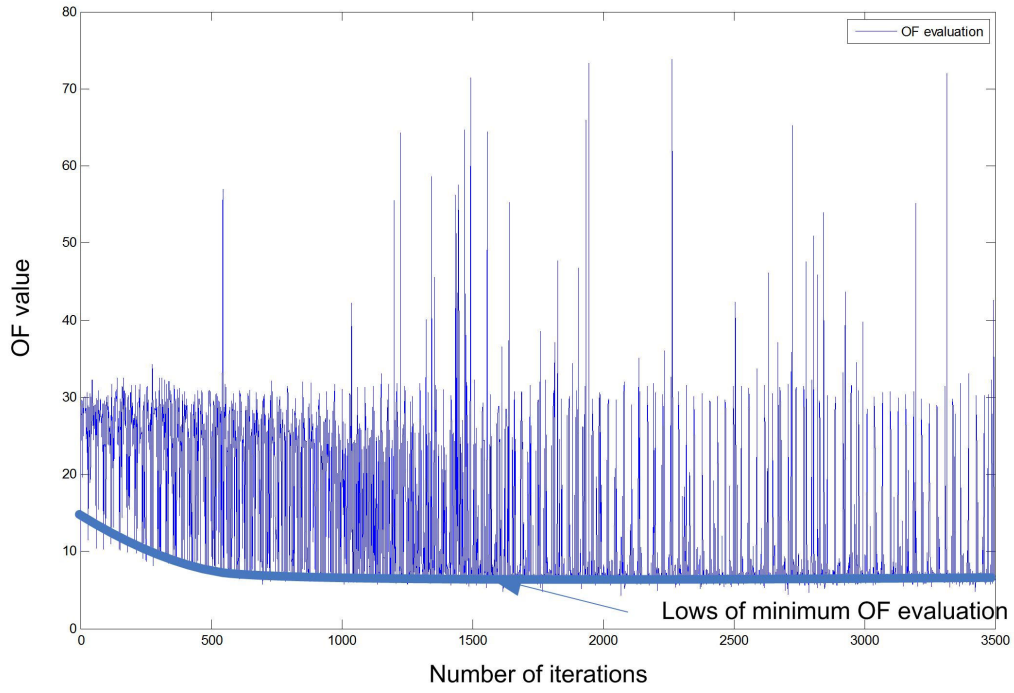

Figure 6.2-3 ISE values recorded along the iterations of the first 3500 iterations

7 CONCLUSION

Previous research conducted in optimization with off-line simulation has paved the way for the research presented in this thesis. A careful review of what has been incorporated into the off-line application of the OE-EMT method reveals the technique's capability, as well as its limitations. The limitations associated with the off-line application of the OE-EMT method provided the most important impetus for the current research. The limitations of the previous research can be summarized as follows:

Challenges in transferring optimized parameters from simulation model to actual hardware

The final result of the optimization process obtained in off-line simulation software cannot always be applied to a real-world hardware device directly. Additional steps must be incorporated in order to apply the optimization result to real-world hardware.

Modeling inadequacies

This limitation of the previous research derives from the modeling requirement of a real system. The off-line simulator is unable to be interfaced with external system directly. Hence, a system in the real world must be modeled or conceptualized in order to be simulated in an off-line environment.

Extreme Simulation time

Off-line simulation speed becomes slower as the size of a simulation case becomes large or a simulation case becomes complex. Often, the required number of OF evaluations can become very large. Thus, the total execution time for the optimization process becomes prohibitively large.

The above limitations of off-line OE-EMT method were largely overcome in the current research. The major contributions are summarized hereafter.

7.1 Contributions

The obvious contribution of my research is its successful migration of the OE-EMT method from an off-line simulation environment to a real-time simulation environment.

During the development and implementation of my research, two specific contributions have been made as follows:

An investigation into real-time OE-EMT implementation methods and identification of the suitable method

Different methods of implementation were investigated. The method which could facilitate the proposed approach better was selected. Two different methods of implementation were reported in section 5.1.1.

The first method involved using a single simulation run for all the necessary OF evaluations. Constructing the NLO algorithm implementation as a building block in the simulation case made this method possible. In this method, only a single real-time simulation run was conducted. This run consisted of several contiguous time slots, each reserved for testing one optimization candidate. The candidate, i.e., optimization parameter set, was selected. The corresponding OF was evaluated over a fixed time slot. It was assumed that all the transients died out in this time slot, so that the next set of parameters could be tested in the next time slot. This approach had the advantage that the simulation run set up time was only required once. However, it was found to have several disadvantages in that there was no guarantee that all transients would die down before the next set of parameters was input. Also, there was no guarantee that the external device would start from the same initial conditions. The complexity of implementation associated with this first method was another disadvantage.

In the second method, the NLO algorithm was implemented as a separate application. Each OF evaluation was a separate real-time simulation run. Managing the OF evaluation was conducted through communication between the NLO implementation and the OF evaluator, i.e., a real-time simulator. The necessary number of OF evaluations was made with the corresponding number of simulation runs. Although this method required a setup interval at the beginning of each simulation run, there was no problem with residual transient from earlier runs. Also it is easier to ensure that the runs begin with the same initial condition. The complete decoupling between the NLO implementation and the real-time simulator is the most obvious advantage of this method. This advantage makes the incorporation of various NLO algorithms much easier.

Hence, this second method was found to be more suitable for real-time implementation of the OE-EMT method. The selected method had been utilized as a general implementation guide in the subsequent implementation phase of the proposed approach.

Comparisons among various optimization algorithms

Three different optimization algorithms were implemented and evaluated. The first algorithm was the down-hill Simplex algorithm; the second, a Genetic Algorithm (GA) and the third, the Particle Swarm Optimization (PSO) Algorithm. These three algorithms were implemented following the implementation method which was identified by the implementation method search.

Then, these algorithms were evaluated using various applications. The applications included a simple mathematical function minimization (i.e., in chapters 5.1.1, 5.1.2.1 and 5.1.2.2); power system controller optimization (i.e., in chapters 5.2.1.1 and 5.2.1.3); hardware controller optimization (i.e., in chapters 6.1 and 6.2).

In practice, the down-hill Simplex algorithm was found to be the most effective among these three algorithms for the problems considered.

This research claims the following three contributions as measures necessary for overcoming shortcomings associated with the off-line approach:

Direct optimization of physical hardware

Real-world hardware can be directly interfaced with the real-time simulation environment, in the same way as in the intended application. The real-time input and output feedback constructs a true closed loop between the real-world hardware and the simulator. In contrast, the off-line approach requires a model of the real-world hardware. Hence, there is the possibility that some hardware details might be overlooked.

An approach was developed where the parameters of the external hardware could be selected directly by the NLO algorithm. This approach ensures that most of the real-world hardware can be preserved during the optimization process.

Therefore, the need for modeling as in the off-line approach and the associated compromise is eliminated by this proposed approach. This advantage was verified in a number of examples (i.e., in chapters 6.1 and 6.2).

Direct transfer of parameters to field application

A communication method for the transmission of an optimization candidate from an NLO algorithm implementation to an external hardware controller was developed. The communication method was based on a simple RS-232C protocol. Then, this communication method was utilized in the real-world hardware optimization experiments (i.e., in chapters 6.1 and 6.2). This communication method facilitated the overall optimization process by eliminating any user intervention during the process, making the optimized controller immediately ready for field deployment.

Reduction of simulation time through parallel simulation

Two benefits can be derived from the real-time nature of the simulation. The first benefit is the immediate reduction in the simulation time. A simulation case runs in real-time in the real-time simulation environment, whereas the off-line counterpart consumes more time for each simulation run. This advantage, reduction in the necessary simulation time was verified (in chapter 0).

The second benefit is the utilization of parallelism. This utilization allows many trial parameter sets to be evaluated concurrently. In addition to utilizing the inherent reduction in simulation time, the research proposed a new approach, i.e., a method of utilizing parallelism in the real-time simulation hardware.

This new method, i.e., parallel OF evaluations in optimization process, can expedite the execution of some of heuristic-based optimization algorithms. Two heuristics-based optimization algorithms, GA and PSO, were selected and evaluated using this new approach (in chapters 5.1.2.1, 5.1.2.2 and 5.2.1.3).

7.2 Suggestions for future research

As a result of the foregoing research, further investigation in the following areas is recommended.

Termination criterion

A method for preventing an excessive number of iterations during the optimization process must be addressed. In particular, when the heuristics-based optimization algorithms were utilized, excessive number of optimization candidate evaluations occurred, revealing the need for better iteration termination criteria. Better criteria would produce the final optimization result reasonably well with substantially fewer evaluations.

New optimization algorithms

Many new optimization algorithms are still being developed and introduced. For example, one such newly introduced optimization algorithm is the artificial immune system (AIS) approach [141]. Many of these newly developed optimization algorithms are heuristics-based. Thus, the approach proposed in this research could accommodate the new optimization algorithms in conjunction with a real-time simulation environment.

Further verification of the outcome from the proposed approach

Although this research brought the optimization process one step closer to the real world application, the effectiveness of the optimization result obtained from the proposed method remains uncertain. The external hardware device can be optimized in conjunction with real-time simulation. However, this procedure does not fully guarantee that the optimized device would perform in exactly the same way as in the simulated environment. Information must be compiled to see how often field applications provide sub optimal performance. This could be used to design more robust optimization approaches and to improve digital models of the power network.

Reference

[1] B. F. Wollenberg, "The price of change [electricity supply industry deregulation]," *Potentials, IEEE,* vol. 16, pp. 14-16, 1997.

[2] H. Jin and W. Xu, "Extended transmission line loadability curve by including voltage stability constrains," presented at the Electric Power Conference, 2008. EPEC 2008. IEEE Canada, 2008.

[3] R. D. Christie, B. F. Wollenberg, and I. Wangensteen, "Transmission management in the deregulated environment," *Proceedings of the IEEE,* vol. 88, pp. 170-195, 2000.

[4] W. Long, D. Cotcher, D. Ruiu, P. Adam, S. Lee, and R. Adapa, "EMTP-a powerful tool for analyzing power system transients," *Computer Applications in Power, IEEE,* vol. 3, pp. 36-41, 1990.

[5] P. G. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt, "A digital TNA for testing relays," in *WESCANEX '91 'IEEE Western Canada Conference on Computer, Power and Communications Systems in a Rural Environment'*, 1991, pp. 47-50.

[6] H. W. Dommel, "Digital Computer Solution of Electromagnetic Transients in Single- and Multiphase Networks," *IEEE Trans on Power Apparatus and Systems,* vol. PAS-88, pp. 388-399, 1969.

[7] H. W. Dommel, "Nonlinear and Time-Varying Elements in Digital Simulation of Electromagnetic Transients," *Power Apparatus and Systems, IEEE Transactions on,* vol. PAS-90, pp. 2561-2567, 1971.

[8] H. W. Dommel, "Transformer Models in the Simulation of Electromagnetic Transients," presented at the 5th Power Systems Computing Conference, Cambridge, England, 1975.

[9] W. S. Meyer and H. W. Dommel, "Numerical Modelling of Frequency-Dependent Transmission-Line Parameters in an Electromagnetic Transients Program," *Power Apparatus and Systems, IEEE Transactions on,* vol. PAS-93, pp. 1401-1409, 1974.

[10] P. G. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt, "A real time digital simulator for testing relays," *Power Delivery, IEEE Transactions on,* vol. 7, pp. 207-213, 1992.

[11] J.-H. Jeon, S.-K. Kim, C.-H. Cho, J.-B. Ahn, and E.-S. Kim, "Deve lopment of Simulator System for Microgrids with Renewable Energy Sources," *Journal of electrical engineering & technology,* vol. 1, pp. 409-413, 2006.

[12] *PSCAD User's manual*. Winnipeg: Manitoba HVDC Research Center Inc., 2003.

[13] D. A. Woodford, A. M. Gole, and R. W. Menzies, "Digital Simulation of DC Links and AC Machines," *Power Apparatus and Systems, IEEE Transactions on,* vol. PAS-102, pp. 1616-1623, 1983.

[14] CIGRE WG 14-02, "First benchmark model for HVDC control studies," *Electra,* vol. 135, pp. 55-75, 1991.

[15] E. W. Kimbark, *Direct Current Transmission*: Wiley-Interscience, 1971.

[16] A. M. Gole and V. K. Sood, "A static compensator model for use with electromagnetic transients simulation programs," *Power Delivery, IEEE Transactions on,* vol. 5, pp. 1398-1407, 1990.

[17] R. A. Otto, T. H. Putman, and L. Gyugyi, "Principles and Applications of Static, Thyristor-Controlled Shunt Compensators," *Power Apparatus and Systems, IEEE Transactions on,* vol. PAS-97, pp. 1935-1945, 1978.
[18] L. Gyugyi, "Dynamic compensation of AC transmission lines by solid-state synchronous voltage sources," *Power Delivery, IEEE Transactions on,* vol. 9, pp. 904-911, 1994.
[19] Dingyu Xue, YangQuan Chen, and Derek P. Atherton. (2007). *Linear Feedback Control*.
[20] J. M. Hammersley and D. C. Handscomb, *Monte Carlo methods*. Norwick: Fletcher & Sons, Ltd., 1975.
[21] A. GoIe, S. Filizadeh, R. Menzies, and P. Wilson, "Optimization-enabled electromagnetic transient simulation," in *Power Engineering Society General Meeting, 2004. IEEE*, 2004, p. 1133 Vol.1.
[22] R. Kuffel, J. Giesbrecht, T. Maguire, R. P. Wierckx, and P. G. McLaren, "A fully digital power system simulator operating in real time," presented at the Electrical and Computer Engineering, 1996. Canadian Conference on, 1996.
[23] C. Dufour and J. Belanger, "A PC-Based Real-Time Parallel Simulator of Electric Systems and Drives," presented at the Parallel Computing in Electrical Engineering, 2004. PARELEC 2004. International Conference on, 2004.
[24] H. Kopetz, *Real-Time Systems*, Second Edition ed.: Springer, 2011.
[25] G. Wenzhong, W. Ge, and N. Jiaxin, "Development of low voltage ride-through control strategy for wind power generation using real time digital simulator," presented at the Power Systems Conference and Exposition, 2009. PES '09. IEEE/PES, 2009.
[26] P. Forsyth, R. Kuffel, R. Wierckx, C. Jin-Boo, Y. Yong-Beum, and K. Tae-Kyun, "Comparison of transient stability analysis and large-scale real time digital simulation," presented at the Power Tech Proceedings, 2001 IEEE Porto, 2001.
[27] P. Forsyth and R. Kuffel, "Utility applications of a RTDS Simulator," presented at the Power Engineering Conference, 2007. IPEC 2007. International, 2007.
[28] L. M. Wedepohl, "Application of Matrix Methods to the Solution of Travelling-Wave Phenomena in Polyphase Systems," *Proc. IEE,* vol. 110, 1963.
[29] S. L. Woodruff, "Complexity in power systems and consequences for real-time computing," in *Power Systems Conference and Exposition, 2004. IEEE PES*, 2004, pp. 1770-1775 vol.3.
[30] T. Bi, Y. Zhang, X. Xiao, P. Forsyth, and R. Wierckx, "Large scale power system simulation and PMU testing using a real time digital simulator," in *Power Engineering Conference, 2007. IPEC 2007. International*, 2007, pp. 383-388.
[31] M. Steurer, "PEBB based high-power hardware-in-loop simulation facility for electric power systems," presented at the Power Engineering Society General Meeting, 2006. IEEE, 2006.
[32] J. B. Lee, C. H. Jung, I. D. Kim, and Y. K. Baek, "Protective relay testing and characteristic analysis for high impedance faults in transmission lines," in *Power Engineering Society Summer Meeting, 1999. IEEE*, 1999, pp. 1076-1081 vol.2.
[33] R. C. Gonzalez, A. F. Valdivieso, and O. E. Bolado, "Directional comparison protection for lines, buses, and transformers," in *Protective Relay Engineers, 2009 62nd Annual Conference for*, 2009, pp. 513-526.

[34] G. D. Hachtel, R. K. Brayton, and F. G. Gustavson, "The sparse tableau approach to network analysis and design," *IEEE Trans., Circuit Theory,* vol. CT-18, pp. 101-103, Jan. 1971 1971.
[35] Serge Lang, *Linear Algebra*. New York: Springer, 1987.
[36] M. Heidari, "Decision Support Algorithms for Power System and Power Electronic Design," Doctor of Philosophy, Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, 2010.
[37] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal,* vol. 7, pp. 308-313, 1965.
[38] R. Hooke and T. A. Jeeves, "`` Direct Search" Solution of Numerical and Statistical Problems," *J. ACM,* vol. 8, pp. 212-229, 1961.
[39] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal,* vol. 7, pp. 155-162, 1964.
[40] G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell, *Engineering Optimization*: Wiley-Interscience, 1983.
[41] A. S. Fraser, "Simulation of genetic systems," *Journal of Theoretical Biology,* vol. 2, pp. 329-346, 1962.
[42] R. C. Eberhart and Y. Shi, "Comparison between Genetic Algorithms and Particle Swarm Optimization " presented at the 7th IEEE International Conference on Evolving Computation, San Diego(CA, USA), 1998.
[43] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science,* vol. 220, pp. 671-680, May 13, 1983 1983.
[44] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* vol. 26, pp. 29-41, 1996.
[45] V. Jalili-Marandi and V. Dinavahi, "Large-scale transient stability simulation on graphics processing units," in *Power & Energy Society General Meeting, 2009. PES '09. IEEE*, 2009, pp. 1-6.
[46] S. Filizadeh, "Optimization-Enabled Electromagnetic Transient Simulation," Ph.D, Electrical and Computer Engineering, University of Manitoba, Winnipeg, 2004.
[47] A. M. Gole, S. Filizadeh, R. W. Menzies, and P. L. Wilson, "Optimization-Enabled Electromagnetic Transient Simulation," *IEEE TRANSACTIONS ON POWER DELIVERY,* vol. 20, pp. 512-518, JANUARY 2005 2005.
[48] U. N. Gnanarathna, A. M. Gole, and R. P. Jayasinghe, "Efficient Modeling of Modular Multilevel HVDC Converters (MMC) on Electromagnetic Transient Simulation Programs," *Power Delivery, IEEE Transactions on,* vol. PP, pp. 1-1.
[49] A. B. Dehkordi, "Improved Models of Electric Machines for Real-Time Digital Simulation," Ph.D, Electrical and Computer Engineering, University of Manitoba, Winnipeg, 2009.
[50] X. Lin, "System Equivalent for Real Time Digital Simulator," Doctor of Philosophy, Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, 2010.
[51] P. Kundur, J. Paserba, V. Ajjarapu, G. Andersson, A. Bose, C. Canizares, N. Hatziargyriou, D. Hill, A. Stankovic, C. Taylor, T. Van Cutsem, and V. Vittal, "Definition and classification of power system stability IEEE/CIGRE joint task force

on stability terms and definitions," *Power Systems, IEEE Transactions on,* vol. 19, pp. 1387-1401, 2004.

[52] A. M. Gole, "Simulation tools for system transients: an introduction," in *Power Engineering Society Summer Meeting, 2000. IEEE*, 2000, pp. 761-762 vol. 2.

[53] M. Roitman, E. H. Watanabe, and F. J. Lyra, "Power systems analog simulation enhancement using a new programmable electronic model," *Power Systems, IEEE Transactions on,* vol. 4, pp. 286-292, 1989.

[54] H. Doi, M. Goto, T. Kawai, S. Yokokawa, and T. Suzuki, "Advanced power system analogue simulator," *Power Systems, IEEE Transactions on,* vol. 5, pp. 962-968, 1990.

[55] A. Moshref and S. Khan, "Harmonic analysis for industrial power systems computation techniques and filtering," in *Pulp and Paper Industry Technical Conference, 1992., Conference Record of 1992 Annual*, 1992, pp. 115-121.

[56] M. Hirakami and W. Neugebauer, "Transient Network Analyzer Operation with Digital Computer Control and Analysis," *Power Apparatus and Systems, IEEE Transactions on,* vol. PAS-100, pp. 1597-1607, 1981.

[57] N. Watson and J. Arrillaga, *Power systems electromagnetic transients simulation* London: The Institute of Engineering and Technology, 2003.

[58] A. Colombo, J. P. Couvreux, W. Gielessen, P. G. Kendall, and J. Vontobel, "Determination of Transient Recovery Voltages by Means of Transient Network Analyzers," *Power Apparatus and Systems, IEEE Transactions on,* vol. PAS-87, pp. 1371-1380, 1968.

[59] SIEMENS ENERGY. (2012). *PSS/E - Siemens*. Available: http://www.energy.siemens.com/us/en/services/power-transmission-distribution/power-technologies-international/software-solutions/pss-e.htm

[60] Powertech. (2012). *TSAT - Transient Security Assessment Tool*. Available: http://www.powertechlabs.com/areas-of-focus/smart-utility/dsatools-software/transient-security-assessment-tool/

[61] Prabha Kundur, *Power System Stability and Control*: McGraw-Hill Professional, 1994.

[62] N. Kshatriya, "Power System Controller Design by Optimal Eigenstructure Assignment," DOCTOR OF PHILOSOPHY, Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, 2011.

[63] D. P. Bernardon and F. D. Veiga, "Use of the ATP Draw Software for Studies of Parallelism in Distribution Networks Served by Different-Source Substations," presented at the Power Engineering, Energy and Electrical Drives, 2007. POWERENG 2007. International Conference on, 2007.

[64] J. B. Lee, C. W. Ha, and C. H. Jung, "Development of digital distance relaying algorithm in combined transmission lines with underground power cables," presented at the Power Engineering Society Summer Meeting, 2001. IEEE, 2001.

[65] X. Lei, E. Lerch, D. Povh, and O. Ruhle, "A large integrated power system software package-NETOMAC," presented at the Power System Technology, 1998. Proceedings. POWERCON '98. 1998 International Conference on, 1998.

[66] R. Krebs and O. Ruhle, "NETOMAC real-time simulator - a new generation of standard test modules for enhanced relay testing," presented at the Developments in Power System Protection, 2004. Eighth IEE International Conference on, 2004.

[67] F. H. Branin, Jr., "Computer methods of network analysis," *Proceedings of the IEEE,* vol. 55, pp. 1787-1801, 1967.
[68] A. M. Gole, S. Filizadeh, and P. L. Wilson, "Inclusion of Robustness Into Design Using Optimization-Enabled Transient Simulation," *IEEE TRANSACTIONS ON POWER DELIVERY,* vol. 20, pp. 1991-1997, JULY 2005 2005.
[69] S. Filizadeh and A. M. Gole, "OPTIMAL DESIGN OF POWER ELECTRONIC SYSTEMS USING ELECTROMAGNETIC TRANSIENT SIMULATION," presented at the IEEE CCECE/CCGEI, Saskatoon, Canada, 2005.
[70] S. Filizadeh, A. M. Gole, D. A. Woodford, and G. D. Irwin, "An Optimization-Enabled Electromagnetic Transient Simulation-Based Methodology for HVDC Controller Design," *Power Delivery, IEEE Transactions on,* vol. 22, pp. 2559-2566, 2007.
[71] S. Filizadeh, M. Heidari, A. Mehrizi-Sani, J. Jatskevich, and J. A. Martinez, "Techniques for Interfacing Electromagnetic Transient Simulation Programs With General Mathematical Tools IEEE Taskforce on Interfacing Techniques for Simulation Tools," *Power Delivery, IEEE Transactions on,* vol. 23, pp. 2610-2622, 2008.
[72] M. Heidari, S. Filizadeh, and A. M. Gole, "Support Tools for Simulation-Based Optimal Design of Power Networks With Embedded Power Electronics," *Power Delivery, IEEE Transactions on,* vol. 23, pp. 1561-1570, 2008.
[73] A. Mehrizi-Sani and S. Filizadeh, "An Optimized Space Vector Modulation Sequence for Improved Harmonic Performance," *Industrial Electronics, IEEE Transactions on,* vol. 56, pp. 2894-2903, 2009.
[74] J. M. Renders and H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 1994, pp. 312-317 vol.1.
[75] Y. John and L. Bogju, "A simplex genetic algorithm hybrid," in *Evolutionary Computation, 1997., IEEE International Conference on*, 1997, pp. 175-180.
[76] C. Z. Janikow and Z. Michalewicz, "A specialized genetic algorithm for numerical optimization problems," in *Tools for Artificial Intelligence, 1990.,Proceedings of the 2nd International IEEE Conference on*, 1990, pp. 798-804.
[77] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*: Addison-Wesley Longman Publishing Co., Inc., 1989.
[78] A. Bakirtzis, V. Petridis, and S. Kazarlis, "Genetic algorithm solution to the economic dispatch problem," *Generation, Transmission and Distribution, IEE Proceedings-,* vol. 141, pp. 377-382, 1994.
[79] L. Min, W. Jie, Z. Jun, and G. La-Mei, "Power dispatching of distributed wind-Solar power generation hybrid system based on genetic algorithm," in *Power Electronics Systems and Applications, 2009. PESA 2009. 3rd International Conference on*, 2009, pp. 1-4.
[80] D. C. Walters and G. B. Sheble, "Genetic algorithm solution of economic dispatch with valve point loading," *Power Systems, IEEE Transactions on,* vol. 8, pp. 1325-1332, 1993.

[81] S. Smith, "The simplex method and evolutionary algorithms," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 1998, pp. 799-804.
[82] K. Kyu-Ho, R. Sang-Bong, K. Soo-Nam, and Y. Seok-Ku, "Application of ESGA hybrid approach for voltage profile improvement by capacitor placement," *Power Delivery, IEEE Transactions on,* vol. 18, pp. 1516-1522, 2003.
[83] G. Guo and Y. Shouyi, "Evolutionary parallel local search for function optimization," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* vol. 33, pp. 864-876, 2003.
[84] A. Kirillov. (2012). *AForge.NET Framework*. Available: http://www.aforgenet.com/framework/
[85] A. Kirillov. (2007). *AForge.NET*. Available: http://www.codeproject.com/KB/recipes/aforge.aspx
[86] L. Guangming, K. Lishan, L. Yongsheng, and R. Sarker, "Analysis the Impact of Genetic Operators in Evolution Strategies," in *Genetic and Evolutionary Computing, 2008. WGEC '08. Second International Conference on*, 2008, pp. 88-91.
[87] M. Buckland, *AI Techniques for Game Programming* Course Technology PTR, 2002.
[88] J. Dalton. (2012). *GA Roulette wheel selection*. Available: http://www.edc.ncl.ac.uk/highlight/rhjanuary2007g02.php/
[89] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. San Francisco: Morgan Kaufmann Publishers, 2001.
[90] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. London: John Wiley & Sons, 2005.
[91] M. R. AlRashidi and M. E. El-Hawary, "A Survey of Particle Swarm Optimization Applications in Electric Power Systems," *Evolutionary Computation, IEEE Transactions on,* vol. 13, pp. 913-918, 2009.
[92] Z. Wen, H. Lei, and B. Lesieutre, "Optimal Load Modal Parameters Using Particle Swarm Optimization," in *Natural Computation, 2008. ICNC '08. Fourth International Conference on*, 2008, pp. 540-544.
[93] P. Mitra and G. K. Venayagamoorthy, "A DSTATCOM controller tuned by Particle Swarm Optimization for an Electric Ship Power System," in *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, 2008, pp. 1-6.
[94] H. Chao-Ming, H. Chi-Jen, and W. Ming-Li, "A particle swarm optimization to identifying the ARMAX model for short-term load forecasting," *Power Systems, IEEE Transactions on,* vol. 20, pp. 1126-1133, 2005.
[95] Y. Bo, C. Yunping, Z. Zunlian, and H. Qiye, "Solving Optimal Power Flow Problems with Improved Particle Swarm Optimization," in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, 2006, pp. 7457-7461.
[96] G. Zwe-Lee, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *Power Systems, IEEE Transactions on,* vol. 18, pp. 1187-1195, 2003.
[97] "ECMA-334 C# Language Specification 4 th edition (June 2006)," ed. Geneve: ECMA International, 2006.

[98] (2003). *AdaptiveView Resource page*. Available: http://www.adaptiveview.com/resources/index.html
[99] I. Sun Microsystems. (2000). *Java Language Specification, Second Edition*. Available: http://java.sun.com/docs/books/jls/second_edition/html/intro.doc.html#237601
[100] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," presented at the Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, 2002.
[101] 김형수, 문경준, 황기헌, 박준호, 정정원, and 김성학, "A Study on Large Scale Unit Commitment Using Genetic Algorithm," *Journal of KIEE* vol. 1997, pp. 174-176, 1997.
[102] C. W. W. Hasselfield, P.; Penner, L.; Lau, M.; Gole, A.M., "An automated method for least cost distribution planning " *Power Delivery, IEEE Transactions on,* vol. 5, pp. 1188 - 1194, 1990
[103] A.M. Gole and A. Daneshpooy, "Towards Open Systems: A PSCAD/EMTDC to MATLAB Interface," in *IPST 97*, Seattle, 1997.
[104] W. Ren, M. Sloderbeck, M. Steurer, V. Dinavahi, T. Noda, S. Filizadeh, A. R. Chevrefils, M. Matar, R. Iravani, C. Dufour, J. Belanger, M. O. Faruque, K. Strunz, and J. A. Martinez, "Interfacing Issues in Real-Time Digital Simulators," *Power Delivery, IEEE Transactions on,* vol. 26, pp. 1221-1230, 2011.
[105] J. Jin-Hong, K. Jong-Yul, K. Seul-Ki, A. Ong-Bo, and P. JuneHo, "Development of HILS(Hardware In-Loop Simulation) system for MMS(Microgrid Management System) by using RTDS," presented at the Power Electronics and Motion Control Conference, 2008. EPE-PEMC 2008. 13th, 2008.
[106] M. Andrus, M. Steurer, C. Edrington, F. Bogdan, H. Ginn, R. Dougal, E. Santi, and A. Monti, "Real-time simulation-based design of a power-hardware-in-the-loop setup to support studies of shipboard MVDC issues," presented at the Electric Ship Technologies Symposium, 2009. ESTS 2009. IEEE, 2009.
[107] A. Monti, S. D'Arco, and A. Deshmukh, "A new architecture for low cost Power Hardware in the Loop testing of power electronics equipments," in *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, 2008, pp. 2183-2188.
[108] J. N. Paquin, J. Belanger, L. A. Snider, C. Pirolli, and L. Wei, "Monte-carlo study on a large-scale power system model in real-time using eMEGAsim," in *Energy Conversion Congress and Exposition, 2009. ECCE 2009. IEEE*, 2009, pp. 3194-3202.
[109] R. C. Durie and C. Pottle, "A transputer-based real-time digital transient network analyzer," in *Circuits and Systems, 1992., Proceedings of the 35th Midwest Symposium on*, 1992, pp. 1260-1264 vol.2.
[110] Eui-chol Roh, Kyu-Beom Jeong, and Nam-seob Choi, *Power electronics*, Second ed. Seoul: Mun-un Dang, 1997.
[111] *RTDS Manual Set*. Winnipeg: RTDS Technologies, Inc. , 2006.
[112] Manitoba HVDC Research Center. (2011). *PSCAD X4 Online Help*.
[113] Il Do Yoo and A.M. Gole, "Compensating for Interface Equipment Limitations to Improve Simulation Accuracy of Real-Time Power Hardware In Loop Simulation," *IEEE Trans on Power Delivery,* 2012.

[114] M. A. Pai, *Energy Function Analysis for Power System Stability*. Dordrecht: Kluwer Academic Publishers, 1989.
[115] H. Li, M. Steurer, K. L. Shi, S. Woodruff, and D. Zhang, "Development of a Unified Design, Test, and Research Platform for Wind Energy Systems Based on Hardware-in-the-Loop Real-Time Simulation," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS,* vol. 53, pp. 1144-1151, AUGUST 2006 2006.
[116] S.-M. BAEK and J.-W. PARK, "Optimal Tuning of Nonlinear Parameters of a Power System Stabilizer Using a Real-Time Digital Simulator," in *2011 IEEE Industry Applications Annual Meeting*, Orlando, FL USA, 2011.
[117] Telecommunications Industry Association, "TIA TIA-232-F: Interface between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange," ed, 1997.
[118] Texas Instruments Incorporated. (2010, C28x IQmath Library. Available: http://www.ti.com/lit/sw/sprc990/sprc990.pdf
[119] M. Heidari, "Decision Support Algorithms for Power System and Power Electronic Design," Ph.D, Electrical and Computer Engineering, University of Manitoba, Winnipeg, 2010.
[120] Freescale Semiconductor, "MPC7450 RISC Microprocessor Family Reference Manual," ed, 2005.
[121] "IEEE Standard for a Versatile Backplane Bus: VMEbus," *ANSI/IEEE Std 1014-1987,* p. 0_1, 1987.
[122] PCI Industrial Computers Manufacturers Group, "AdvancedTCA PICMG 3.0 Short Form Specification," ed, 2003.
[123] Defense Advanced Research Projects Agency Information Processing Techniques Office, "RFC 791 - INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION," ed, 1982.
[124] V. Cerf and R. Kahn, "A Protocol for Packet Network Intercommunication," *Communications, IEEE Transactions on,* vol. 22, pp. 637-648, 1974.
[125] Brian W. Kernighan and Dennis Ritchie, *The C Programming Language*: Prentice Hall, 1988.
[126] IBM. (2004). *IBM PowerPC 750GX RISC Microprocessor User's Manual*.
[127] ABB. (2012). *Blackwater back-to-back*. Available: http://www.ng.abb.com/industries/ap/db0003db004333/6789fdb700c450e0c125774a004f4d64.aspx
[128] CIGRE WG 14-02. (1991, April) First benchmark model for HVDC control studies. *Electra*. 55-75.
[129] A. M. Gole, "HVDC Transmission Lecture Note," ed, 2004.
[130] Analog Device. (2012). *ADSP-21062: SHARC, 40 MHz, 120 MFLOPS, 5v, Floating Point* Available: http://www.analog.com/en/processors-dsp/sharc/adsp-21062/processors/product.html
[131] IBM. (2005). *PowerPC 750CX and 750CXe Microprocessor*. Available: https://www-01.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_750CX_and_750CXe_Microprocessor
[132] Freescale Semiconductor, "MPC7448 RISC Microprocessor Hardware Specifications," ed, 2007.

[133] M. Szechtman, T. Wess, and C. V. Thio, "A benchmark model for HVDC system studies," in *AC and DC Power Transmission, 1991., International Conference on*, 1991, pp. 374-378.
[134] Manitoba HVDC Research Center, "CIGRE BENCHMARK MODEL for HVDC CONTROLS," ed, 2012.
[135] T. Maguire and J. Giesbrecht, "Small Time-step (< 2μSec) VSC Model for the Real Time Digital Simulator " presented at the International Conference on Power Systems Transients (IPST'05) Montreal, Canada 2005.
[136] "TMS320C2000™ Experimenter Kit Overview," Texas Instruments Incorporated, Ed., ed, 2009.
[137] J. Borggaard. (2009). *Nelder-Mead optimization search*. Available: http://people.sc.fsu.edu/~jburkardt/m_src/nelder_mead/nelder_mead.html
[138] Texas Instruments. (2005). *TMS320C6713 FLOATING-POINT DIGITAL SIGNAL PROCESSOR*. *SPRS186L*.
[139] DSP Global. *UART SERIAL COMMUNICATION PORT EXPANSION DAUGHTER CARDS WITH RS-232-C AND RS-422 INTERFACES FOR TI AND SPECTRUM DIGITAL DSK DEVELOPMENT SYSTEMS*.
[140] SPECTRUM DIGITAL. (2003). *TMS320C6713 DSK Technical Reference*. *506735-0001*.
[141] P. Mitra and G. K. Venayagamoorthy, "An Adaptive Control Strategy for DSTATCOM Applications in an Electric Ship Power System," *Power Electronics, IEEE Transactions on,* vol. 25, pp. 95-104, 2009.