Faculty of Engineering of the University of Porto

FEUP

Application of Autoencoders and Evolutionary Swarms to Optimization Problems

Cristiano Frederico Teixeira Moreira

Dissertation integrated in the
Master Degree in Electrical and Computer Engineering
Major Energy

Supervisor: Prof. Doutor Vladimiro Henrique Barrosa Pinto de Miranda

July of 2009

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2008/2009

A Dissertação intitulada

"Aplicação de Auto-modeladores Entrópicos em Problemas de Optimização"

foi aprovada em provas realizadas em 20 /Julho/2009

o júri

Presidente Professora Doutora Maria Teresa Costa Pereira da Silva Ponce de Leão
Professora Auxiliar da Faculdade de Engenharia da Universidade do Porto

Professora Doutora Ana Maria Pinto de Moura
Professora Auxiliar do Departamento de Economia, Gestão e Engenharia Industrial da Universidade de Aveiro

Professor Doutor Vladimiro Henrique Barrosa Pinto de Miranda
Professor Catedrático da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor – Cristiano Frederico Teixeira Moreira

Faculdade de Engenharia da Universidade do Porto

Abstract

In the last few years there has been an increasing interest and development in the field of applying computational intelligence and namely evolutionary algorithms and Artificial Neural Networks to electrical power systems.

Then, this thesis proposes the application of missing sensor restoration with autoencoders to electrical power systems. An autoencoder is used to learn the interrelationships between the variables of a power system through inspection of historical data. Once trained, the autoencoder is used to restore one or more arbitrary lost signals. Recovery techniques include alternating projection onto convex sets (POCS) and iterative search algorithms like EPSO.

The first search technique, unconstrained search, involves simply minimizing the error between the missing sensors inputs and outputs on the autoencoder, while the second, constrained search, looks at the error between the entire input pattern and output pattern (both missing and known sensors) to achieve a final answer.

Subsequently, one proposes a hybrid method, applying the alternating projection onto convex sets (POCS) followed by constrained search with optimization algorithm EPSO.

Initially, the signal restoration techniques are applied in the example of the parabola ($y(x) = x^2$). This example highlights the difficulties of the task, the merits of the approach and the behavior of each method. Afterwards, the recovery methods are applied to a 24 bus power system, "The IEEE Reliability Test System – 1996", one area.

Key Words: Entropy, Evolutionary Programming, Optimum Solution, Evolutionary Particle Swarm Optimization, Minimum Square Error, Neural-Network, POCS, Signal restoration.

Acknowledgements

To Professor Vladimiro Henrique Miranda, for proposing this Thesis, for the opportunity to work with him, his endless confidence, support and continuous new ideas that enhanced this work. Additionally, he allowed me the privilege of working in such recognized institution as INESC Porto.

I thank my parents and my brothers for all of the efforts they did so that I have accomplished this stage and, in particular, the development of this work.

To my colleagues and friends, especially Nuno Fonseca and Pedro Soares, for accompanying me in this journey and making me having a good time whenever was possible.

I would also like to thank to all people that supported me along the work and also of my academic course.

To all my sincere gratitude,

Cristiano Moreira

Index

List of figures

List of tables

Abbreviations e Symbols

List of abbreviations (ordered alphabetically)

ANN	Artificial Neural Network
EPSO	Evolutionary Particle Swarm Optimization
FEUP	*Faculdade de Engenharia da Universidade do Porto*
GA	Genetic Algorithms
IEEE	Institute of Electrical and Electronics Engineers
kV	kilo-Volt (voltage unit – 1 kV = 10^3 V)
MAE	Mean Absolute Error
MISED	Missing Sensor Data
MSE	Mean Squared Error
MSR	Missing Sensor Restoration
MVA	Mega-Volt-Ampère (power unit – 1 MVA = 10^6 VA)
MW	Mega-Watt (active power unit – 1 MW = 10^6 W)
NN	Neural Network
OPF	Optimal Power Flow
PCA	Principal Component Analysis
POCS	Alternating Projections Onto Convex Sets
PSO	Particle Swarm Optimization
RBM	Restricted Boltzmann Machine
RTS	Reliability Test System
SCADA	System Control and Data Acquisition

List of symbols

p.u.	Per Unit

Chapter 1

Introduction

This master thesis was developed in INESC Porto, integrated in the Master Degree in Electrical and Computer Engineering at the Faculty of Engineering of the University of Porto (FEUP).

A new approach to restoring missing data in the context of an Energy Management System or a Distribution Management System is presented in this work, using autoencoders and data restoration algorithms. This approach is based on the properties of autoencoders, which are neural networks with a special architecture so that they have as many inputs as outputs. When properly trained, an autoencoder stores in the weights of the neural network the information representing the real system whose data was used to train it. Any input pattern coherent with the real system will produce a similar output with negligible error. However, an input with some data inconsistent with the real system will generate a significant error between the input and the output vector. This property has been used to reconstruct missing sensor signals and will be used in this thesis to reconstruct voltage and active and reactive power values, as well as switching device status.

1.1 Problem to be addressed and its context

When the first power plants showed up in late 19^{th} century, they were built close to industrial and populational centers and were generally connected by radial networks to the main consumption centers. Therefore, it was extremely simple to explore this kind of system by controlling turbines and alternator excitation to keep the system balanced in terms of frequency and voltage while respecting the contractual values. As the industry developed and grew so did the energy demand, thus the need to increase the number of power plants, increase power in existing plants and its respective connection to the existing grid. This increase in complexity of the power system, resulted in a new control techniques being used, such as, automatic frequency control systems so that response time was faster and more adequate in production system, thus maintaining a constant equilibrium between active power produced and consumed + losses.

For a long period of time, the essential concept in the exploration of electric power systems remained the same that is, controlling the production through the system frequency with some efficiency gains in individual equipment control, namely the turbine-alternator set. In addition to the frequency control, there was also economic dispatch with the production costs in each power plant with system operators present in different points of the network. At that time, the power system security was essentially achieved through a high reserve margin available in the generation and transmission systems, in a way that unpredicted increases in electric energy consumption or equipment faults wouldn't cause big disturbance in the system resulting in prolonged downtimes and industrial production cuts.

This was not, by far, an ideal exploration technique thus, as the energy costs increased steeply during the last decades of the 20^{th} century and the growth in energy consumption, with cities growing at fast rates, resulted in a significant evolution in dimension and complexity of the electric energy systems, consequently the need to find alternative exploration and control philosophies. One of the first alternatives the electric companies tried was using computer simulation to aid them in their decisions, however it soon became apparent that, since the simulation was not in real-time and the real system configuration is constantly changing, either because of faulty equipments causing changes in configuration or different load diagrams than was predicted, the simulated system configuration was very different compared to the real one. Therefore, power companies had to find a better way to explore the systems, so electric networks started being explored as a centralized system and not as small clusters or individually controlled components. This meant that the electric system would be controlled in real-time with all its inherent advantages, however they needed to mix two different control components that were previously executed separately, them being data acquisition and effective control of the system. For this idea to work they needed to send periodic information about the system variables in real-time to the control center, thus they used a system monitor to read the data from the sensors sweeping them periodically. This allows a very good and as most exact as possible view of the electric system, but requires investment in good telemetry systems throughout the electric network.

Current technology is able to get good information from the network, however gross errors will certainly exist, namely due to the following:

- Bad measurement device calibration;
- Lack of measurement;
- Telemetry equipment errors due to malfunction/noise in the transmission;
- Non-simultaneous measurements;
- Measurements taken during transient incidents;
- Inaccurate network parameters;
- Network configuration errors due to errors in information (open/closed) from the switches;
- Errors in the mathematical model for not being correctly validated;
- Errors due to system asymmetries, namely unbalanced phases.

Electric power system's reliability and security is an issue that's been increasingly more important for power companies, because of stricter regulation (through downtime/quality penalties) and also consumers that don't tolerate power outages. On the other hand, power

companies, especially distribution network operators, were inefficient, thus they had to start making big investments to improve efficiency and reduce operating costs. One way to improve efficiency is with the use of real-time central control systems that allow for a better exploitation of the existing equipments.

In the 21th century there is a new paradigm in energy called distributed generation. Since the Kyoto protocol, its greenhouse gas emission reduction plan, and the ever increasing effects of global warming in the environment, causing extreme climate conditions, public interest in alternative energy production increased. Moreover, with the European governments' subsidies and differentiated tariffs between selling to and buying from the grid, the distributed micro-generation energy systems, such as hydro, solar photovoltaic, wind, biogas, biomass, have increasing demand. As this distributed generation is integrated in the existing power network infrastructure new challenges are presented, especially since some is very intermittent technology (wind, solar) which requires a very good communication and processing capabilities between the sensors and the control system to react to sudden changes in power generation. It is also important to forecast the evolution in demand and how much distributed power can be added to a certain point of the network without big investments.

One of the main particularities of Distribution System State Estimation is the lack of real-time measurements. In distribution, real-time measurements are typically found at the main substations; lines and loads are not usually monitored – not even low voltage substations. There is a generalized uncertainty about the power demand conditions and the line loading, thus pseudo-measurements need to be introduced in order that the state estimation mathematical models can be established and a unique solution can be obtained.

In order for Distribution System State Estimation to be an effective and useful tool, acceptable accuracy has to be achieved. For this to happen, real-time measurements need to be introduced in addition to pseudo-measures [1]. However, the fact that the majority of measurements used in state estimation are pseudo-measurements reveals the paramount importance of their appropriate modeling, so that they represent the network conditions as realistic as possible.

In recent years, a lot of changes occurred in the way that every power system operates. Large penetration of small scale generation – widely known by the term distributed generation – is now a common feature of many distribution networks.

Active distribution networks seem to be panacea to all the problems caused by distributed generation and it is widely accepted that active management of distribution network will increase the capacity of distributed generation that the distribution network can accommodate. A necessary function for the materialization of active distribution networks is the state estimation function.

The monitoring of complex systems is usually performed by using a set of sensors which measure the quantities being controlled (mechanical stress in airplanes or buildings; voltage, current or power in electrical networks; flow in water distribution networks, etc.) The choice of the number and type of sensors and their location is certainly the first task to be solved and probably may arise when some sensor locations prove to be either impossible or too expensive.

1.2 Objectives

In a power system, certain degrees of redundancy are present among the data collected from various measurements across the grid. An autoencoder is used to capture these correlations present between all of the data. Then restoration algorithms will use the autoencoder to search for missing measurements.

The purpose of this dissertation is therefore to apply Missing Sensor Restoration with Autoencoders to Power Systems in order to generate pseudo-measurements or to recover measurements from sensors, which for some reason are missing.

Furthermore, because of the nature of this thesis, the aim is to prove the concept and not develop a commercially efficient and robust model.

The missing sensor restoration algorithms with autoencoders will be tested first in an example, the parabola, and then in a power system, the IEEE 24 bus Reliability Test System.

1.3 Brief thesis organization

This thesis is organized in 5 chapters and two annexes, where this introductory chapter is the first one.

In Chapter 2 a brief state of the art relating to autoencoders is described, including an introduction on the history of neural networks. It will be described the characteristics of the autoencoders, their current applications and structure.

Chapter 3 is used to introduce missing sensor restoration with autoencoders. On this chapter are presented three main restoration techniques that use autoencoders and a fourth hybrid technique that combines two restoration algorithms. The first recovery technique combines the autoencoder with alternating projection onto convex sets, a powerful, fast and simple algorithm that has been used in many applications, namely tomography. Then will be described the unconstrained with EPSO, constrained search with EPSO and a hybrid algorithm that join alternating projections onto convex sets and constrained search with EPSO.

In Chapter 4, the described restoration algorithms will be applied to an example, the parabola. This example verifies the autoencoders ability to properly restore missing inputs and determines the performances of the various recovery techniques used.

In Chapter 5, the IEEE 24-bus Reliability Test System is used to evaluate the application of the mentioned techniques on power systems. The recovery schemes will be used to restore values of voltage amplitude and voltage angle, active and reactive injected powers, and switching device status.

Chapter 6 presents some relevant conclusions and also includes some perspectives of future work, related with the problems and new concepts introduced.

This Thesis ends with two Annexes. Annex A includes the EPSO algorithm formulation. In Annex B, the IEEE 24-bus network data parameters and topology are presented.

Chapter 2

Brief description of the state of the art of Autoencoders

2.1 Neural Networks

2.1.1 Introduction

We, humans and other animal beings perform various complex nonlinear tasks with the aid of information processing via biological neural networks. This is done by the huge amount of complex interconnections of the neuronal cells (neurons) which interact amongst each other by exchanging brief electrical pulses. The biological neural network is the motivation of its computer science version, popularly known as artificial neural network (ANN). Many scientists and engineers, however, often drop the initial artificial tag to name it just neural network (NN).

Inspired by the biological nervous system, ANN operates on the principle of largely interconnected simple elements operating as a network function. In doing so, no previous knowledge is assumed, but data, records, measurements, observations are considered. ANN research stands on the fact of *learning* from data to mimic the biological capability of linear and nonlinear problem solving.

Basically, we can design and train neural networks for solving particular problems which are difficult to solve by the human beings or the conventional computational algorithms. The computational meaning of the training comes down to the adjustments of certain *weights* which are the key elements of the ANN. This is one of the key differences of the neural network approach to problem solving than conventional computational algorithms which work step-by-step. This adjustment of the weights takes place when the neural network tries to correlate the correspondence between the input and target by adjusting its weights. Following many iterations of incremental training with many input-output pairs, the neural network weights are optimally adjusted by comparing its output and the designed target. This depends on various factors, like, the learning algorithm, network architecture, etc [2].

2.1.2 History and Background

The first basic neuronal model was proposed in the 1940s by two neuroscientists, Warren McCulloch and Walter Pitts (McCulloch and Pitts 1943). McCulloch and Pitts model of neuron contained the concept of spikes (action potential) carrying information. According to their model, the spikes carry information across the interconnected network of brain: each spike representing a binary 1 (Boolean TRUE), and lack of information or spike representing a binary 0 (Boolean FALSE). The McCulloch-Pitts neuron was much like the operation of a transistor. However the basic strength and significance of their model was establishment of the neurons as computational elements. The idea behind this model was highly responsible for the design of the modern digital computers by John von Neumann and later on, the ANN. Ideas like threshold, interconnection etc can be considered to be directly reciprocated from the McCulloch-Pitts model to the modern ANN [2].

McCulloch, who was by training a psychiatrist and neuroanatomist, spent some twenty years thinking about the representation of event in the nervous system. From 1941 to 1951 he worked in Chicago. Chicago at that time was one of the centers of neural of network research, mainly through the work of the Rashevsky group in the Committee on Mathematical Biology at the University of Chicago. Rashevsky, Landahl, Rapaport and Shimbel, among others, carried out many early investigations of the dynamics of neural networks, using a mixture of calculus and algebra. In 1942 McCulloch was introduced to Walter Pitts, then a 17 year old student of Rashevsky's. Pitts was a mathematical prodigy who had joined the Committee sometime in 1941. In any event Pitts was already working on algebraic aspects of neural networks, and it did not take him long to see the point behind McCulloch's quest for the embodiment of mind.

It was Beurle (1956) however, who first provided a detailed analysis of the triggering an propagation of large-scale brain activity. Beurle focused, not on the activation of individual neurons, but on the proportion of neurons becoming activated per unit time in a given volume element of a slab model brain tissue consisting of randomly connected neurons. In modern terms, this is the *continuum approximation* neural activity. Beurle's work triggered many computer simulations of randomly connected neural networks.

It was Hebb's proposal of synaptic modification during learning, however, that triggered even more work on neural networks; specifically on adaptive networks which could learn to perform specified tasks. Early work toward this goal was carried out by Uttley (1956), who demonstrated that neural networks with modifiable connections could indeed learn to classify simple sets of binary patterns into equivalence classes. Uttley's first suggestion was that synaptic weights represent conditional probabilities [3].

Following this initial period of interest, some limitations were identified by Minsky and Papert when they published their book [4], in which they showed the deficiencies of perceptron models, stirring up a general feeling of frustration among researchers.

In consequence to the cooling of the initial enthusiasm, the field had to survive to a period of increase frustration and disrepute, having only a relatively few researchers continued their efforts in order to suppress the limitations identified by Minsky and Papert.

The interest in neural networks have re-emerged in the early 80's, only after some important theoretical developments were attained, such as the notably discovery of error back-propagation, and most significant, new hardware was developed, which at the time opened up a all other world of infinity possibilities for a large number of research fields, with

the increased processing capacities. With the appearance of these new technologies and theoretical knowledge's, and mainly because of the perseverance showed by the few remaining neural networks researchers, the neural network concept end development could not be left behind, and so it was revived the interest in this field.

This renewed enthusiasm has been reflected in the number of researchers and the corresponding increase in funding, conferences and papers related with neural networks. Another important sign of the current believe in neural networks is that nowadays it is common to engineering universities to teach and pass the knowledge of neural networks.

However, I believe that it is important for us to keep in mind that there is still so little we know about the biological systems upon which the neural networks concept is developed, so it is extremely important that further studies are made and that hopefully they would show new possible applications and future developments in this area.

The work developed for this dissertation pretends to be one of them, presenting a new application of artificial neural networks that will allow modeling and recovery of measurements on power systems in a more robust, efficient and new way.

2.1.3 Applications

Neural networks are applied widely for solving different problems which in general are difficult to solve by humans or conventional computational algorithms. A comprehensive list of neural networks applications including applications like adaptive channel equalizer, word recognizer, process monitor, sonar classifier, risk analysis systems. A list is provided below which gives an idea of the versatile uses of the neural networks for different applications.

Since neural networks are best at identifying patterns or trends in data, they are well suited for all kinds of prediction or forecasting, including:

- Sales forecasting;
- Meteorology forecasting;
- Wind energy source prediction;
- Analysis of risk management;

However these are only a micro-sample of the current applications for neural networks, since that an ANN has a very wide range of successful applications to the real world problems.

2.1.4 Artificial Neural Networks (ANN)

2.1.4.1 Basic structure of the Artificial Networks

Biological neuronal network comprises of numerous (in the order of trillion) interconnections among the neurons. The power of neuron comes from its collective behavior in a network where all neurons are interconnected. Hypothetically, it is assumed that in a human brain approximately 100 billion neurons are acting collectively via approximately 10^{14} interconnections. Similarly, ANN also develops from the interconnections of several unit

neurons or nodes. The arrangement of the neurons is quite arbitrary. It depends on several factors, like, the nature of application, inspiration from real biological structure seen under the microscope and so forth. However, most commonly, as a rule, in the artificial networks, the following layers of neurons are used:

- **Input layer**: Number of neurons in this layer corresponds to the number of inputs to the neuronal network. This layer consists of passive nodes, i.e., which do not take part in the actual signal modification, but only transmits the signal to the following layer;

- **Hidden layer**: This layer has arbitrary number of layers with arbitrary number of neurons. The nodes in this layer take part in the signal modification, hence, they are active;

- **Output layer**: The number of neurons in the output layer corresponds to the number of the output values of the neural network. The nodes in this layer are active ones [2].

The basic structure of the ANN is shown in Fig. 2.1.

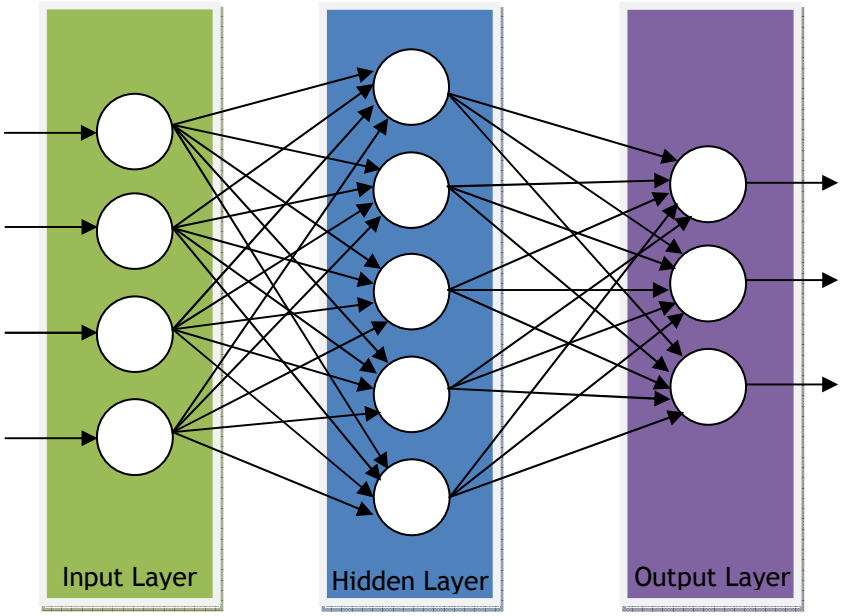

Figure 2.1 - Basic Structure of a Feedforward Artificial Neural Network.

2.1.4.2 Structure of a neuron

A brief description of the typical structure of the biological neuron is given below to compare with the ANN structure. A typical biological neuron consists of three main parts:

- **Cell body**: The cell is the most important part of the neuron. It contains the nucleus and is responsible for the information processing;
- **Axon**: The axon carries the nerve signals away from the neuron (cell body). Each neuron contains only one axon, usually heavily branched, in order to reach as many other neurons as possible. The axon connects to other neurons through so called synapses;
- **Dendrites**: The dendrites are the nerve endings. They carry incoming signals from other neurons to the cell body.

The operation of the biological neuron can be summarized as the signals from other neurons are summed together and compared against a threshold to determine if the neuron shall excite ("fire"). This is the motivation for the structure of the ANN. For ANN, the structure of a neuron mainly consists of the sum and squash unit. The inputs pass through the specific weights and then the weighted inputs are summed. A weight is the strength of the connection between two neurons. The weighted sum is then passed through a transfer function to produce the final output. The transfer function is chosen to map the input(s) to the output(s) [2].

The structure of an artificial neuron vis-à-vis biological counterpart is shown in Fig. 2.2. In Fig. 2.2, a fully interconnected structure is shown, i.e., where all the input values are passed onto the hidden layer. The operation of the network is as follows. The input values, passed to the hidden layer from the input layer, are multiplied by the weights, the predetermined numbers (processing units of the ANN). The weighted inputs after passing through the transfer function (shown as sigmoid in Fig. 2.2) produce the outputs. In a network learning operation, this output is compared against the objective target value and the deviation is fed back to modify the weights. This process is iterated until the output correctly approximates the target, i.e., the error is gradually decreased to the minimum. In the end, the neural network gets fully trained, i.e., the weights achieve optimum values (error becomes minimum) to produce the output within a specified accuracy level corresponding to the input data.

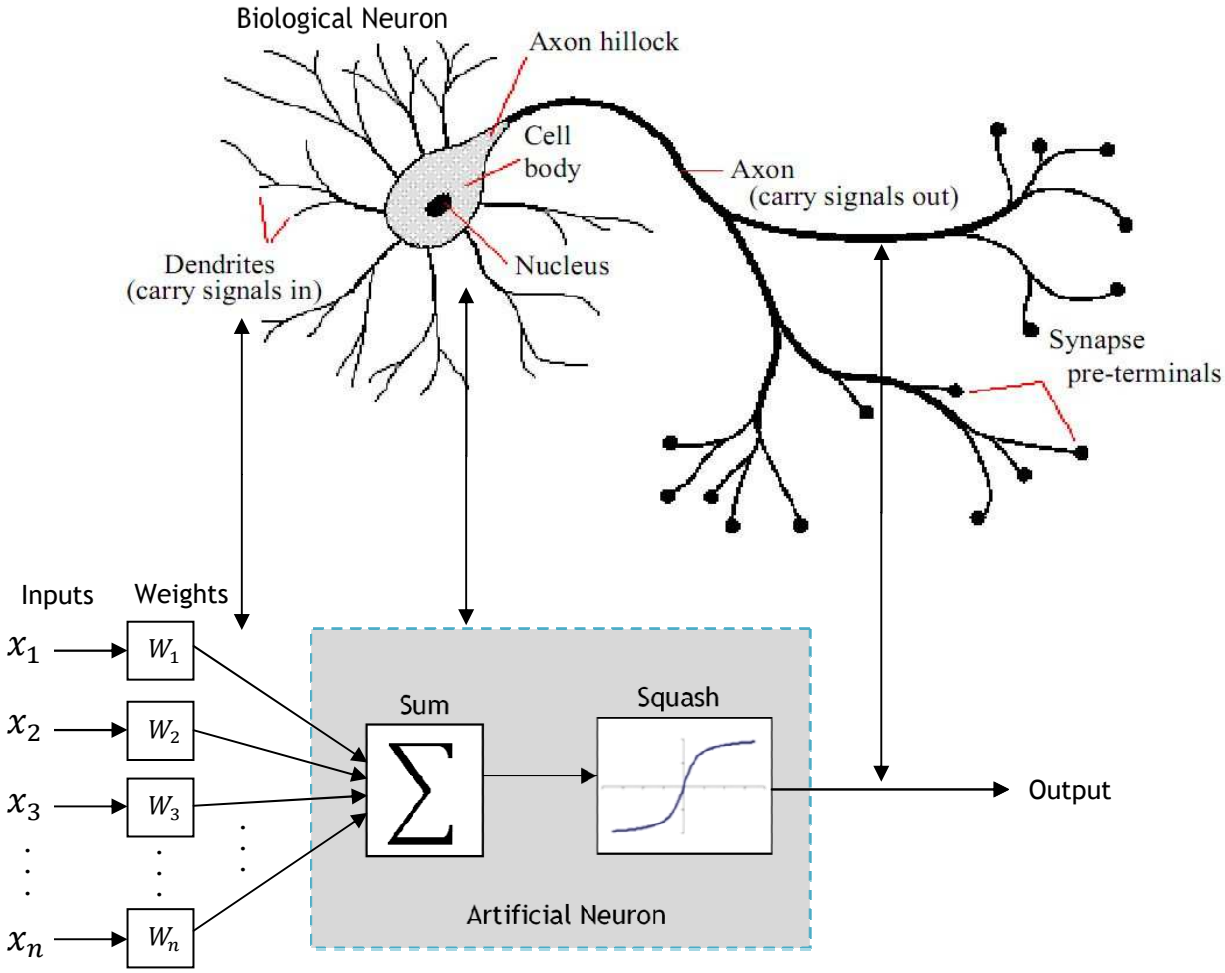

Figure 2.2 - Comparison between Biological and Artificial Neurons [5].

A detailed view on the artificial neuron is shown on figure 2.3. It represents an artificial neuron with “m” inputs and one output.

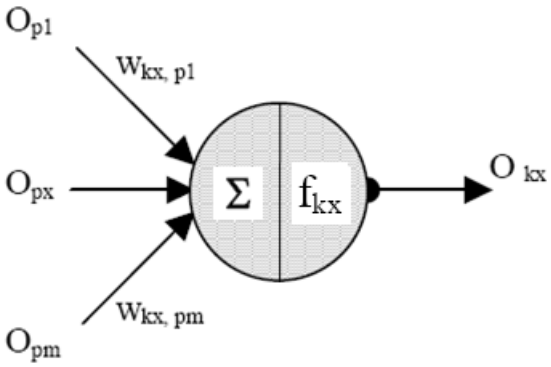

Figure 2.3 - Artificial "neuron" scheme [5].

From a mathematical point of view, the neuron output can be represented by equation 2.1, where the outputs from neurons “l”, “x” and “m” from layer “p”, once multiplied by the respective weight values that determines the connection weight between two neurons, will be the inputs to neuron “x” from layer “k”.

$$O_{kx} = f_{kx}\left(\sum_{pi=0}^{pn}(W_{kx,pi} \times O_{pi})\right) \quad \textbf{(2.1)}$$

“ f_{kx}” is the neuron activation function that will weight how powerful the output (if any) should be from the neuron, based on the sum of the inputs. This activation function can be modeled by various types of functions, some of the most commonly used are the threshold, sigmoid and step wise. This way the output from this neuron will be one of the inputs of some neurons of the next layer.

The structure of an artificial neural network can be seen as a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. This artificial neuron have a large number of entries (electric pulses) and produces only one electric pulse as it output, which is the result of an internal non-linear process that transforms the inputs sum in the output. As dictated in figure 2.1, it is usual to see these “neurons” organized by layers with feed forward connections, or in other words unidirectional connections towards the next layer, from the input of the network to the output. For each connection is defined a weight, which will multiply by the “neuron” output signal before the signal gets to the input of the next “neuron”.

One of the most important features of an artificial neural network emerges from the train concept, contrarily to the concept of calculus or the concept of analytic solution determination. On a human nervous system, the synaptic connections change throughout the life time of a neuron and the amount of incoming impulses needed to activate a neuron (threshold) also change. These changes allow the brain to adapt it responses to human body and the associated needs making the information processing more accurately. It is upon this “adaptation” concept that emerges the concept of training the artificial neural network, providing an adaptation of ANN’s internal parameters in order to obtain the desired behavior or output from the neural network in response to the inputs.

The internal parameters that are adapted in the train are:

- The weight associated to each connection;
- The internal non-linear function, when not defined externally by the user;
- The number of connections and the connections between each pair of neurons;
- And in some cases the number of neurons in the hidden layers and the number of hidden layers;

Summarily, an ANN pretends to implement computationally the information processing capacity of a brain, assuming a structure similar to a biological neural network, considering the main features, in order to be capable of solving large complex problems that would be impossible or complex to solve by an analytic process. ANNs are not intelligent, but they are good for recognizing data patterns and making simple rules for large complex problems. They also have excellent training capabilities which explain why they are often used in artificial intelligence research [5].

2.1.4.3 Transfer function

Transfer function in the ANN maps the input(s) to the output(s). Hence, it is an important element of the network for a successful network design. It is the key element to invoke the nonlinear relationship between the input and the output. Without the transfer function the whole operation is linear and could be solved using linear algebra or similar methods [2].

The search of the most efficient activation functions for the neurons in the middle layer and for the neurons of the output layer is an important process and depends of the characteristics of the problem in question. The neuron activation function is the one responsible to weight how powerful the output (if any) should be from the neuron, based on the sum of the inputs. This activation function can be modeled by various types of functions, some of the most commonly used are the threshold, sigmoid, Gaussian, linear and step wise but it can also be used the Elliot, the sin and cosh activation functions.

The adopted activation function for all the neurons of the autoencoder in the hidden layer is the symmetric sigmoid function illustrated in equation (2.2), since it has been shown to be the most suitable function to improve the performance for the autoencoder. For the output layer, the activation function selected was, in some cases, the symmetric sigmoid and in others the linear activation function, depending of which has better performance.

$$\begin{cases} y = \dfrac{2}{1+\exp(-2sx)} - 1 \\ -1 \le y \le 1 \end{cases} \tag{2.2}$$

Where:

y – Output of the respective neuron;

x – Sum of all the input of the respective neuron, which correspond to the outputs from the neurons of the previous layer;

s – Steepness value of the activation function;

The steepness of an activation function says something about how fast the activation function goes from the minimum to the maximum. A high value for the activation function will also give a more aggressive training. One has opted by a steepness value of 0.5 in order to avoid a very aggressive training.

The sigmoid activation function is one of the most used activation functions, being efficient on most of the applications for neural networks. For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, however all three must be considered rough approximations to a real neuron.

2.1.4.4 Autoassociative Neural Network (Autoencoder)

Autoencoder networks are feedforward neural networks used for learning efficient codings. These networks attempt to reconstruct the input data at the output layer. The targets at the output layer are the same as the input data, thus the size of the output layer is also the same as the size of the input layer. Autoencoders are supervised neural networks which are trained using a gradient descent method, such as backpropagation. Since the size of the hidden layer in an autoencoder is smaller than the size of the input data, the dimensionality of input data is reduced (encoding) to a smaller dimensional code space at the hidden layer. This means it is being used for dimensionality reduction [6]. The outputs from the hidden layer are then reconstructed into the original data at the output layer. Like Principal Component Analysis (PCA) [7], the autoencoders can give mappings in both directions between the data and the code space.

By using more number of hidden layers in an autoencoder neural network, a high-dimensional input data can be reduced to a much smaller code space. However, training a multilayer network with more than one hidden layer is tedious and further it will not also give good results. This is due to the fact that the weights at deep hidden layers are hardly optimized. The time required for training such network is also extremely high.

2.1.4.5 Structural definition

The architecture of a three layer autoencoder neural network is depicted in Figure 2.4. An autoencoder is a feedforward neural network with one or more hidden layers that attempts to reconstruct its input activities at its output. Hence the main difference of the autoencoder and the traditional neural network is the size of the output layer. In an autoencoder the size of the output layer is always the same as the size of the input layer. Besides that, the size of the deepest hidden layer of an autoencoder network is always smaller than the sizes of the input and output layers. As shown in Figure 2.4, the autoencoder network comprises two components namely “encoder” and “decoder”. “Encoder” network transforms the high-dimensional input data into a low-dimensional code and the “decoder”

network (which is similar to the "encoder" network) reconstructs the original high-dimensional data from the low-dimensional code.

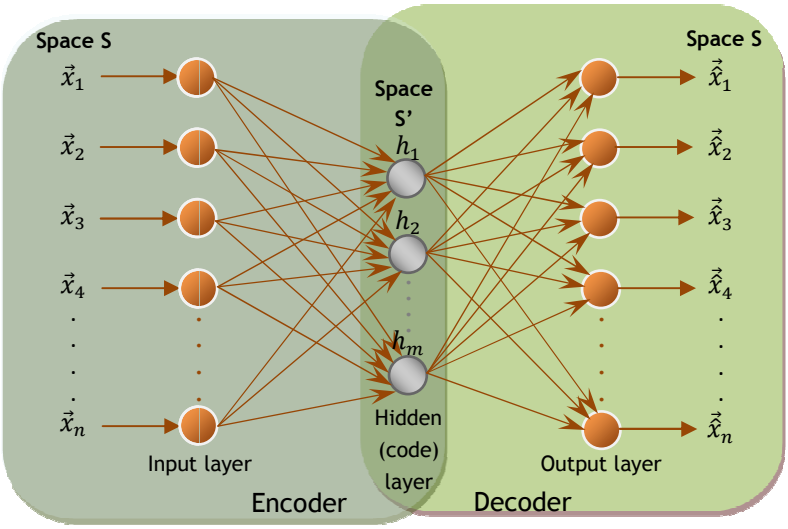

Figure 2.4 - Three layer autoencoder architecture.

So in order to define the autoencoder structure one has to define 4 aspects:

- Number of hidden layers;
- Number of neurons for the hidden layers;
- Neurons activation functions;
- Connections between neurons of different layers.

In process to define the most desirable number of hidden layers and the actual number of neurons in each hidden layer one has to take into account that this decisions will directly affect both the running time of executing and training an artificial neural network as well it will affect the autoencoder capability to perform a good encode/decode process for the original solution space dimension.

Moreover the autoencoder must be capable of properly represent the different solutions of input space in the reduced code space in order to allow the Missing Sensor Restoration algorithm (MSR) to perform a proper evolution towards the missing sensor data (MISED).

One has tried different configurations for the neural network autoencoder. However configurations with more than one hidden layer has shown to be less efficient in the encoding/decoding procedure and more demanding in computational effort on execution and in training. So analyzing these results one as opted to adopt an autoencoder with only one inner layer to represent the reduced solution (code) space.

Having defined the most desirable structure for the autoencoder one needs to define the proper dimension for the inner layer. To do this, one has made several experiences considering different coded space dimensions in order to establish the most advantageous number of neurons for the middle layer comparing the behavior in the recovery process. Theoretically the increase of loss information is proportional to the increase in the reduction of solution space dimension S, moreover there will be a limit for how much it is possible to reduce the S', once after that limit it would be impossible to decode S' to the original space S and therefore the restoration mechanism can no longer retrieve the missing input(s).

Now focusing on the pattern of connections between the neurons of different layers and the respective propagation of data, there are two main distinct types of networks: the Feed-forward networks and the recurrent networks or Feed-back networks. The last ones have connections between neurons contrarily to the normal direction of the data propagation throughout the network. The Autoencoder neural network is a Feed-forward network, where the data flow from the input to output neurons is strictly forward. The data processing extends over multiple units, but no feed-back connections from the outputs to inputs of neurons in the same layer or previous layers are considered. It has been shown (Hornik, Stinchcombe [8]) that only one layer of hidden units suffices to approximate much kind of functions, provided that the activation functions of the hidden layer neurons are non-linear. The process of establishing the connections between neurons is strictly an integrant part of the training procedure; where the training algorithm will define the connections to be considered in order to improve the performance of the neural network to perform the desired encode/decode procedure.

2.1.4.6 Applications

Autoencoders has been applied to missing sensor data restoration [9],principal component analysis [7], pattern classification, object and data recognition [10], non-linear data dimension reduction [6],feature detection, etc.

2.1.4.7 Autoencoder training procedure

In order to learn and understand how to react or proceed under certain circumstances, one needs to successfully pass throughout a learning process that would provide one with reasoning power to look for and choose the right step to make between all the possible decisions in the solution space. In the same way, in order to make it possible to train an autoencoder neural network that would properly encode a solution space (composed by “n” variables that define the solution) and then be capable of reproducing at the end the same original solution space, it is necessary to submit the autoencoder to a similar concept of learning procedure.

In a superficial definition, this learning technique as any other learning process in life is no more than a “trial and error” method, where for several scenarios or input data the autoencoder will produce an output vector that will be compared to the desired output. If the actual output is too far from the desired one it will be submitted to the input data again, adjusting its internal parameters, in order to produce a good approximation of the actual output to the desirable one. It is possible in a certain way to compare this to the learning process that happens with human brain’s, being this kind of learning based mostly on experience.

In the training procedure of an artificial neural network one must ensure that mainly three important aspects are fulfilled:

- Ensure that there is no “noise” on the input data that will cause the artificial neural network train to fail its objective.
- Guarantee the generality of recognition of possible solutions in the space by the autoencoder. In other words, ensure that most of the solution space is represented in the input data used to train the network.
- Ensure that the artificial neural network is not over-trained, in order to avoid that the autoencoder performs correctly the encoding and decoding procedures only for the input data used in the train, losing the capability to perform the same tasks with the same quality for different input data.

The third aspect relies only in the way the training procedure is executed and the stoppage criterion adopted for the train.

However, the fulfillment of those two other aspects mentioned above relies only in the quality and quantity of the gathered data with the purpose to train and test the performance of the autoencoder neural network. In fact it is necessary to obtain a large number of possible scenarios to train the autoencoder, consequently, in power systems these scenarios are achieved thought several loads, topologies and power flows.

The training and test sets are of groups of examples used during the learning phase for autoencoder training and for verifying the ANN’s generalization capabilities respectively. The validation set is used to evaluate autoencoder performance and must never be used during the learning phase [11].

2.1.4.8 Pre-treatment of the input and output data sets

Before the training of the autoencoder takes place, one must pre-treat the input and output train data set, transforming the range of the input and output values to a normalized interval of [-1, 1]. This scale adjustment process, known as standardization procedure, increases the performance and efficiency of the autoencoder training, once it allows a better adjustment of the input variables to the range of the activation function, and also allows the neural network to be less affected by the different ranges of the variables in the training data set.

There are three main methods to standardize the data:

- Z-core method.
- Decimal scaling method.
- Min-Max method.

The “Min-Max method” is the best standardization procedure when the minimum and maximum values of the data set are known. Therefore this is the method applied here to perform the standardization since that the minimum and maximum values of the variables that compose the input vectors can be easily obtained.

$$\boldsymbol{y'} = \frac{\boldsymbol{y - min_a}}{\boldsymbol{max_a - min_a}} \times (\boldsymbol{max_A - min_A}) + \boldsymbol{min_A} \tag{2.3}$$

Where:
y' – Standardized value for the considered variable;
y – Variable value in the original representation interval;
min_A – Minimum value of the standardized range of values (-1);
max_A – Maximum value of the standardized range of values (1);
min_a – Minimum value of the "original" range of values;
max_a – Maximum value of the "original" range of values;

2.1.4.9 Training algorithm

With the purpose of training the autoencoder properly, an adaptive gradient-based algorithm called "Resilient Back-Propagation algorithm" was adopted, which belongs to the most widely used class of algorithms for supervised learning of neural networks. This learning algorithm is an update of the "Back-Propagation algorithm" that considers a fixed learning rate to determine how the weights should evolve. However, it is well known that this approach tends to be inefficient.

So in order to overcome the inherent difficulty of choosing the right learning rates, adaptive gradient-based algorithms were developed, like the "Resilient Back-Propagation algorithm", which have individual step sizes by controlling the weight update for each weight in order to minimize oscillations and maximize the length of the step size. This way it becomes possible to speed-up the learning process and to easily avoid local optimums in the neural network training process.

The "RPROP" algorithm works in much the same way as the name suggests. After propagating an input through the autoencoder neural network, the error is calculated and then it is propagated back through the network while the weights are adjusted in order to make the error smaller.

Another training particularity is that aiming for a more efficient way to minimize the mean square error for all the training data set the training on data is executed sequentially one input at a time, instead of training on the combined data. This provides a very efficient way of avoiding getting stuck in local minima.

First the input is propagated through the ANN to the output, after which the error on a single output neuron can be calculated as:

$$\boldsymbol{e_k = d_k - y_k} \tag{2.4}$$

Where:
e_k – Error on the single neuron k;
d_k – Desired output for the neuron k;
y_k – Calculated output for the neuron k;

This error value is then used to calculate the δ_k value, which is again used for adjusting the weights. The δ_k value is calculated by:

$$\boldsymbol{\delta_k = e_k \times g'(y_k)} \tag{2.5}$$

Where:
$g'(y_k)$ – Derived activation function for the given neuron;

After being calculated the δ_k values, are calculated the δ_j values for preceding layers. The δ_j values are calculated from the δ_k values of this layer by the equation (4.6):

$$\delta_j = g'(y_j)\sum_{k=0}^{K}\delta_k w_{jk} \tag{2.6}$$

Where:
K – Total number of neurons in the layer;
w_{jk} – Connection weight between the neurons j and k;

Note that in the previous equation it is not considered the learning parameter. In fact, in this training algorithm a set of more advanced parameters provides a more qualified guess to how much the weight should be adjusted.

Using these δ values, are calculated the weight slopes according to the equation (2.7):

$$\Delta w_{jk} = \delta_j y_k = -\frac{\partial E}{\partial w_{jk}} \tag{2.7}$$

Where:
E – Batch error measurement defined as the sum of the mean square error for the entire training set;

In order to update the weights, for each weight if:

$$\frac{\partial E(t-1)}{\partial w_{jk}} \cdot \frac{\partial E(t)}{\partial w_{jk}} > 0 \tag{2.8}$$

Then

$$\Delta_{jk}(t) = \min(\Delta_{jk}(t-1)\cdot\eta^{+}\cdot\Delta_{\max}) \tag{2.9}$$

If:

$$\frac{\partial E(t-1)}{\partial w_{jk}} \cdot \frac{\partial E(t)}{\partial w_{jk}} < 0 \tag{2.10}$$

Then

$$\Delta_{jk}(t) = \min(\Delta_{jk}(t-1)\cdot\eta^{-}\cdot\Delta_{\min}) \tag{2.11}$$

Where:
Δ_{jk} – Step size between j and k;
Δ_{min} – Minimum weight step to prevent the weight from becoming too small;
Δ_{min} – Maximum weight step to prevent the weight from becoming too large;

Finally if slope is negative then:

$$w_{jk}(t) = w_{jk}(t-1) - \Delta_{jk}(t) \tag{2.12}$$

If slope is positive then:

$$w_{jk}(t) = w_{jk}(t-1) + \Delta_{jk}(t) \tag{2.13}$$

This process continues until a specified stopping criterion is reached. The stopping criterion that was used is the mean square error measurement of the validation data while training the network for the training data set input values. When the mean square error value reaches a certain pre-specified limit, the training is stopped. Moreover if the desirable value for the mean square error is never reached, the training can also be stopped by defining in the beginning of the training the maximum number of training epochs defined as the number of times that the all data set is applied to the inputs of the network [5].

Chapter 3

Missing Sensor Restoration with Autoencoders

3.1 Introduction

Fault tolerant measurements are an essential requirement for system identification, control and protection [12]. Measurements can be corrupted or interrupted due to sensor failure, broken or bad connections, bad communication, or malfunction of some hardware or software, etc. These corrupted or interrupted measurements are referred to as missing sensors or lost sensors hereafter in this thesis. In contrast, uncorrupted measurements are referred to as known or healthy sensors.

This thesis proposes a novel robust Missing Sensor Restoration scheme in power systems by combining an autoencoder and a restoration algorithm.

The suggested MSR algorithm relies on a sensor evaluation system or sensor monitor to determine the integrity of sensor data and therefore activate and tell MSR what sensors are missing.

A signal array can generate interdependent reading among the signals. If the dependence is sufficiently strong, the reading may contain redundancy to the degree that the readings from one or more lost signals may be able to be accurately estimated from those remaining. An autoassociative regression machine can learn the data interrelationships through inspection of historical data. Once trained, the autoassociative machine can be used to restore one or more arbitrary lost signals if the data dependency is sufficiently strong [13]. Recovery techniques include alternating projection onto convex sets (POCS) and iterative search algorithms like EPSO [14-15].

State estimation [1, 16-17] is commonly used to identify state variables that are not accessible for direct measurements. The technique can be modified to estimate sensor values.

To estimate the data of the missing sensor, the state estimation method is configured to identify specified systems states using the information from the healthy sensors only. The known states are used to estimate the data of the missing sensors through direct query of the system model. This process is not robust and the solution can be unfeasible for real system applications.

There are three main methods that we will examine for the restoration of missing sensors. The first is a simple application of alternating projections onto convex sets (POCS) [18-20]. The second and the third both employ search techniques.

The first search technique, unconstrained search, involves simply minimizing the error between the missing sensor inputs and outputs on the autoencoder, while the second, constrained search, looks at the error between the entire input pattern and output pattern (both missing and known sensors) to achieve a final answer. The merits of each are discussed below. First, however, some definitions are in order.

The input to the neural network is comprised of two concatenated vectors whose total dimension is N, the input dimension (and necessarily the output dimension as well). The first vector, $\vec{x}_k$, can be thought of as the operating point of the restoration process, and is defined as the set of known sensor values for a given input pattern. The second vector is then $\vec{x}_m$, the set of missing sensor values. Without loss of generality, let us formulate the input as some vector $\vec{x}$:

$$\vec{x} \equiv \{x_{m1}, x_{m2}, \dots, x_{mM}, x_{k1}, x_{k2}, \dots, x_{kK}\}^T, \tag{3.1}$$

where M is the number of missing sensors, K is the number of known sensors, and of course $K+M=N$.

Likewise, on the outputs, we have:

$$\vec{\hat{x}} \equiv \{\hat{x}_{m1}, \hat{x}_{m2}, \dots, \hat{x}_{mM}, \hat{x}_{k1}, \hat{x}_{k2}, \dots, \hat{x}_{kK}\}^T, \tag{3.2}$$

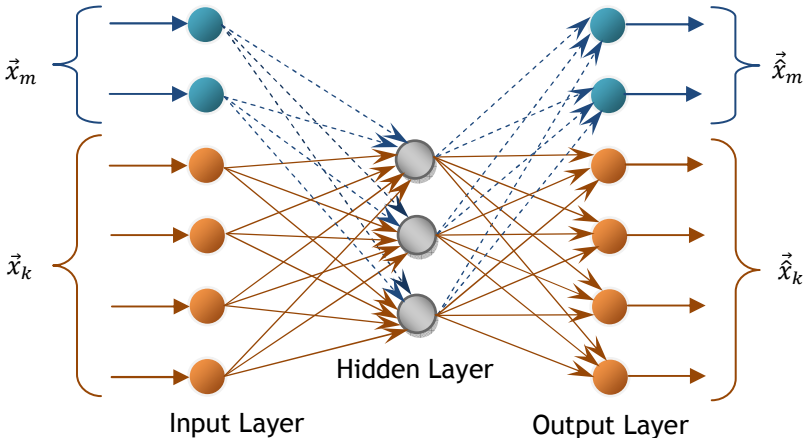

Figure 3.1 – A diagram of a generic 3-layer autoencoder.

3.2 Alternating Projections Onto Convex Sets (POCS)

One approach to restoration of the missing sensor reading is through the method of *alternation projection onto convex sets* (POCS). POCS has been applied to tomography, biomagnetic field reconstruction, signal analysis, image processing, radiography [18], holography, neural networks, and signal restoration.

For the failed sensor restoration problem, a two dimensional illustration of POCS is shown in Figure 3.2.

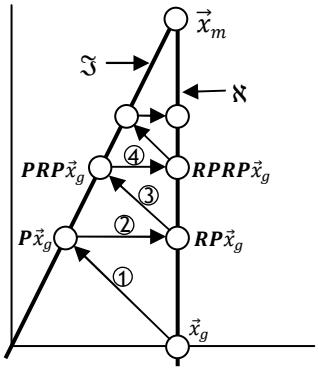

Figure 3.2 - Alternating projections onto convex sets (POCS) illustrated for two intersecting planes. By alternatingly projecting between the subspace labeled $\mathfrak{I}$ and the plane labeled א, convergence is to a point common to the intersection of both.

The linear manifold in Figure 3.2 on which the training data lie is dubbed $\mathfrak{I}$**.** The vector containing sensor data to be restored is denoted by the vector $\vec{x}_m$(for missing). Clearly, the unknown $\vec{x}_m \in \mathfrak{I}$. The vector of sensor readings, with the failed sensor readings set to zero, is denoted by $\vec{x}g$ (for given). The first step is to project the given vector onto the linear manifold defined by the training data. The result after step ①, illustrated in Figure 3.2, is $\boldsymbol{P}\vec{x}g$. This vector, however, cannot be the restoration because the vector entries corresponding to working sensors do not contain the correct readings. Step ②, then, is to replace these vector components with the known correct values. The operation of replacing these vector components with the measured values while retaining whatever values are in the failed sensor location is denoted by **R**. The set of all vectors containing the measured values of the working sensors with arbitrary entries corresponding to the failed sensors forms a linear variety (plane), א, illustrated in Figure 3.2 by a vertical line. The operator, **R**, projects onto the set א the result of step ② , after replacing the appropriate vector components to the correct values is, as shown in Figure 3.2, $\boldsymbol{RP}\vec{x}_m$. The projection is repeated in step ③ followed by insertion of the readings from the working sensors in step ④. The iteration is repeated. The final destination of the iteration is the intersection of א and $\mathfrak{I}$ which is, of course, the vector containing correct values for the failed sensors, $\vec{x}_m$.

Reconstruction criteria can be established for two or more nonlinear convex sets with nonzero intersection will result in convergence to a fixed point lying in the intersection of the sets.

Under the assumption of convexity, our two sets are then S_1 the space defined as the output of the autoencoder, and S_2 the set of all input patterns to the neural network containing $\vec{x}_k$, the known sensors, and an arbitrary $\vec{x}_m$. While the second set is definitely convex, the first requires the assumption of convexity. By choosing some initial $\vec{x}_m$ to create an input vector $\vec{x}$, we then obtain $\vec{\hat{x}}$, the output of the autoencoder. This corresponds to a projection onto the first set, the operator for which we will denote P_1. We then change the outputs $\vec{\hat{x}}_k$ to $\vec{x}_k$ to perform the projection onto the second set, denoted as P_2. If we alternate between these projections, under the assumption of convexity, the series will converge to an answer representing the intersection of the two sets. Thus, a single iteration of this process is defined as the successive application of P_1 and then P_2.

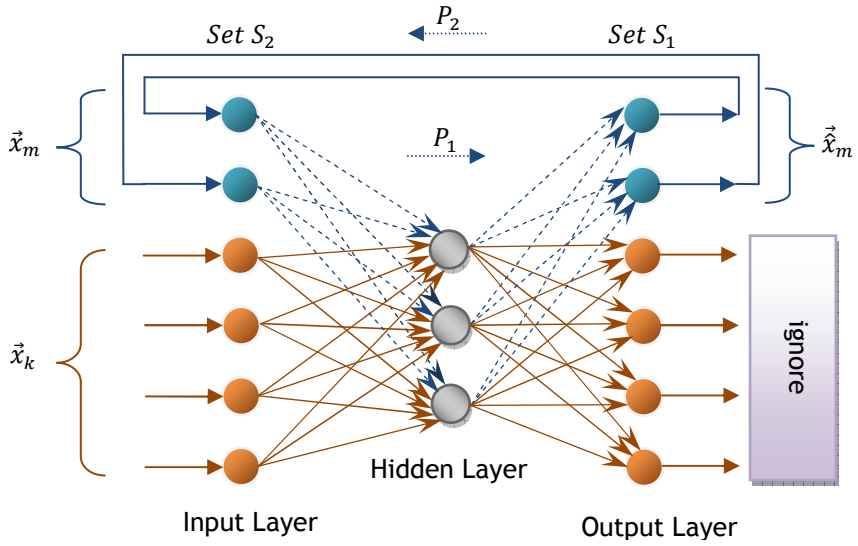

Figure 3.3 – Using a trained autoassociative encoder to perform the POCS operation.

The two top input nodes, on the left, correspond to two missing sensor values. The remainders of the input nodes are fed the values of the operational sensors. The outputs of the top two nodes are fed back to the input with unit gain. The known inputs continue to supply the values obtained from the working sensors. Through POCS theory, the top two nodes will converge, under general conditions, to the values that would have been recorded by the failed sensors.

3.3 Unconstrained Search with EPSO

Because of the potential lack of convexity and other performance issues, we are motivated to find a better method for discovering the true point of convergence. In this case, our iterative operator is simply a single iteration of EPSO which seeks to minimize the error between the missing sensor values and the outputs of the autoencoder corresponding to those missing sensor values, or:

$$\underset{\vec{x}_m}{argmin} \left\| \vec{x}_m - \vec{\hat{x}}_m \right\|, \tag{3.3}$$

This method allows for greater refinement of the missing sensor values over the POCS method described above. Moreover, it should be noted that, if the assumption of convexity were true, both POCS and the unconstrained search [9] would converge to the same value, assuming the two sets intersect.

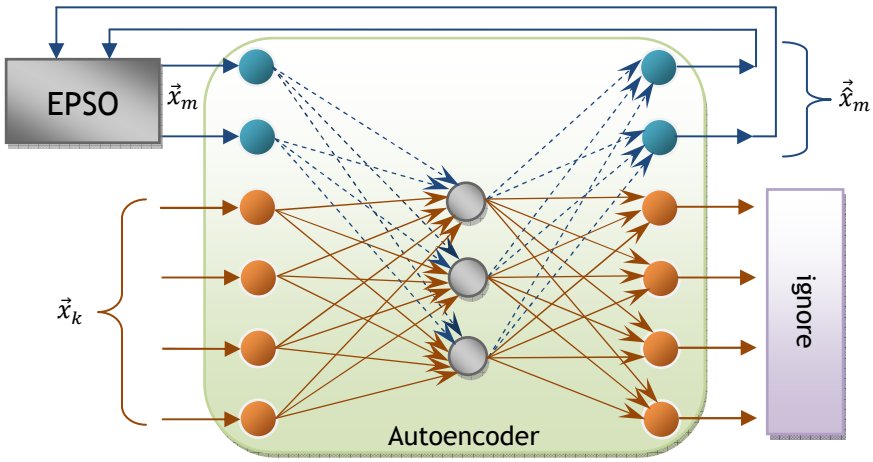

Figure 3.4 - Using a trained autoassociative encoder to perform unconstrained search with EPSO.

The number of missing sensors (M) corresponds to the number of dimensions in which EPSO will perform the search. An increase in the number of missing sensors makes the search more difficult, as we also increase the search space.

This algorithm is especially useful in cases where we have a low efficiency autoencoder or lack of convexity, since unconstrained search (and contrary to POCS) doesn't depend on the autoencoder efficiency. When the number of missing sensors is great so is the possible lack of convexity.

3.4 Constrained Search with EPSO

A notable shortcoming of both POCS and the unconstrained search is that neither one uses the information contained in $\vec{x}_k$ to better refine the final answer. Thus, we define a third method, constrained search, which is similar to the unconstrained search except that it corresponds to a search using EPSO which seeks to minimize the entire output error of the autoencoder, namely:

$$\underset{\vec{x}}{argmin}\left\|\vec{x}-\hat{\vec{x}}\right\|, \tag{3.4}$$

recalling that $\vec{x}$ is a vector composed of $\vec{x}_m$ and $\vec{x}_k$. This way, we actually ensure a smoother match between the input and the output, which can help eliminate spurious answers that, while minimizing the error between consecutive iterations on $\vec{x}_m$, tend not to make sense in the context of the known sensors.

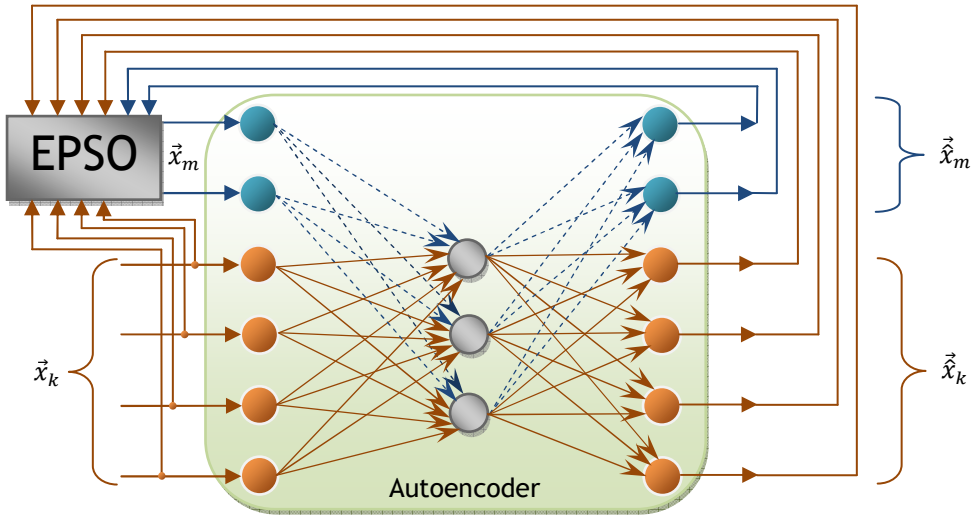

Figure 3.5 - Using a trained autoassociative encoder to perform constrained search with EPSO.

As well as unconstrained search, in constrained search with EPSO the number of missing sensors (M) corresponds to the number of dimensions in which EPSO will perform the search.

3.5 Hybrid strategy: POCS+Constrained Search with EPSO

Alternating projections is a very simple algorithm for computing a point in the intersection of some convex sets, using a sequence of projections onto the sets. Like a gradient or subgradient method, alternating projections can be slow, but the method can be useful when we have an efficient autoencoder.

Alternating projections onto convex sets is a powerful technique of signal recovery and synthesis. Although its performance usually degrades over algorithm evolution in most cases the POCS method has, initially, a stronger convergence than any iterative search algorithm. That's the idea behind the hybrid strategy: join the initial convergence of POCS with the refined and smoother final answer of constrained search.

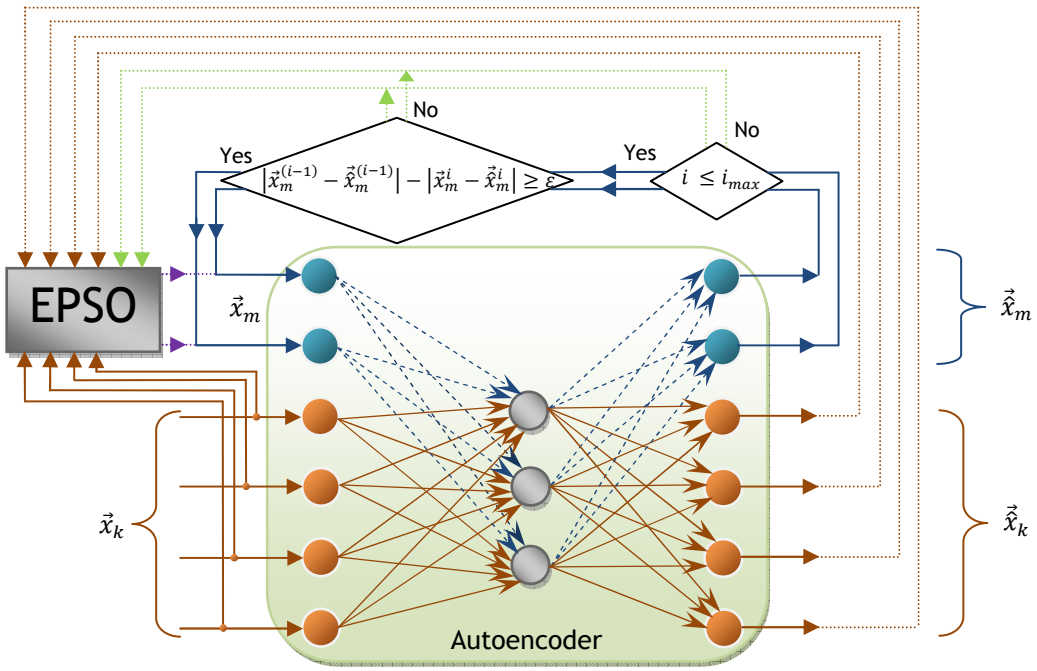

Figure 3.6 - Hybrid strategy: POCS followed by constrained search with EPSO.

The hybrid algorithm developed starts by using POCS until its performance degrades to a value of ε defined by equation 3.5 or it reaches the maximum number of iterations ($i \leq i_{max}$):

$$\left|\vec{x}_m^{(i-1)} - \vec{\hat{x}}_m^{(i-1)}\right| - \left|\vec{x}_m^{i} - \vec{\hat{x}}_m^{i}\right| = \varepsilon, \tag{3.5}$$

Where:

$\vec{x}_m^i$ - vector of missing input(s) at iteration i;
$\vec{\hat{x}}_m^i$ - vector of output(s) for the missing input(s) at iteration i;
$\vec{x}_m^{(i-1)}$ - vector of missing input(s) at iteration i-1;
$\vec{\hat{x}}_m^{(i-1)}$ - vector of output(s) for the missing input(s) at iteration i-1;

Equation 3.5 compares differences between the autoencoder input(s) and output(s) of missing sensor(s) for the current POCS iteration with the same difference for the last POCS iteration. If this difference (ε) is small so is the POCS convergence and is better to change to constrained search with EPSO.

The hybrid algorithm switches to constrained search with EPSO, initializing the swarm in the solution space near POCS solution. By leaving the swarm in a local near the optimal region, EPSO will rapidly converge towards the optimum. As a result of the hybrid strategy, MSR needs less iterations to reach the same solution that the provided simply by constrained search with EPSO.

However, this hybrid strategy depends on the convexity and efficiency of the autoencoder since it uses POCS to make the first search steps.

Chapter 4

Parabola Examples

4.1 Training the autoencoder

In this example was used the function $y(x) = x^2$ to evaluate the performance and restoration ability of the described methods. The next figure is shows the graphic of the parabola:

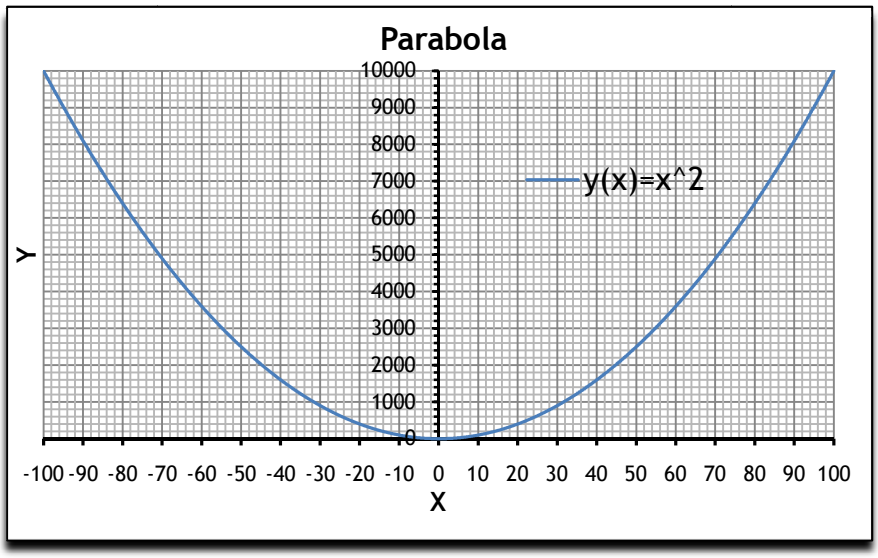

Figure 4.1 – Example: $y = x$^2.

To train the autoencoder that will approximate the parabola one has spitted such function into training and test data sets, as seen in the next figure. The train set is composed by one hundred values and the test set by thirty values, for each one of the forty autoencoder inputs. The autoencoder is trained without any missing input.

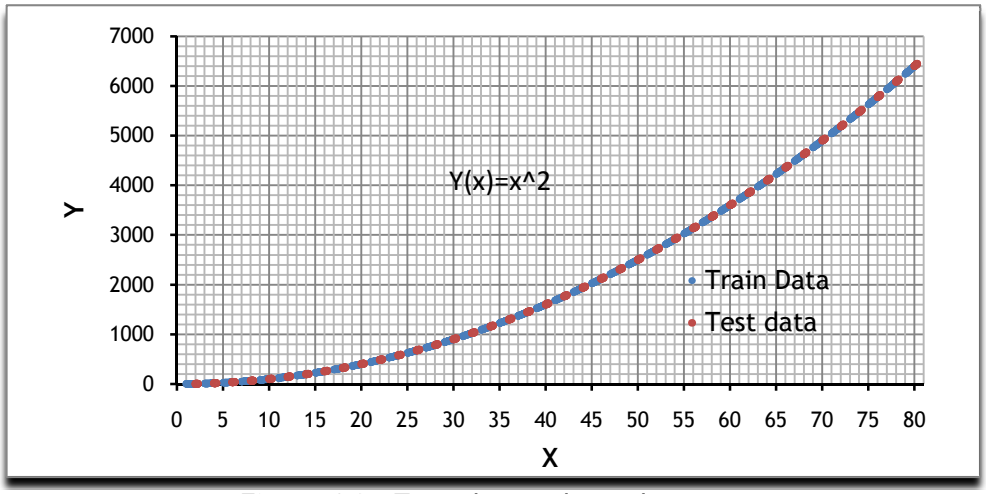

Figure 4.2 – Train data and test data sets

The sets of data obtained were them used to train an autoencoder with 40 neurons at the input layer, 20 in the hidden layer and, of course, 40 neurons at the output layer. The training algorithm used was the Resilient Backpropagation training algorithm and initial weights randomly initialized between]-1,4;1,4[.

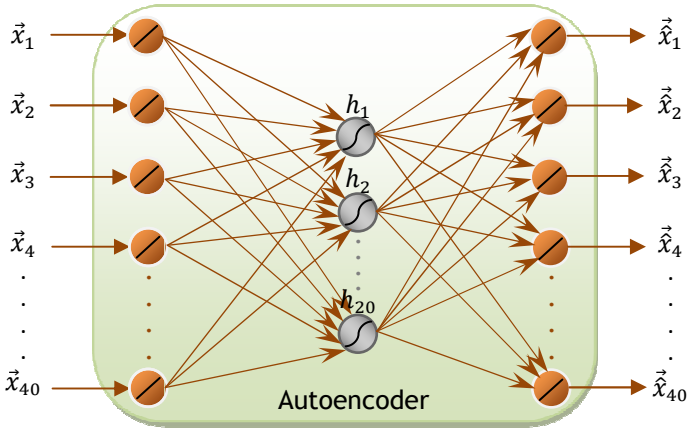

Figure 4.3 – Autoencoder structure used to approximate the parabola.

The hidden layer has a symmetric sigmoid activation function. The output layer has a linear activation function since a single non-linear activation function at the hidden layer has proven to be enough.

In the figure 4.4, below, is shown the evolution of the autoencoder training procedure:

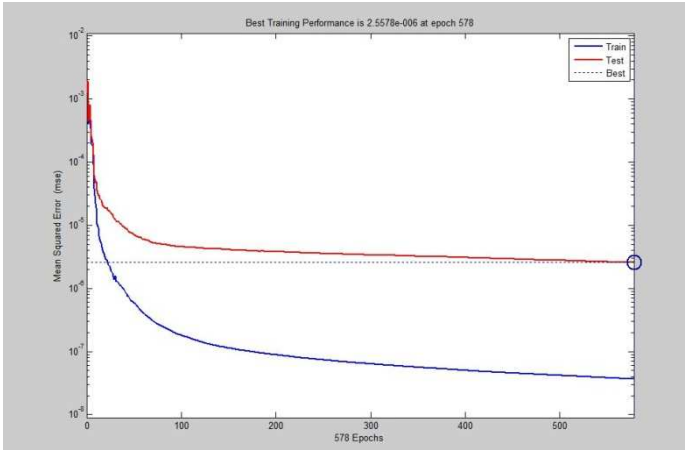

Figure 4.4 – Evolution of Autoencoder training.

4.2 MSR using POCS

In the next figure is shown the evolution of the mean squared error during the execution of POCS, for a missing sensor at input $\vec{x}_1$. The missing sensor was initially set to a value of zero. The data set is the test data (30 values) which correspond to untrained cases by the autoencoder and will determine is ability not only to generalize but also to recover sensor data in such scenarios.

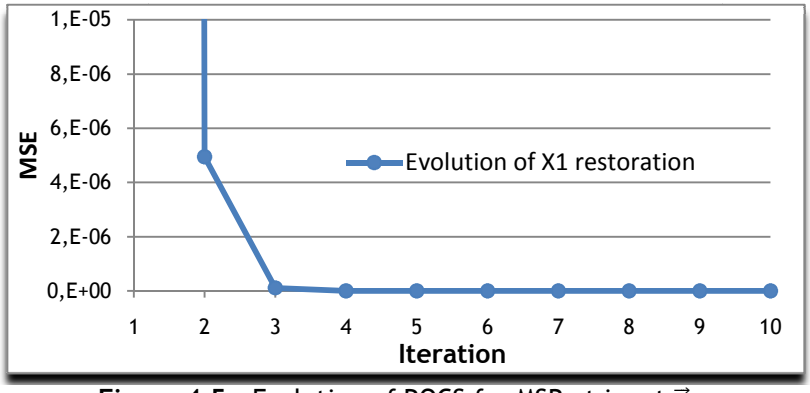

Figure 4.5 – Evolution of POCS for MSR at input $\vec{x}_1$.

In figure 4.5 we can see that, with the application of POCS algorithm, the autoencoder is able to recover the 30 test values missing at input $\vec{x}_1$. After 2 iterations, the restoration algorithm is nearly its final answer indicating a strong convergence. The mean squared error at the tenth iteration is $1{,}23 \times 10^{-9}$, which is a good value. The MSE is referring to standardized values ([-1,-1]). MSE on the original scale will depend of the range scale for that measurement.

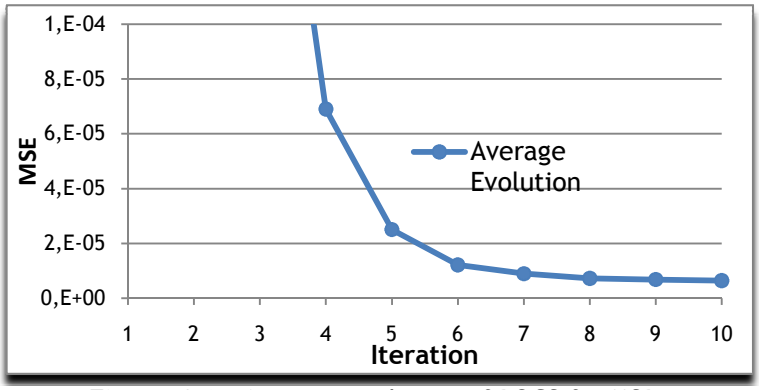

Figure 4.6 – Average evolution of POCS for MSR.

Figure 4.6 shows the average evolution of POCS algorithm for the recovery of inputs $\vec{x}_1$ to $\vec{x}_{40}$ sequentially missing one by one.

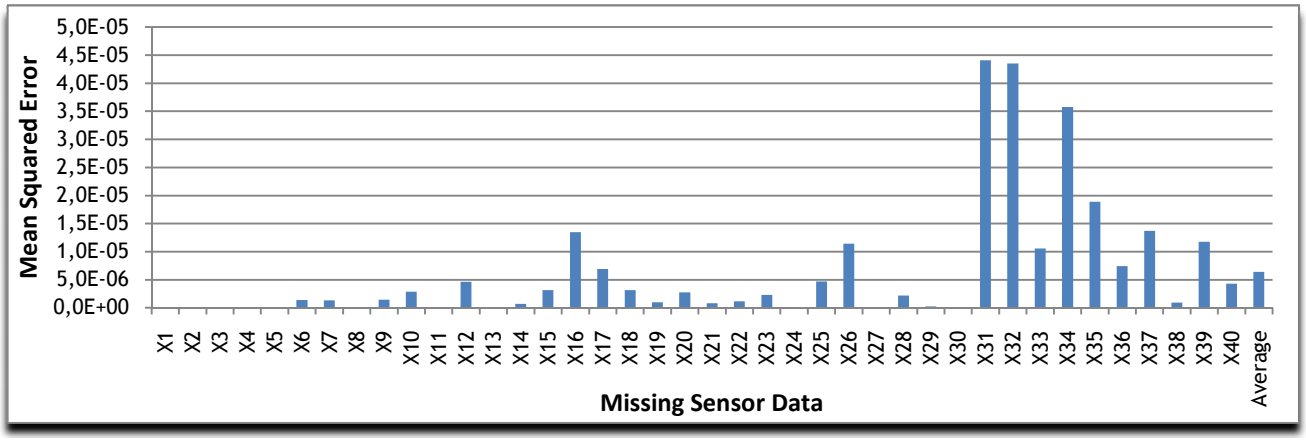

Figure 4.7 – MSE for MSR after 10 iterations with POCS.

In figure 4.7 is shown the mean squared error for the restoration of inputs 1 to 40 ($\vec{x}_1$ to $\vec{x}_{40}$). The error is small and therefore the autoencoder as able to properly approximate the parabola.

Let's see a sample for a single input value restoration using POCS:

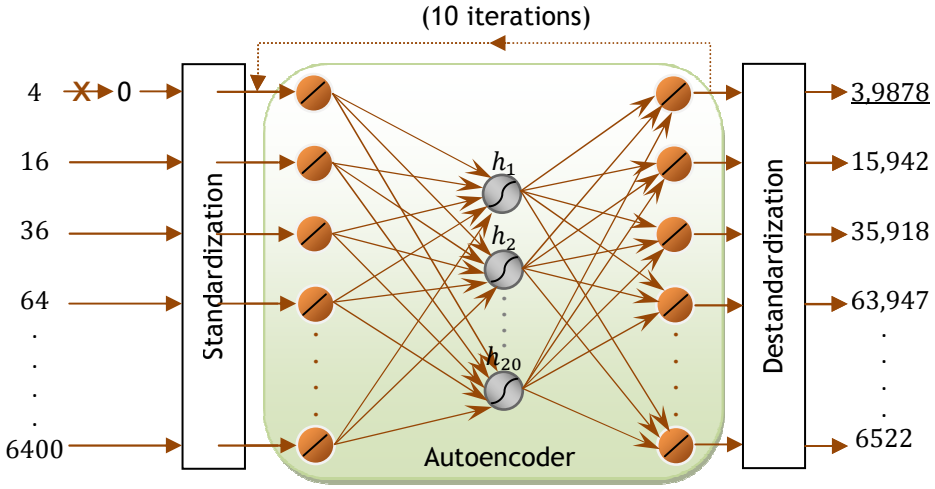

Figure 4.8 – Demonstration of a specific value restoration using POCS.

4.3 Determination of EPSO parameters

4.3.1 Determination of the Communication probability

The communication probability “P” is a parameter externally fixed, within a range of [0,1], that controls the passage of information within the warm. One way to introduce stochastic star topology is through a communication probability “p” among particles that are present in the main EPSO equation (A.7). The communication probability is represented by a diagonal matrix that has binary variables: a value of 1 with a probability p and a value of 0 with a probability (1-p). In the classical star communication topology the “p” value is equal to 1.

Therefore, for each iteration and dimension, there is a probability “P” that a particle will not receive the information available of the best location found by the swarm.

This is a very important parameter because, as we can see, it affects directly the evolution of a particle. For example if “P” assumes a very low value, the particle evolves almost only under the influence of inertia and memory components or, in other words, the “descendent” particle is built almost only of contributions from its ancestor line.

The determination of the best communication probability for a given problem is of the most importance because, if a full star communication is adopted, p=1, it may lead to premature convergence. On the other hand, if a low communication probability between particles is adopted, there is a risk to convert the particle swarm search into something close to a set of parallel individual searches. As already said, the adequate communication probability for a given problem results on more local search by each particle, allows the elimination of disturbing noise, allows the dynamics of particle movement to be more stable.

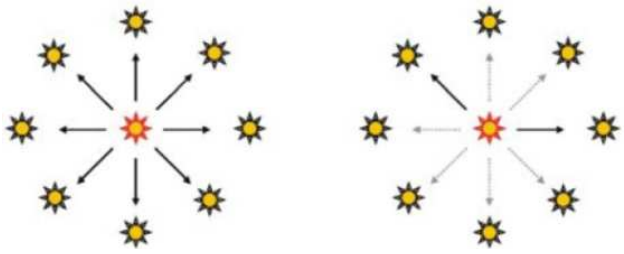

Figure 4.9 – Illustration of the star communication topology (a) and the stochastic star topology (b) [14].

To determine an adequate value to the communication probability “P”, experiments or tests were conducted on the unconstrained and constrained search for the parabola examples by repeating EPSO algorithm for the 30 test data for each value of p and varying only the communication probability value. All other things remaining unchanged, in all cases the population was set to 20 particles, the stop criteria was a fixed number of iterations (25), all weights were randomly initialized in the interval]0,1] and the learning rate has in all cases been set to 0.1.

The fitness function for unconstrained search is the sum of the absolute values of autoencoder input-output differences for the missing inputs. For the constrained search, the fitness function is the sum of the absolute values of autoencoder input-output differences for all input pattern (both known and missing inputs). In all the methods, EPSO seeks to minimize the referred difference, thus this is a minimization problem.

The results of the communication probability tests are presented on figures 4.6 and 4.7, where lower values are better:

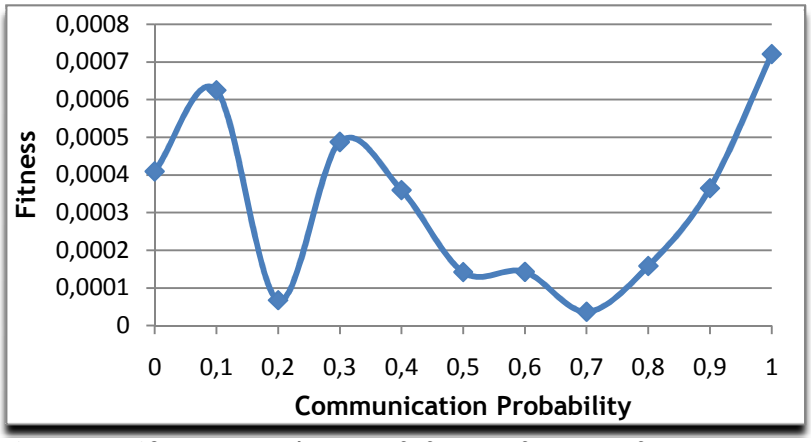

Figure 4.10 - Average (10 runs) evolution of fitness function for unconstrained search (restoration of the 30 test values) at input $\vec{x}_{13}$ as a function of the communication probability.

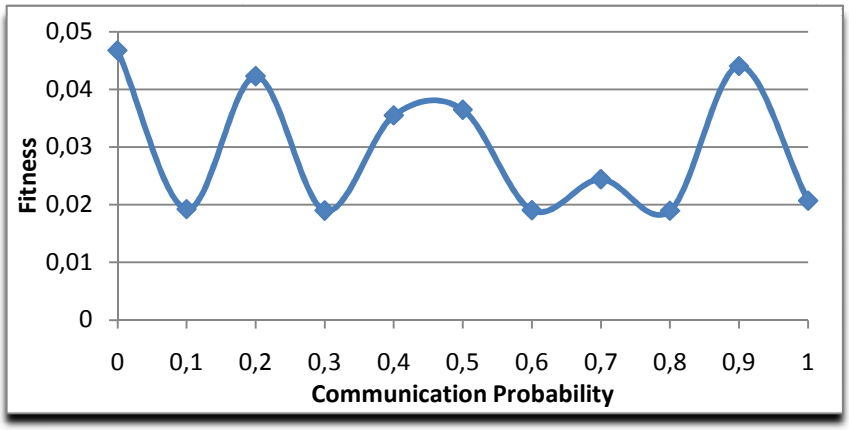

Figure 4.11 - Average (10 runs) evolution of fitness function for constrained search (restoration of the 30 test values) at input $\vec{x}_{13}$ as a function of the communication probability.

Observing the graph in Figure 4.10, it's clear that the best communication probability for unconstrained search is 0.7. For constrained search, some of the communication probabilities came with similar results but the best seems to be 0.6.

4.3.2 Determination of the learning parameter "τ"

Just like the cooperation probability parameter, the learning parameter "τ" is also externally fixed. The purpose of this parameter, on EPSO, is to mutate the weights of each particle, thus creating a new position for it in the search space by changing how the terms

inertia, memory and cooperation evolve when applied to the movement equations as described in Annex A.

The learning parameter is thus as important as the cooperation parameter, since it affects the particle's evolution by mutating its internal parameters in each iteration. This needs to be optimized too, so the algorithm can progress as efficiently as possible, avoiding premature convergence and also getting "stuck" on local optimums. This parameter is also what sets the EPSO apart from the PSO, because it introduces evolution programming/evolution strategies (EP/ES) resulting in competition between the parents and the descendent through an elitist tournament. The higher the learning parameter, the most different is the descendent compared to the parent, but that doesn´t necessarily mean that it is a good thing. That's why it is important to find the optimal value of the learning parameter. To that end, several tests were done with different learning parameters that are represented in figure 4.12. For these tests, the communication probability was set to 0.7 for unconstrained search and 0.6 for constrained search. The remaining parameters are the same as the previous tests.

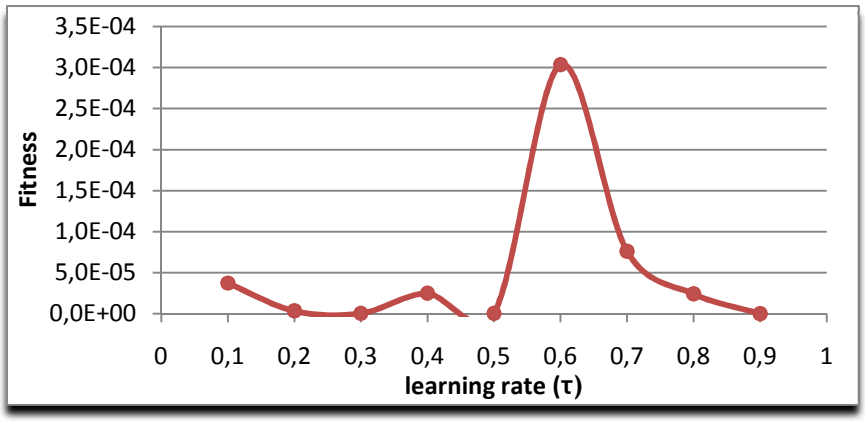

Figure 4.12 – Average (10 runs) of fitness function for unconstrained search (restoration of the 30 values) at input $\vec{x}_{13}$ as a function of the learning rate.

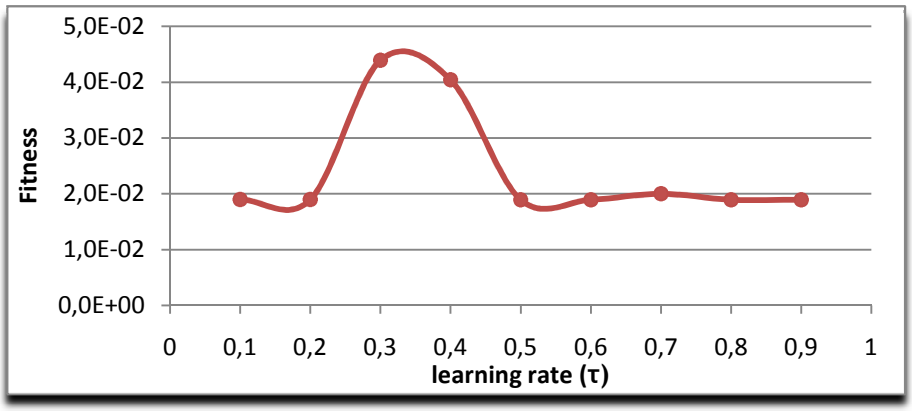

Figure 4.13 – Average (10 runs) of fitness function for constrained search (restoration of the 30 values) at input $\vec{x}_{13}$ as a function of the learning rate.

For unconstrained search the learning parameter was set to 0.5. In the constrained search the learning parameter was set to 0.2.

4.4 MSR using Unconstrained Search with EPSO

Figure 4.9 shows the evolution of the mean squared error during the execution of Unconstrained Search with EPSO, for a missing sensor at input $\vec{x}_1$. The missing sensor is randomly initiated by EPSO. The data set is the test data (30 values).

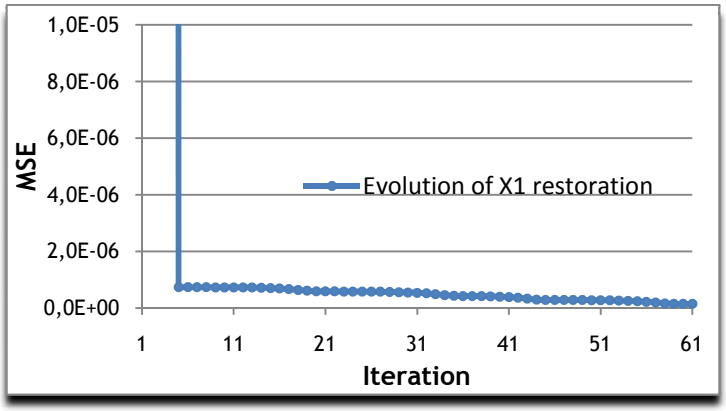

Figure 4.14 – Evolution of Unconstrained Search with EPSO for MSR at input $\vec{x}_1$.

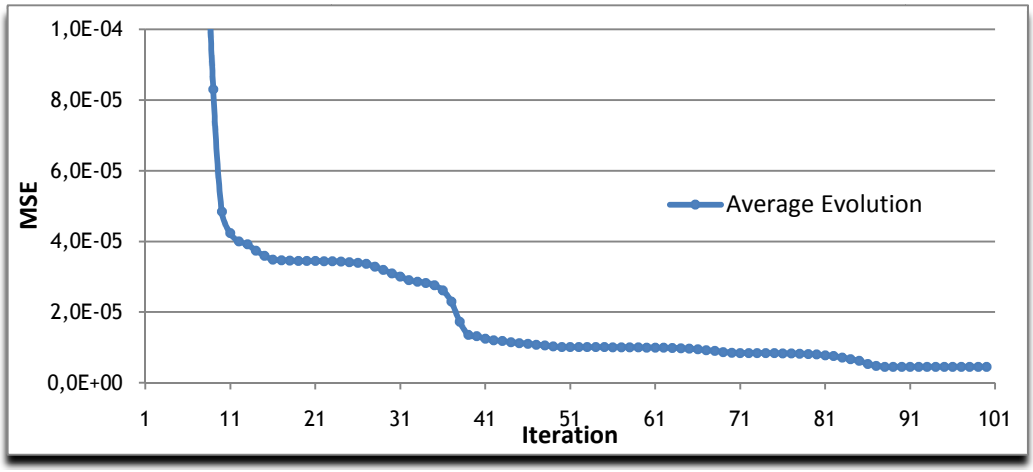

Figure 4.15 – Evolution of Unconstrained Search with EPSO for MSR.

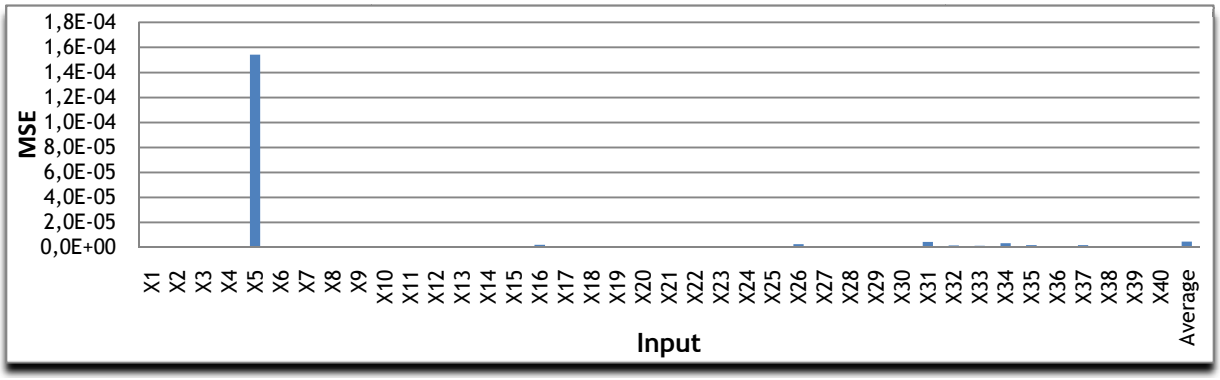

Figure 4.16 – MSE for MSR after 100 iterations using Unconstrained Search with EPSO.

4.5 MSR using Constrained Search with EPSO

In the next figure is shown the evolution of the mean squared error during the execution of Constrained Search with EPSO, for a missing sensor at input $\vec{x}_1$. The missing sensor value is randomly initiated by EPSO. The data set is the test data (30 values) which correspond to untrained cases by the autoencoder and will determine is ability not only to generalize but also to recover sensor data in such scenarios.

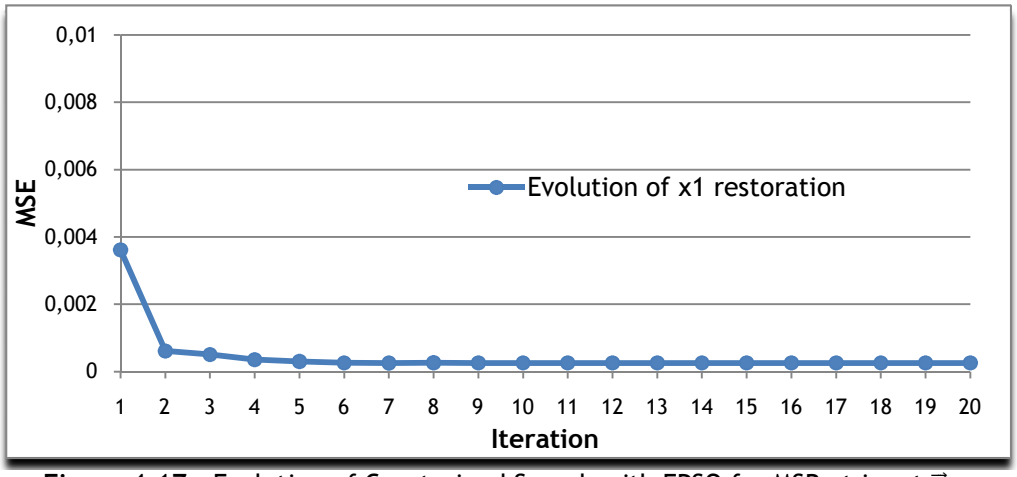

Figure 4.17 – Evolution of Constrained Search with EPSO for MSR at input $\vec{x}_1$.

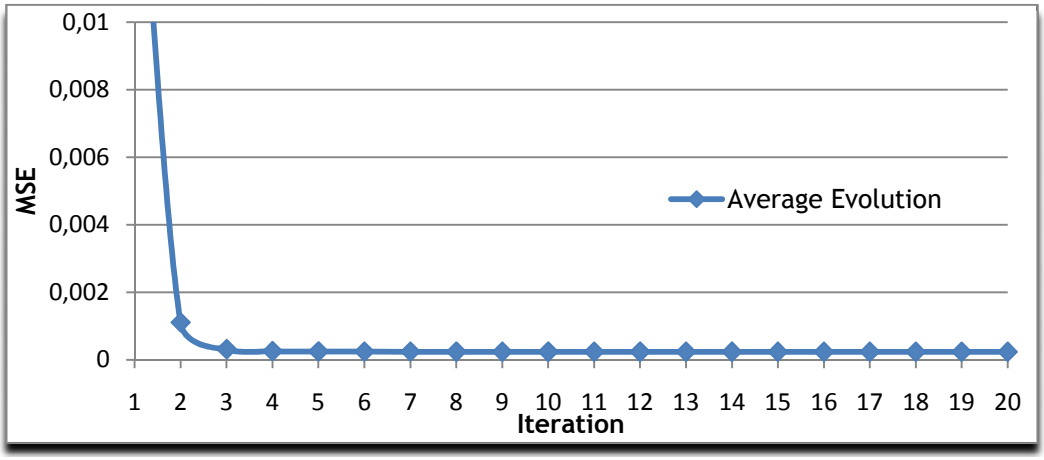

Figure 4.18 – Evolution of Constrained Search with EPSO for MSR one by one.

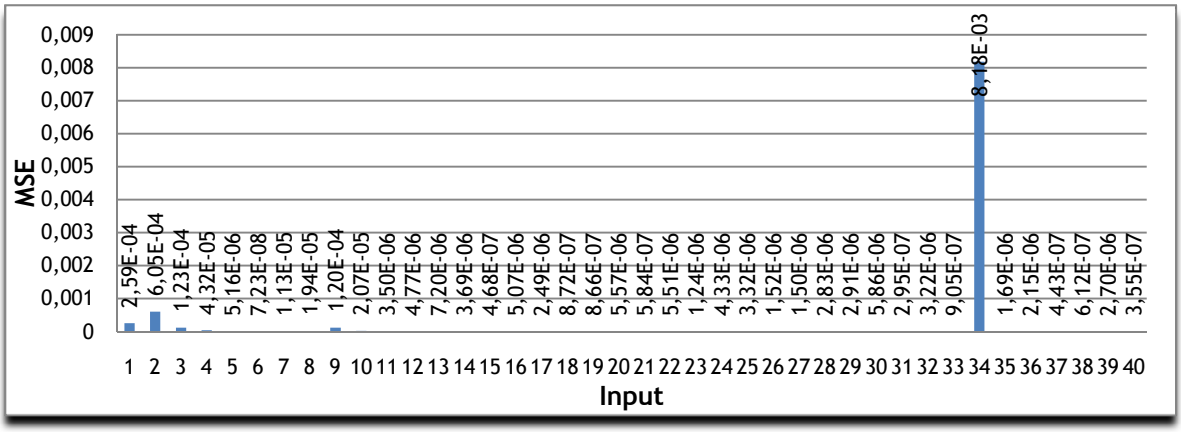

Figure 4.19 – MSE for MSR after 20 iterations using Constrained Search with EPSO.

4.6 MSR using a Hybrid method: POCS+Constrained Search with EPSO

In the next figure is shown the evolution of the mean squared error during the execution of Hybrid method, for a missing sensor at input $\vec{x}_1$. The missing sensor was initially set to a value of zero. The data set is the test data (30 values) which correspond to untrained cases by the autoencoder and will determine is ability not only to generalize but also to recover sensor data in such scenarios.

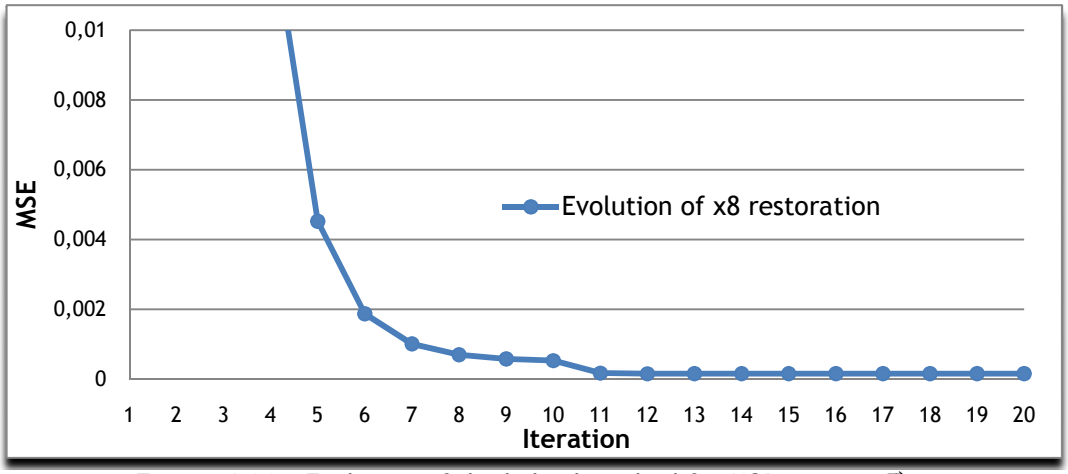

Figure 4.20 – Evolution of the hybrid method for MSR at input $\vec{x}_1$.

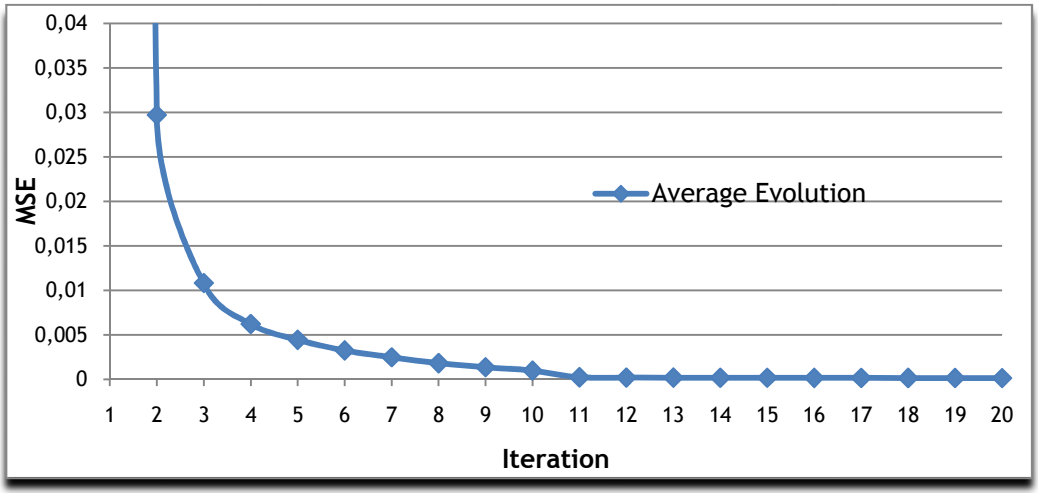

Figure 4.21 – Evolution of the hybrid method for MSR one by one.

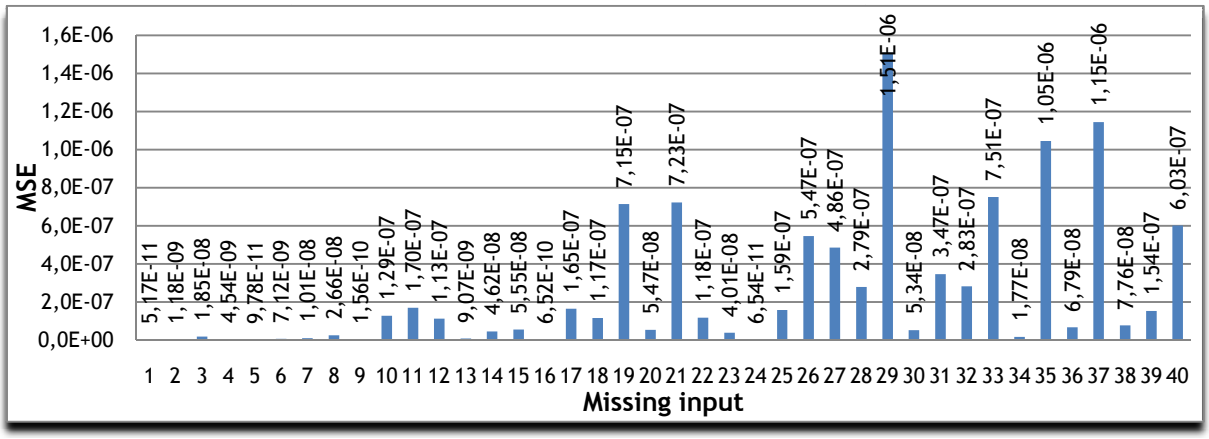

Figure 4.22 – MSE for MSR after 20 iterations with hybrid strategy.

4.7 Multiple Missing Sensor Restoration

Figure 4.23 shows the evolution of Mean Squared Error in function of the number of missing measurements at autoencoder inputs for the various restoration algorithms.

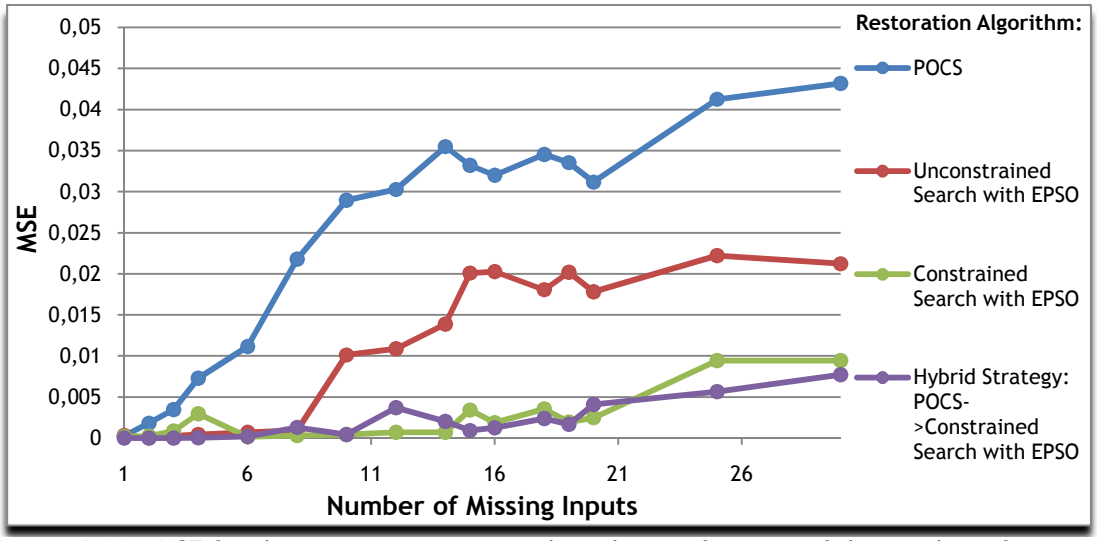

Figure 4.23 – MSE for the various restoration algorithms in function of the number of missing inputs.

4.8 Comparison of the Recovery algorithms

The computational effort mentioned in these experiments was calculated as dictated in equation 4.1.

$$\boldsymbol{Computational\ effort} = 2 \times \boldsymbol{Swarm}_{\boldsymbol{size}} \times \boldsymbol{Iterations}_{\boldsymbol{number}} \tag{4.1}$$

The computational effort value is in fact an important comparison index since that this way one is able to obtain the number of calls to the evaluation procedure (the most time consuming procedure in all the optimization process) accordingly to swarm size and to the number of iterations needed to achieve the optimum solution or any other desirable solution, providing some insight relativity to the required computational capacity in order to solve the problem.

This “index” will be often used to make performance comparisons between different restoration algorithms further in thesis, allowing one to have some sort of comparison between developed strategies.

For unconstrained search with EPSO was chosen a swarm with 5 particles. However, in Constrained search with EPSO one chosen a swarm with 10 particles.

POCS simulates the autoencoder onetime per iteration so one defined his computer effort equals to its number of iterations.

The hybrid strategy starts by using POCS (computational effort = number of iterations) and then switches to constrained search with EPSO using a swarm with 10 particles.

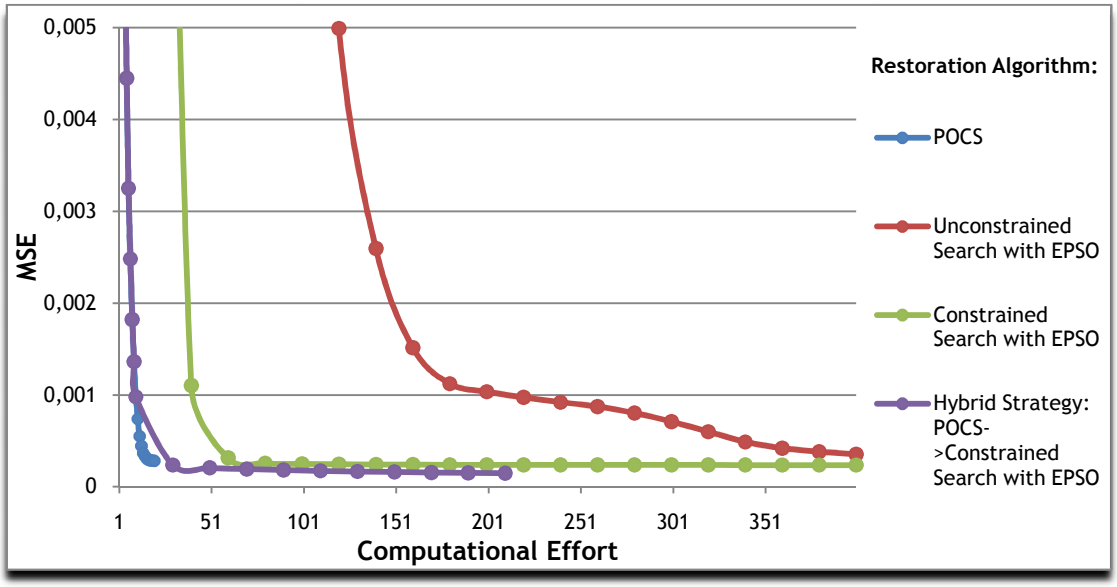

Figure 4.24 - MSE for the various restoration algorithms in function of the computational effort.

Table 4.1 - Comparison of the recovery performance for 10 missing inputs (destandardized values).

		Restoration algorithm							
		POCS		Unconstrained Search with EPSO		Constrained Search with EPSO		Hybrid Strategy: POCS->Constrained Search with EPSO	
Missing Inputs		Recovery value	MAE	Recovery value	MAE	Recovery value	MAE	Recovery value	MAE
X1	4	1,26	2,74	4,88	0,88	4,01	0,01	4,11	0,11
X2	16	22,59	6,59	15,69	0,31	16,13	0,13	16,20	0,20
X3	36	52,43	16,43	32,68	3,32	35,83	0,17	35,66	0,34
X4	64	68,52	4,52	68,35	4,35	63,36	0,64	63,60	0,40
X5	100	86,44	13,56	105,22	5,22	99,23	0,77	100,03	0,03
X6	144	115,70	28,30	148,55	4,55	141,83	2,17	141,30	2,70
X7	196	218,53	22,53	203,73	7,73	195,21	0,79	195,32	0,68
X8	256	293,48	37,48	238,18	17,82	258,09	2,09	258,67	2,67
X9	324	262,29	61,71	334,49	10,49	323,07	0,93	322,41	1,59
X10	400	322,20	77,80	401,90	1,90	397,11	2,89	395,50	4,50
		Total	271,67	-	56,57	-	10,61	-	13,23

4.9 Chapter conclusions

In conclusion, it's worth mentioning that the autoencoder has successfully approximated the parabola and, in combination with restoration algorithms, recovered the values missing at its inputs. Since the parabola example is a simple application and the correlation between the autoencoder inputs is strong, the difficulty of the recovery scheme was not too great.

However, its simplicity was enough to test different restoration algorithms and autoencoders and that way proving the concept.

It was demonstrated in section 4.2 how the missing sensor restoration with POCS behaves in the presence of a single missing input. This analysis was made for all of the forty inputs. The recovery algorithm with POCS has the ability of restoring any missing input, in other words, the algorithm doesn't depend exclusively on an input or certain inputs to get to restore missing inputs.

As for unconstrained search with EPSO, it was revealed convergent for all situations, even when several inputs are missing. However, when we have few missing inputs the convergence speed wasn´t the enough to overcome the method POCS that requests an inferior computational effort. But figure 4.23 shows that when the number of missing inputs increases, the algorithm POCS has a reduced performance, which results in slower convergence or even not converging to the missing inputs values. In these situations, unconstrained search with EPSO has better performance.

As expected, constrained search with EPSO is the recovery method that has revealed the best results (lower MSE) for both situations, when are few or several missing inputs. From this one concluded that the information contained in the healthy measurements is important in the determination of the missing ones.

Finally, the proposed hybrid strategy gives the predicted results. It starts by using the POCS method that has low computational effort and then switches to constrained search with EPSO. The results are similar to a full constrained search with EPSO with the exception of the computational effort which is lower. For some cases, when are several inputs missing, the hybrid strategy gives even better result than constrained search with EPSO. I think this occur because the initial iterations with POCS helps EPSO avoiding local minimums.

The next chapter will be used to test these restoration techniques on a 24-bus AC power system network which is more complex, thus requiring a more sophisticated autoencoder and possibly more iterations on the restoration algorithms to converge to an optimal solution.

Chapter 5

Applying Missing Sensor Restoration with Autoencoders to Power Systems

5.1 Introduction

A characteristic of the transmission power systems is the high number of control and automation equipments. This was due to the fact that, in the past, power companies were vertically organized from generation, transmission, distribution to commercialization. Additionally, the electric utilities invested mostly in the modernization of generation and transmission areas leading to high levels of automation and remote control in these subsystems. This allows the transmission system operator to have a real-time monitoring of the entire network through computerized systems called energy management systems (EMS). On the other hand, the distribution systems that are usually operated in a radial configuration, up until recently (last 10-20 years) had very poor investment, resulting in almost no control, nor automation. This resulted in a very low amount of sensors, thus it being impractical for the traditional real-time monitoring (like EMS) system to be used.

Therefore, the "traditional SCADA – System Control and Data Acquisition are being upgraded to more powerful systems that, to a certain extent, adopt some characteristics present in EMS to distribution networks", called distribution management systems (DMS).

In energy management systems (EMS), there are two types of measurement data collected by the SCADA system, namely, status data of breakers and switches, and analog data of real and reactive power flows, injections, and bus voltages. The status data are used to determine real-time topology of the network. The analog data are used to determine line and transformer loading and voltage profile. These data are noisy due to measurement errors, communication noise, missing data, etc. Errors in telemetered data of breaker and switch status, though the network topology processes in the EMS computer, may result in errors in the determination of the current network topology of the system [21].

5.2 The IEEE 24 Bus RTS

The topology of the power system used to evaluation studies is shown in Figure 5.1. This power system network corresponds to one area of "The IEEE Reliability Test System – 1996" [22].In as been added two switching devices S_1 and S_2 that control the status of lines $L_{1-2}$and L_{11-13}, respectively.

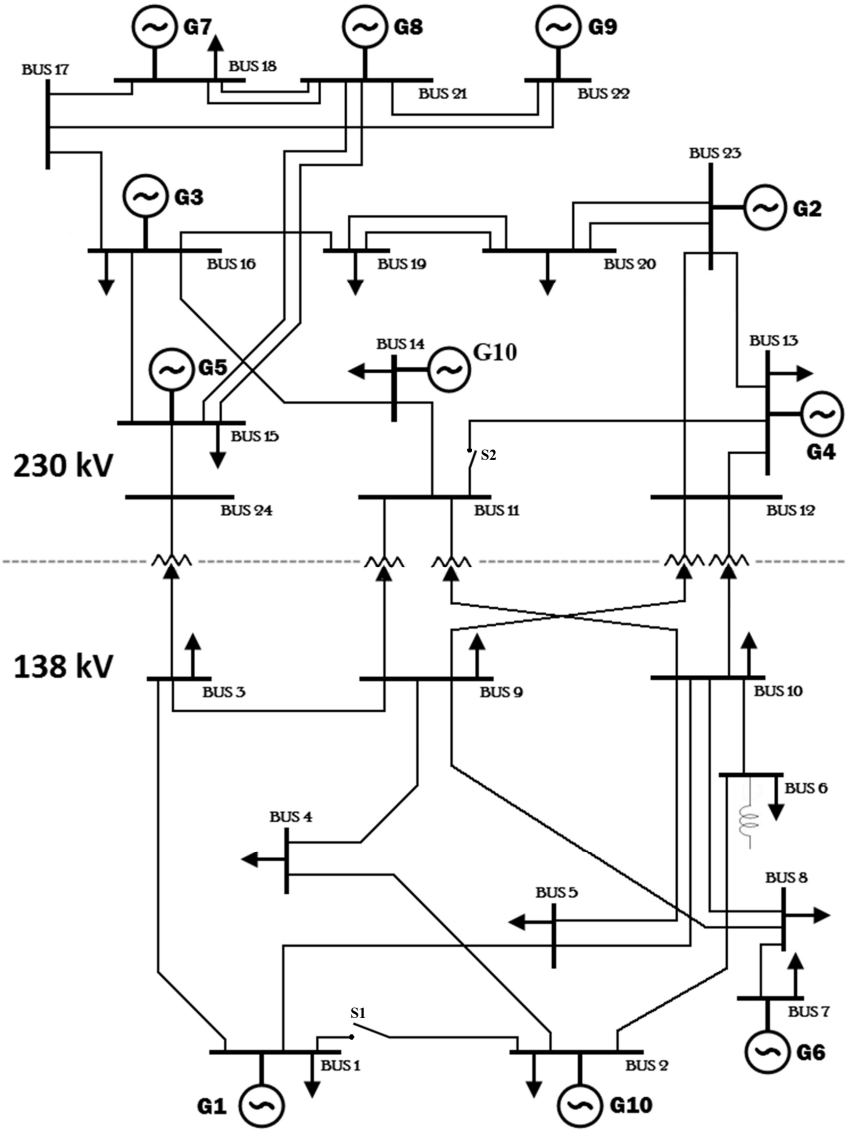

Figure 5.1 - IEEE One Area RTS-96 with some modifications.

In order to simulate different scenarios to train the autoencoder, all active and reactive loads were randomly set using a load profile between 56% and 100% of P_{LOAD}, every power flow.

The power flows were solved in MatPower 3.2 [23] with AC Optimal Power Flow (OPF).

On the next two experiences the switching devices weren't considered, which means that the transmission lines are always available.

5.3 Restoring Voltage Magnitude and Angle

With the purpose of properly training the autoencoder one as realized 10000 Newton-Raphson power flows. These power flows were divided 8500 train data sets, 1000 test data sets and 500 validation data sets. Analyzing the data collected from the power flows, one has observed that there is a strong correlation existent between the voltage amplitudes/angles at the different buses across the power system.

The next step consists in the definition of the autoencoder parameters. Since the power system considered as 24 buses, the autoencoder will have 48 inputs corresponding to the voltage magnitude and voltage angle at each bus.

After several experiments, the best autoencoder structure that one found is present in figure 5.2, below.

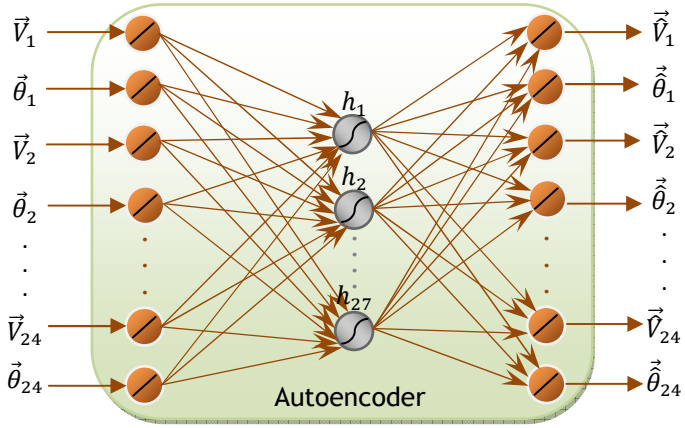

Figure 5.2 - Autoencoder structure used to restore Voltages and Angles.

The autoencoder is composed by one hidden layer with 27 neurons. These neurons have a symmetric sigmoid activation function. The output layer has a linear activation function.

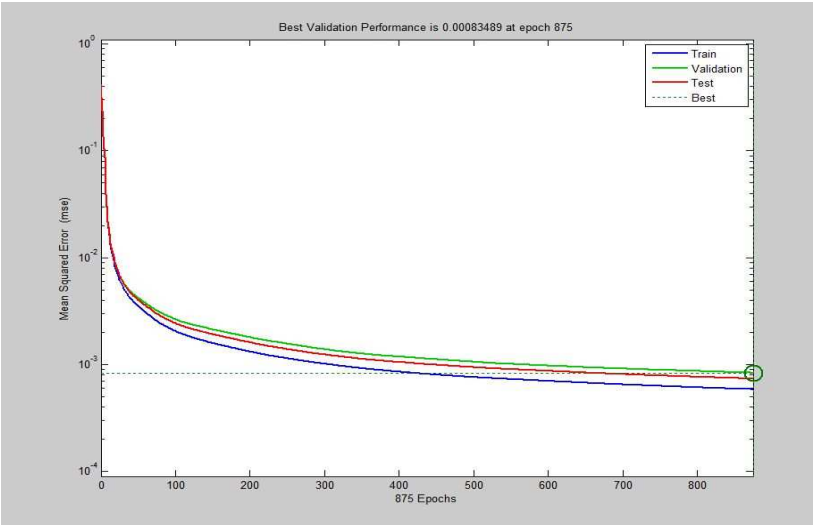

Figure 5.3 - Evolution of Autoencoder training.

Observing the figure 5.3, we can see that the autoencoder training procedure was very smooth. The training procedure was stopped at epoch 875 since the performance evolution became slow.

5.3.1 MSR using POCS

Figure 5.4 represents the evolution for restoration of the voltage magnitude value at bus 1.

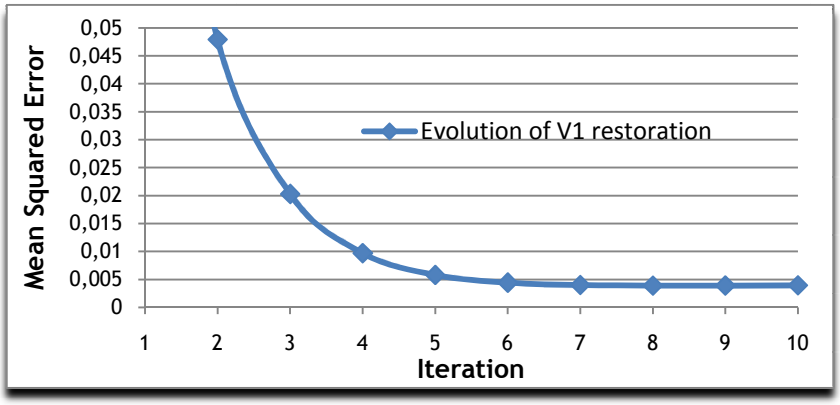

Figure 5.4 - Evolution of POCS for MSR at input $\vec{V}_1$.

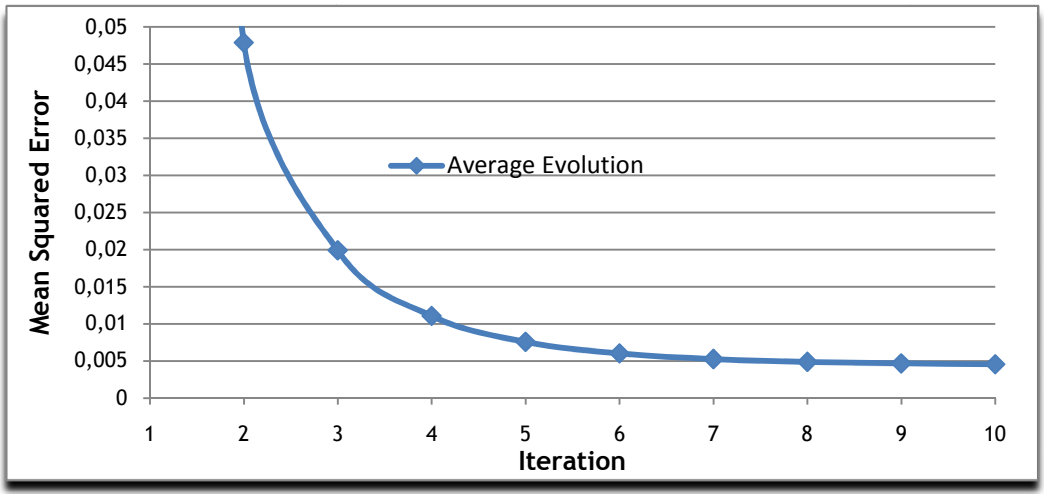

Figure 5.5 - Evolution of MSR with POCS one by one.

In figure 5.5, it's noticeable that the average evolution of the recovery process tends to the same MSE of V1 restoration. As we can see in figures 5.6 and 5.7 (beneath), the restoration algorithm was able to recover all the measurements with an acceptable error. The swing bus is bus 13, by definition, the voltage angle in this bus is always 0º. Then it's obvious that the error in recovery of $\vec{\theta}_{13}$ should be 0. In average, the errors are lower in restoration of voltage angle measurements.

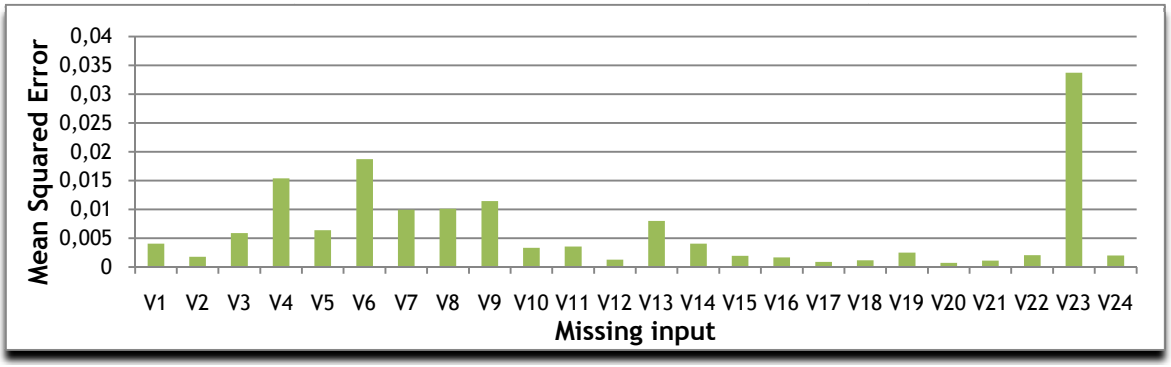

Figure 5.6 - MSE for Restoration of Voltages after 10 iterations with POCS.

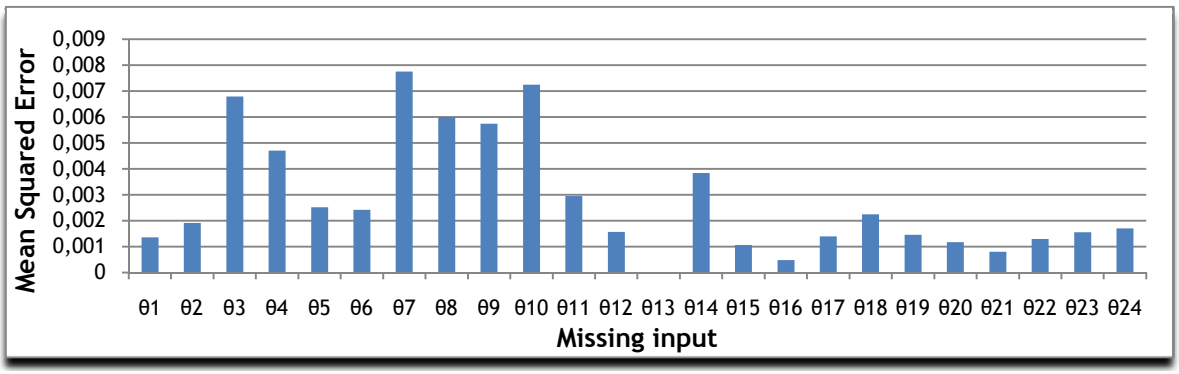

Figure 5.7 - MSE for Restoration of Angles after 10 iterations with POCS.

5.3.2 MSR using Unconstrained Search with EPSO

Figure 5.8 indicates the evolution for restoration of the voltage magnitude value at bus 1 using unconstrained search with EPSO.

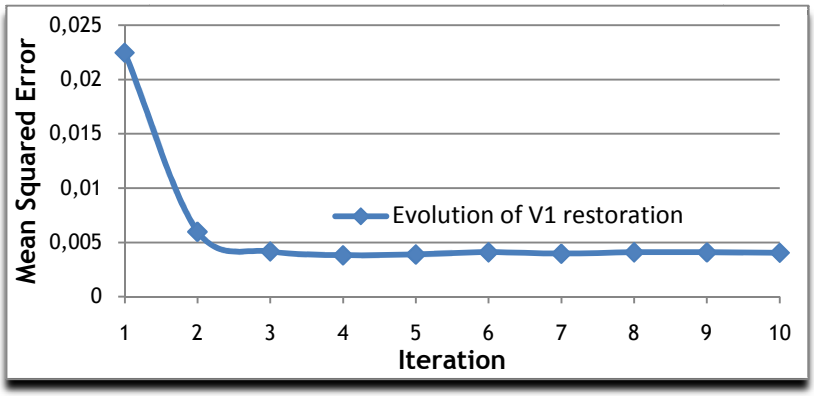

Figure 5.8 - Evolution of unconstrained search with EPSO for MSR at input $\vec{V}_1$.

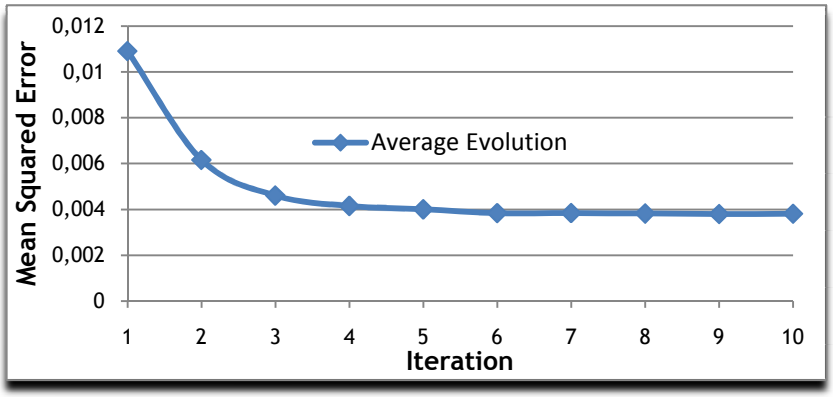

Figure 5.9 - Evolution of MSR using unconstrained search with EPSO one by one.

As it's observed in the graphs, the results for the errors are similar or superiors to the ones obtained using POCS. This is due to the fact that if both POCS and unconstrained search with EPSO converge then the results obtained for these two methods will be the same.

Figure 5.10 - MSE for Restoration of Voltages after 10 iterations using unconstrained search with EPSO.

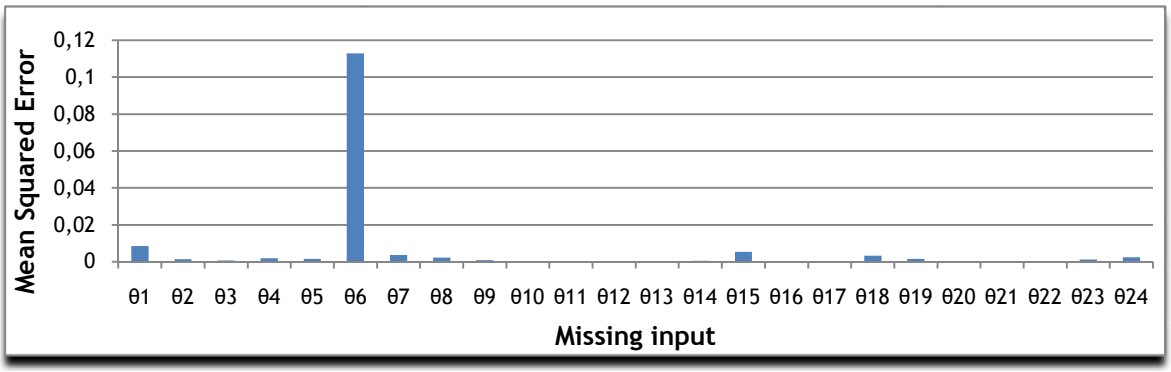

Figure 5.11 - MSE for Restoration of Angles after 10 iterations using unconstrained search with EPSO.

5.3.3 MSR using Constrained Search with EPSO

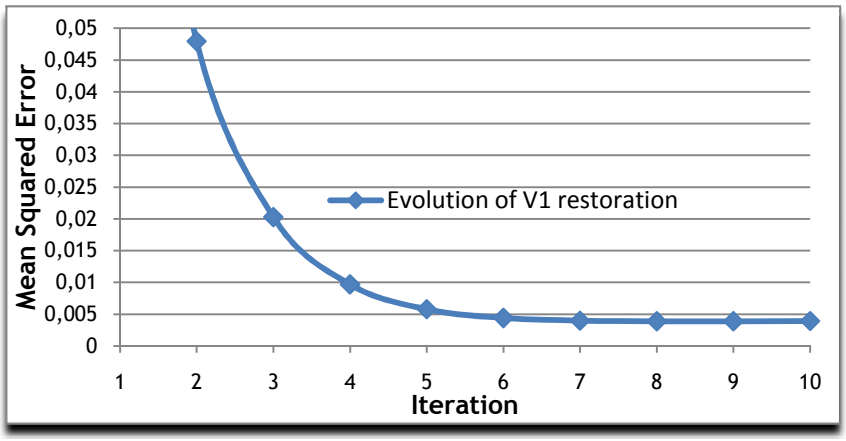

Figure 5.12 - Evolution of constrained search with EPSO for MSR at input $\vec{V}_1$.

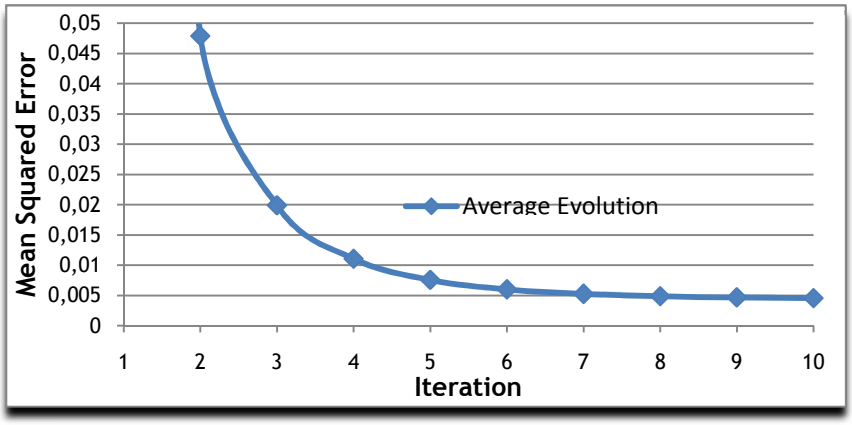

Figure 5.13 - Evolution of MSR using constrained search with EPSO one by one.

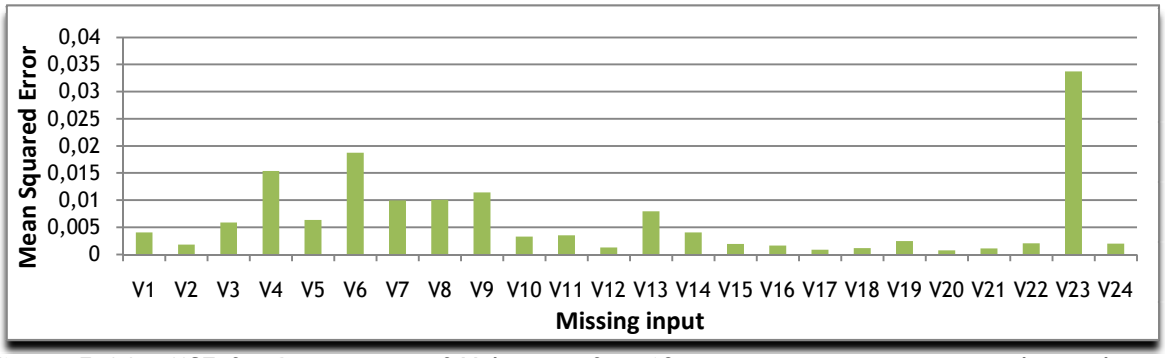

Figure 5.14 - MSE for Restoration of Voltages after 10 iterations using constrained search with EPSO.

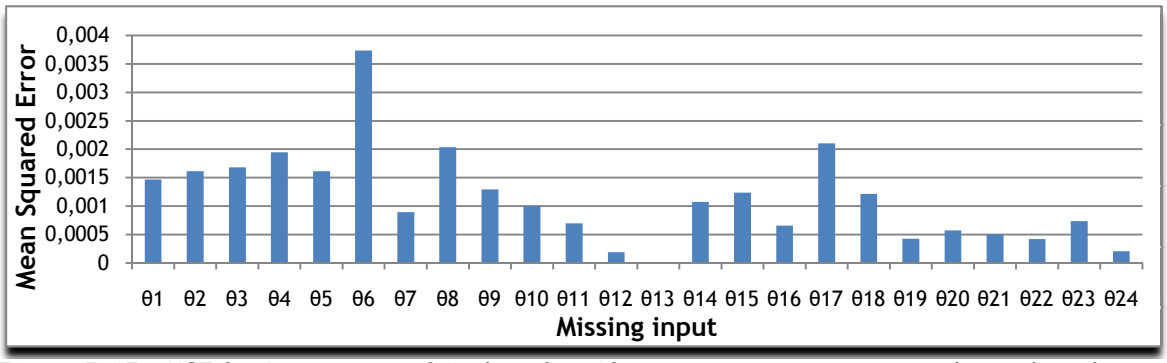

Figure 5.15 - MSE for Restoration of Angles after 10 iterations using constrained search with EPSO.

4.10 MSR using a Hybrid method: POCS+Constrained Search with EPSO

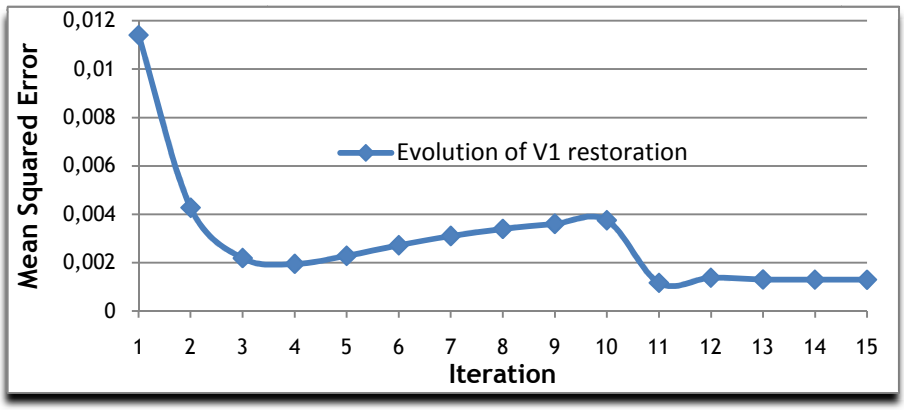

Figure 5.16 - Evolution of hybrid method for MSR at input $\vec{V}_1$.

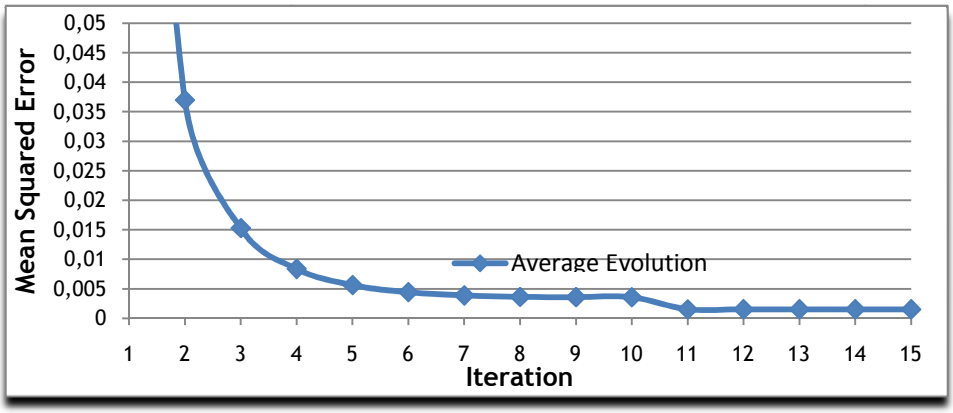

Figure 5.17 - Evolution of MSR using hybrid method one by one.

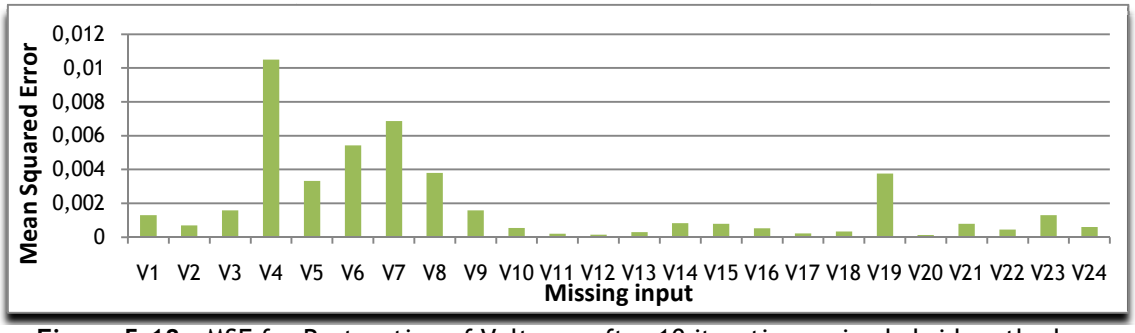

Figure 5.18 - MSE for Restoration of Voltages after 10 iterations using hybrid method.

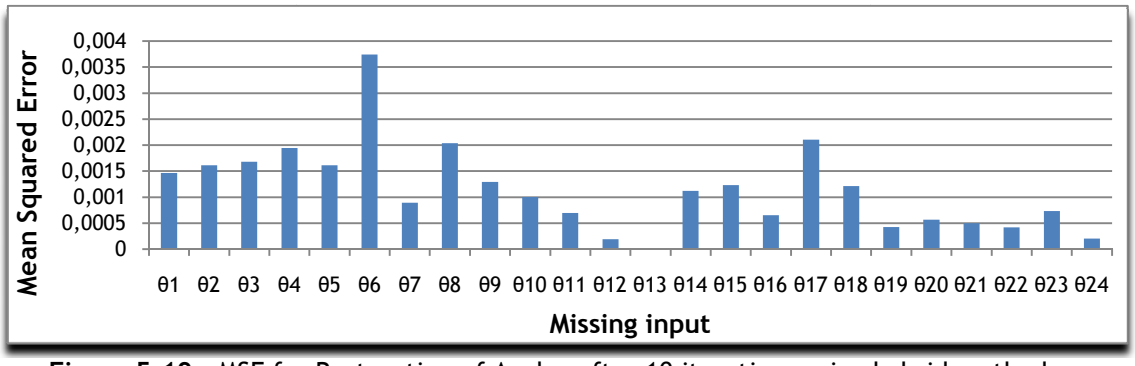

Figure 5.19 - MSE for Restoration of Angles after 10 iterations using hybrid method.

4.11 Multiple Missing Sensor Restoration

Figure 5.20 - MSE for the various restoration algorithms in function of the number of missing inputs.

5.3.1 Comparison of the Recovery algorithms

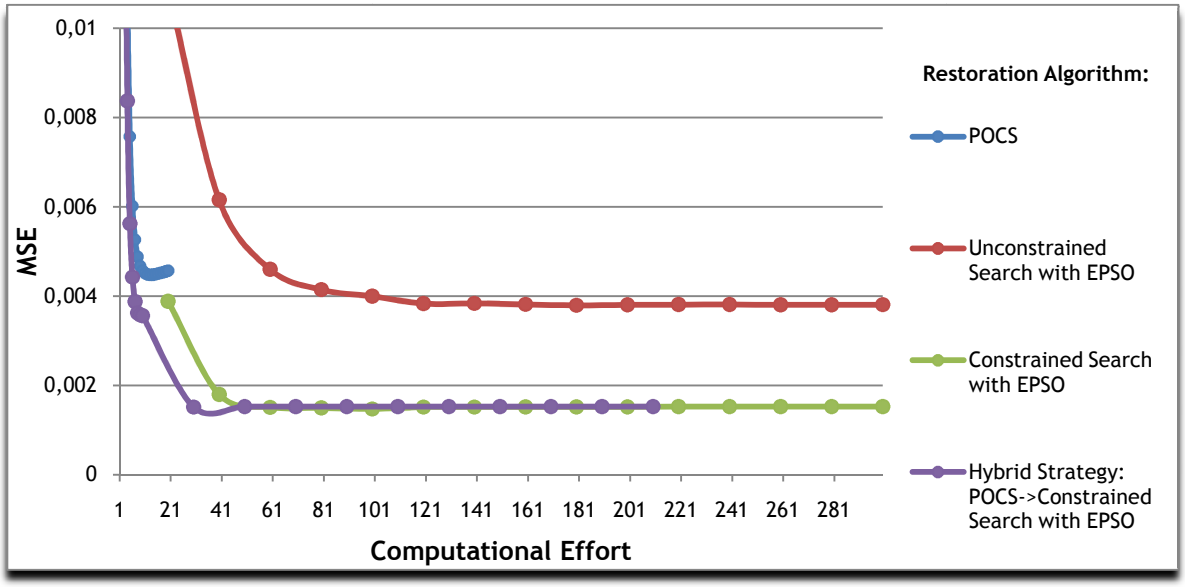

Figure 5.21 - MSE for the various restoration algorithms in function of the computational effort.

Table 5.1 - Comparison of the recovery performance for 10 missing inputs (destandardized values).

		Restoration algorithm							
Missing Inputs		POCS		Unconstrained Search with EPSO		Constrained Search with EPSO		Hybrid Strategy: POCS->Constrained Search with EPSO	
Name	Correct Value	Restored values	Error	Restored values	Error	Restored values	Error	Restored values	Error
V1(p.u.)	0,92884	0,915	0,01	0,942	0,01	0,937	0,01	0,9426	0,01
θ2(degree)	-6,21108	-7,548	1,34	-5,197	1,01	-5,612	0,60	-4,3474	1,86
V2(p.u.)	0,929696	0,923	0,01	0,942	0,01	0,937	0,01	0,9428	0,01
Θ2(degree)	-6,26523	-6,333	0,07	-4,791	1,47	-5,396	0,87	-4,3421	1,92
V3(p.u.)	0,945787	0,922	0,02	0,946	0,00	0,945	0,00	0,9453	0,00
Θ3(degree)	-8,76694	-12,234	3,47	-8,587	0,18	-8,629	0,14	-8,8644	0,10
V4(p.u.)	0,923358	0,938	0,01	0,939	0,02	0,929	0,01	0,9299	0,01
Θ4(degree)	-10,9824	-9,786	1,20	-9,429	1,55	-10,216	0,77	-9,7357	1,25
V5(p.u.)	0,941243	0,924	0,02	0,939	0,00	0,945	0,00	0,9482	0,01
Θ5(degree)	-10,7963	-13,467	2,67	-10,916	0,12	-10,452	0,34	-9,4429	1,35
		MAE	8,81		4,39		2,74		6,53

5.4 Restoring Injected Active and Reactive Power

The following experiment consists on the recovery of injected active and reactive power in all buses. The power system considered as 24 buses so the autoencoder will have 48 inputs corresponding to the injected active and reactive power at each bus.

After several experiments, the best autoencoder structure that one found is present in figure 5.22, below.

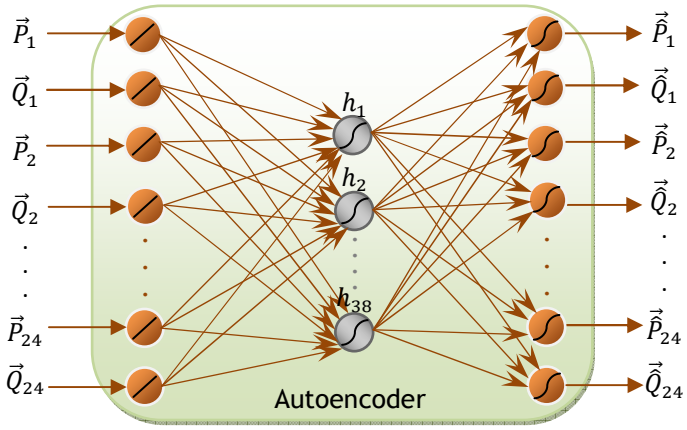

Figure 5.22 - Autoencoder structure used to restore injected Active and Reactive Power.

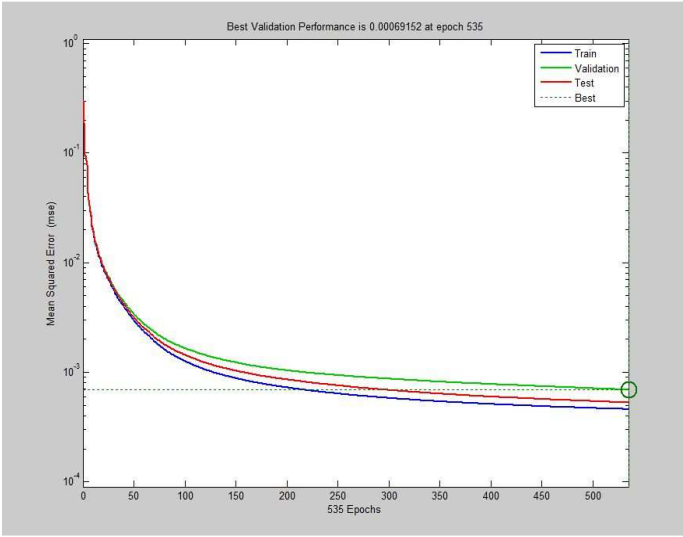

Figure 5.23 - Evolution of the training procedure for the autoencoder used to restore injected Active and Reactive Power.

5.4.1 MSR using POCS

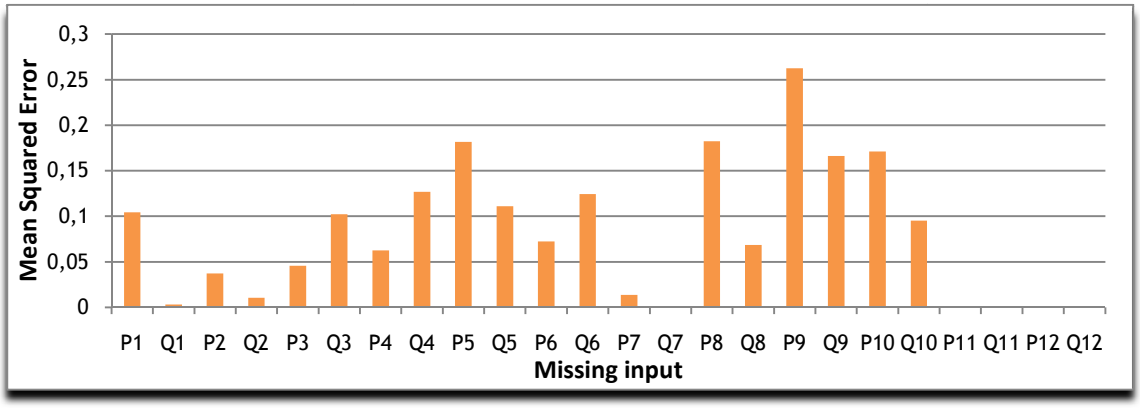

Figure 5.24 - MSE for Restoration of $\vec{P}_1$ to $\vec{P}_{12}$ and $\vec{Q}_1$ to $\vec{Q}_{12}$ after 20 iterations using POCS.

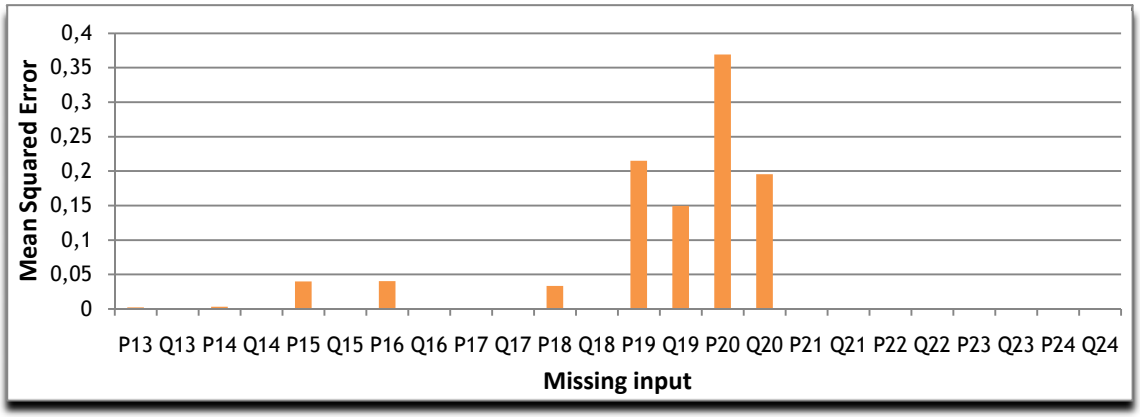

Figure 5.25 - MSE for Restoration of $\vec{P}_{13}$ to $\vec{P}_{24}$ and $\vec{Q}_{13}$ to $\vec{Q}_{24}$ after 20 iterations using POCS.

5.4.2 MSR using Unconstrained Search with EPSO

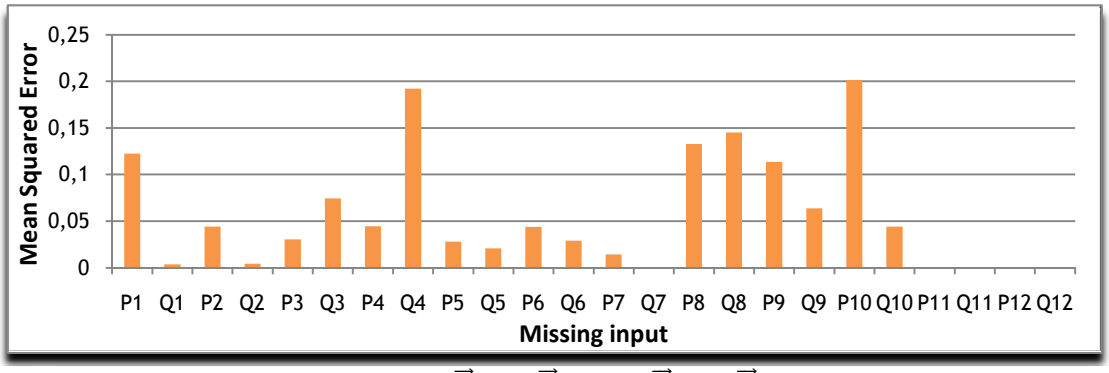

Figure 5.26 - MSE for Restoration of $\vec{P}_1$ to $\vec{P}_{12}$ and $\vec{Q}_1$ to $\vec{Q}_{12}$ after 20 iterations using Unconstrained Search with EPSO.

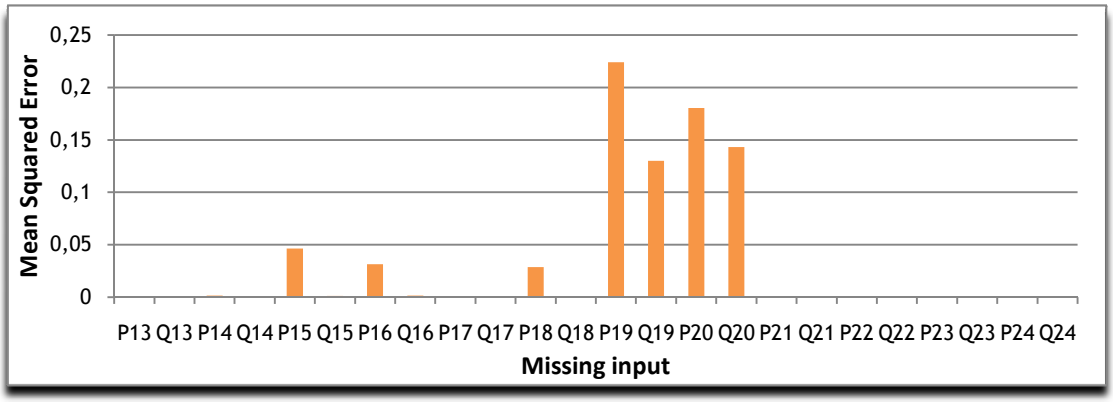

Figure 5.27- MSE for Restoration of $\vec{P}_{13}$ to $\vec{P}_{24}$ and $\vec{Q}_{13}$ to $\vec{Q}_{24}$ after 20 iterations using Unconstrained Search with EPSO.

5.4.3 MSR using Constrained Search with EPSO

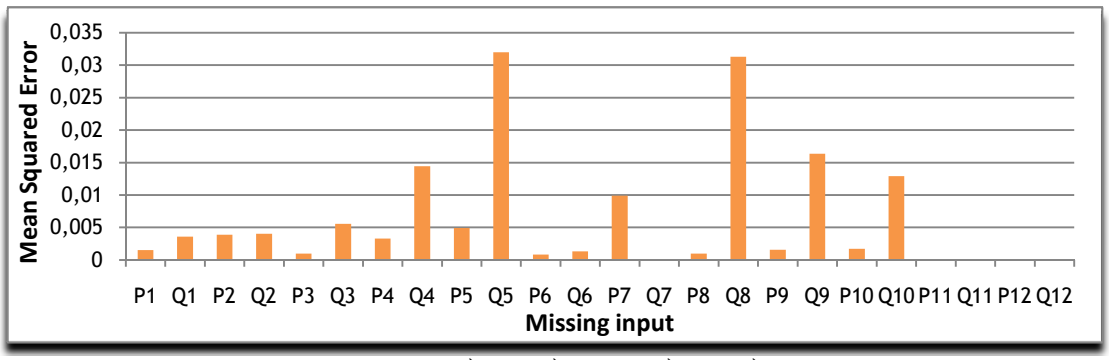

Figure 5.28 - MSE for Restoration of $\vec{P}_1$ to $\vec{P}_{12}$ and $\vec{Q}_1$ to $\vec{Q}_{12}$ after 20 iterations using Constrained Search with EPSO.

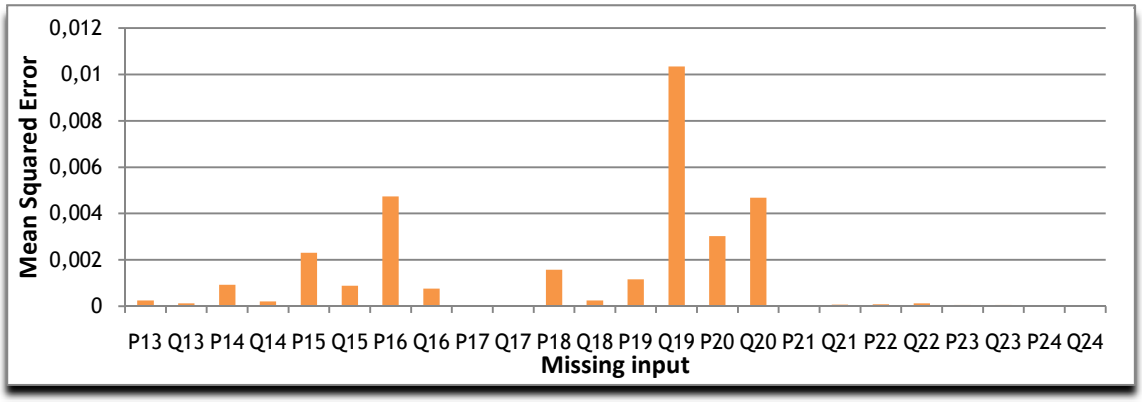

Figure 5.29 - MSE for Restoration of $\vec{P}_{13}$ to $\vec{P}_{24}$ and $\vec{Q}_{13}$ to $\vec{Q}_{24}$ after 20 iterations using Constrained Search with EPSO.

5.4.4 MSR using a Hybrid method: POCS+Constrained Search with EPSO

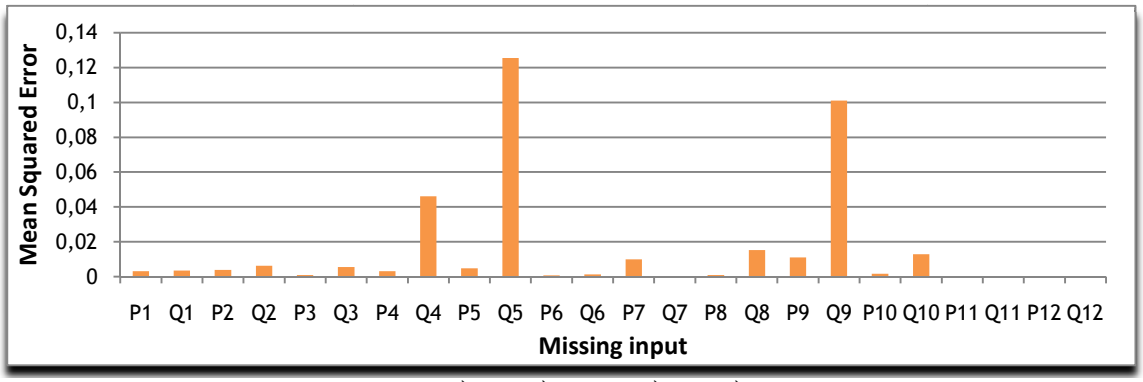

Figure 5.30 - MSE for Restoration of $\vec{P}_{1}$ to $\vec{P}_{12}$ and $\vec{Q}_{1}$ to $\vec{Q}_{12}$ after 20 iterations using a hybrid strategy.

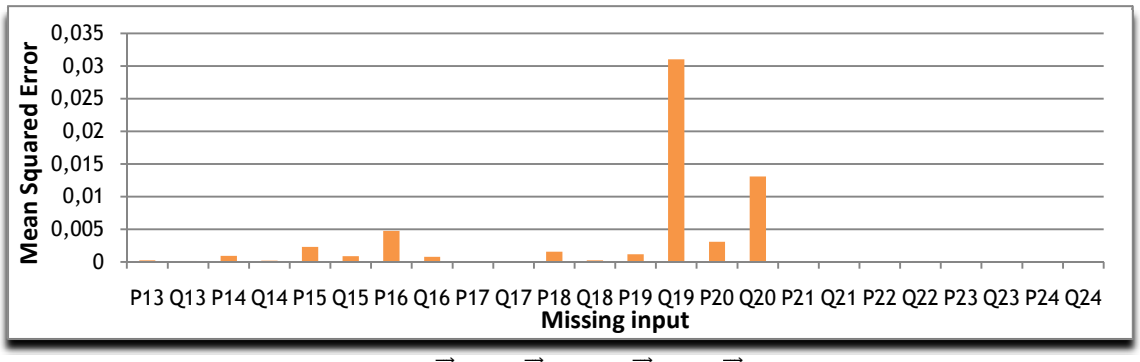

Figure 5.31 - MSE for Restoration of $\vec{P}_{13}$ to $\vec{P}_{24}$ and $\vec{Q}_{13}$ to $\vec{Q}_{24}$ after 20 iterations using a hybrid strategy.

5.4.5 Multiple Missing Sensor Restoration

Figure 5.32 - MSE for the various restoration algorithms in function of the number of missing inputs.

5.4.6 Comparison of the Recovery algorithms

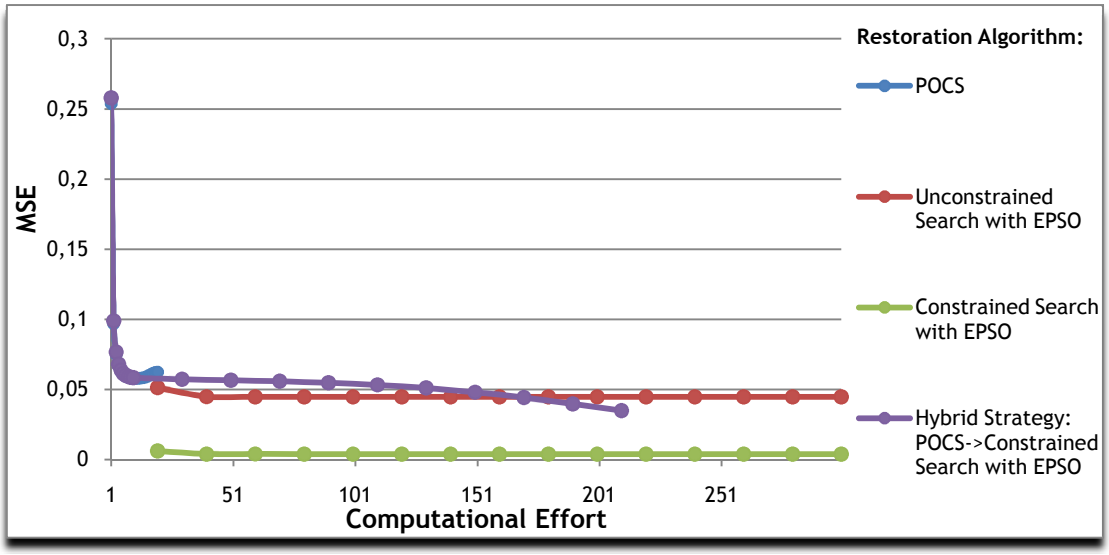

Figure 5.33 - MSE for the various restoration algorithms in function of the computational effort

Table 5.2 - Comparison of the recovery performance for 10 missing inputs (destandardized values)

		Restoration algorithm							
Missing Inputs		POCS		Unconstrained Search with EPSO		Constrained Search with EPSO		Hybrid Strategy: POCS->Constrained Search with EPSO	
Name	Correct Value	Restored Value	Error	Restored Value	Error	Restored Value	Error	Restored Value	Error
P1 (MW)	94,65	125,91	-31,26	91,74	2,91	90,68	3,97	90,90	3,75
Q1 (MVAR)	-70,65	-68,32	-2,33	-65,91	-4,74	-64,77	-5,88	-65,95	-4,70
P2 (MW)	106,29	101,84	4,46	115,48	-9,18	115,56	-9,27	106,00	0,30
Q2 (MVAR)	-68,58	-62,01	-6,57	-65,39	-3,19	-68,91	0,33	-65,54	-3,05
P3 (MW)	108,75	107,16	1,58	116,19	-7,45	117,90	-9,16	114,51	-5,76
Q3 (MVAR)	22,62	35,38	-12,76	23,74	-1,13	22,91	-0,30	22,68	-0,06
P4 (MW)	61,04	70,13	-9,08	52,20	8,84	58,42	2,63	69,24	-8,20
Q4 (MVAR)	9,72	9,46	0,26	13,56	-3,84	12,13	-2,41	14,53	-4,81
P5 (MW)	64,86	44,32	20,54	63,74	1,12	61,77	3,09	44,05	20,82
Q5 (MVAR)	9,66	8,65	1,01	10,89	-1,24	10,78	-1,12	10,91	-1,25
		MAE	8,99		4,36		3,81		5,27

5.5 Restoring Voltages, Angles and Branches Status

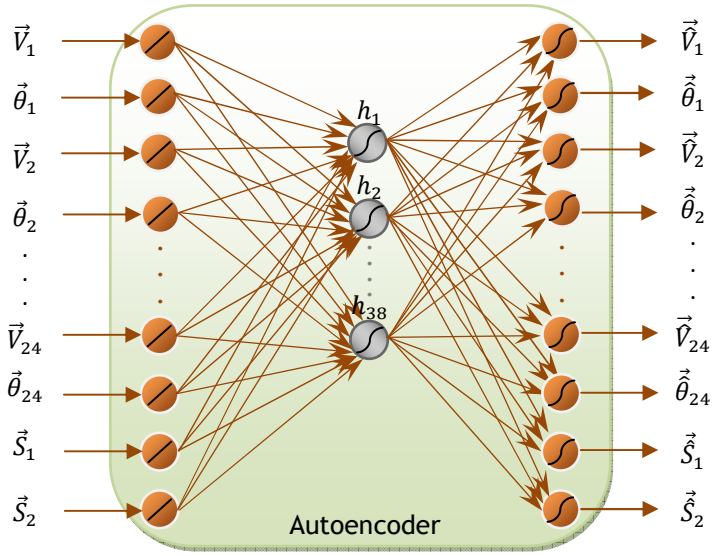

Figure 5.34 - Autoencoder structure used to restore Voltages, angles and switching device status.

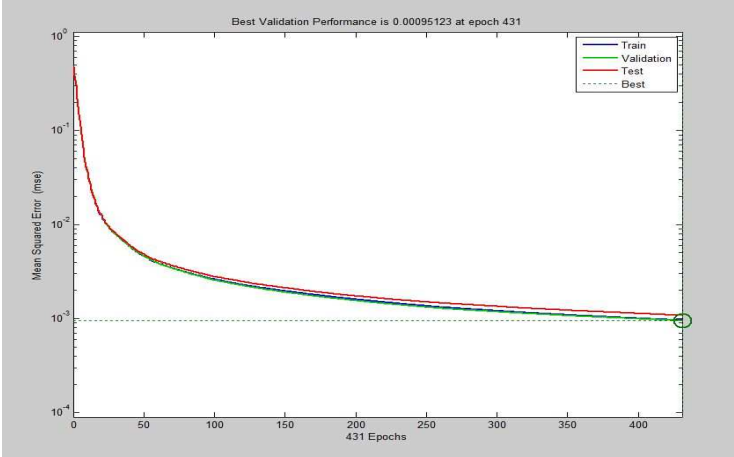

Figure 5.35 - Evolution of the training procedure for the autoencoder used to restore voltage magnitude, angle and branches status.

5.5.1 MSR using POCS

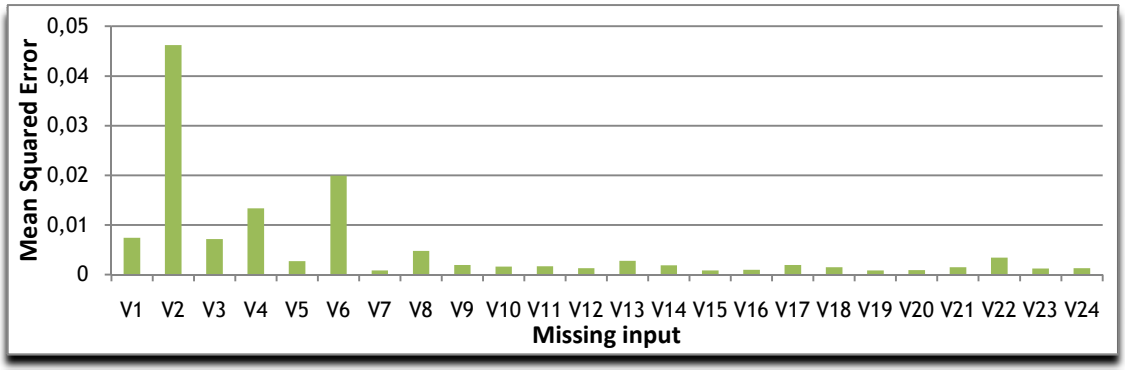

Figure 5.36 - MSE for Restoration of Voltage magnitude after 20 iterations using POCS.

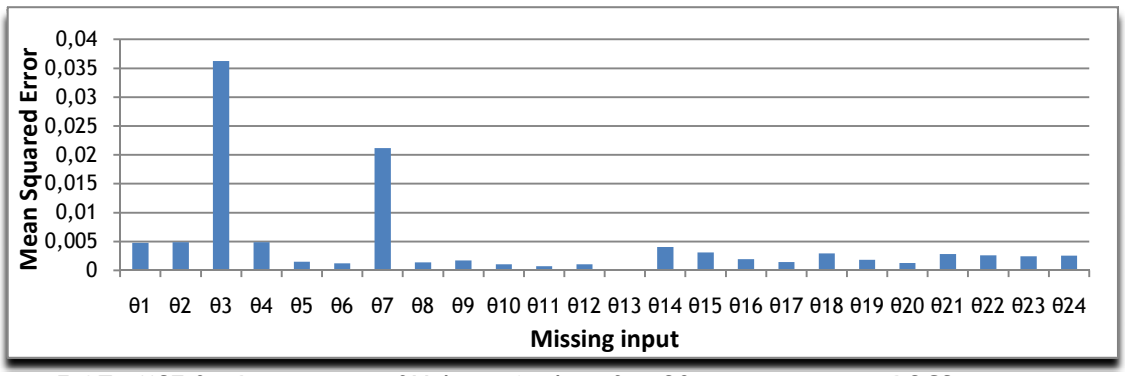

Figure 5.37 - MSE for Restoration of Voltage Angles after 20 iterations using POCS.

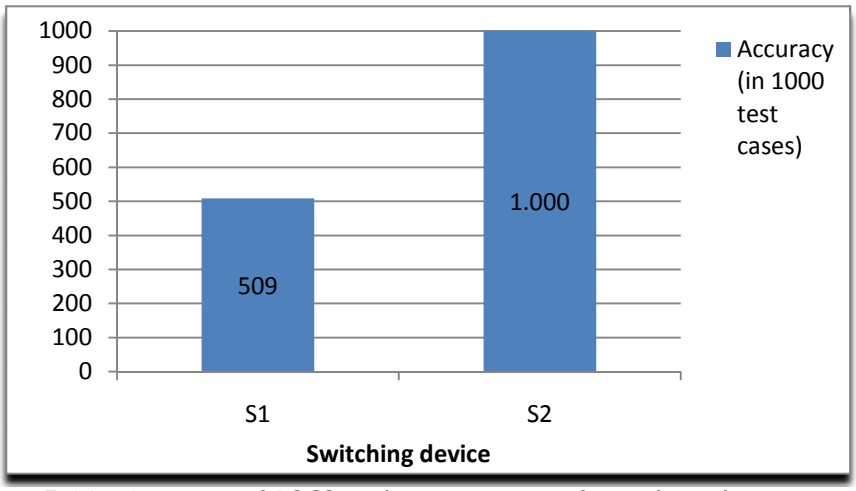

Figure 5.38 – Accuracy of POCS in the restoration of switching device status.

5.5.2 MSR using Unconstrained Search with EPSO

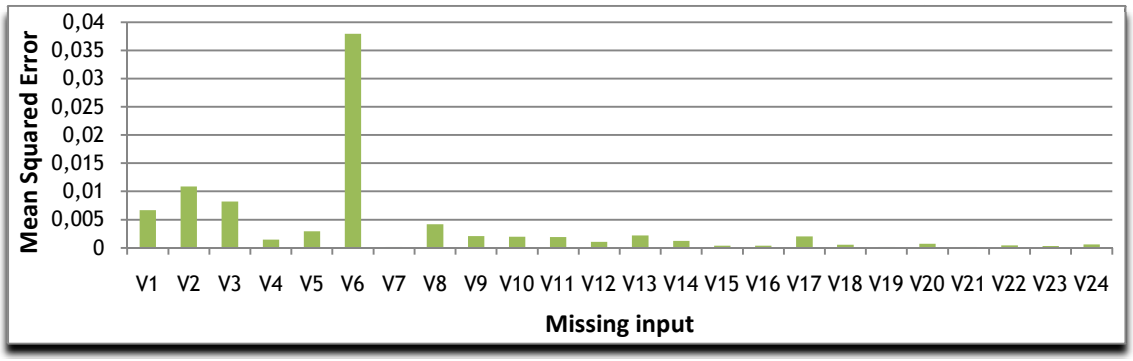

Figure 5.39 - MSE for Restoration of Voltage Magnitudes after 20 iterations using Unconstrained Search with EPSO.

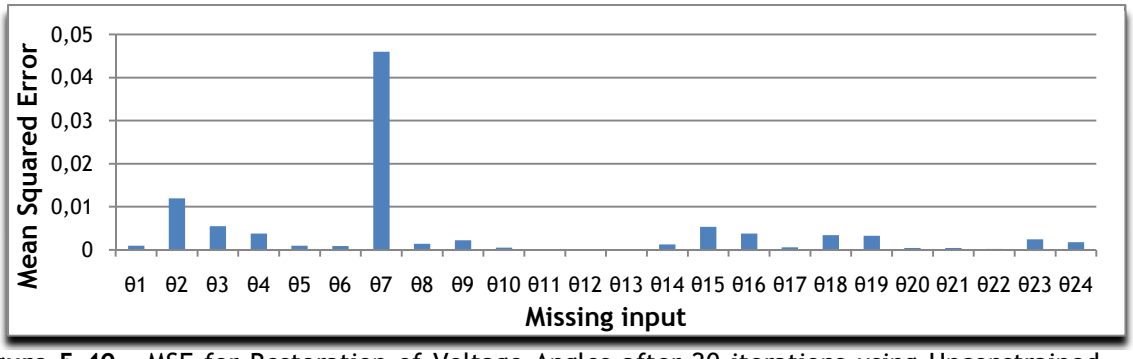

Figure 5.40 - MSE for Restoration of Voltage Angles after 20 iterations using Unconstrained Search with EPSO.

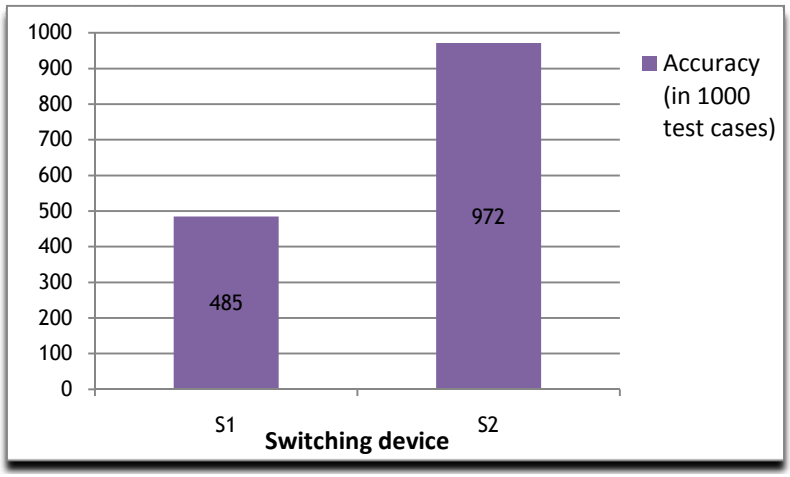

Figure 5.41 - Accuracy of Unconstrained Search with EPSO in the restoration of switching device status.

5.5.3 MSR using Constrained Search with EPSO

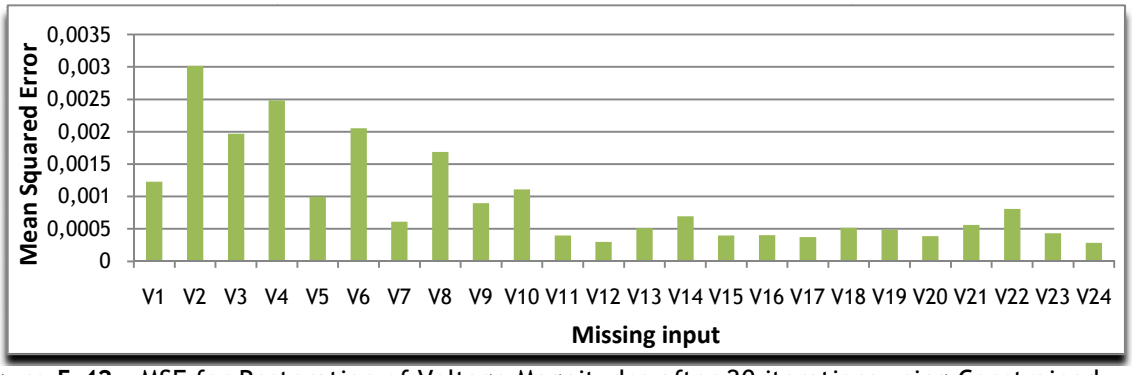

Figure 5.42 - MSE for Restoration of Voltage Magnitudes after 20 iterations using Constrained Search with EPSO.

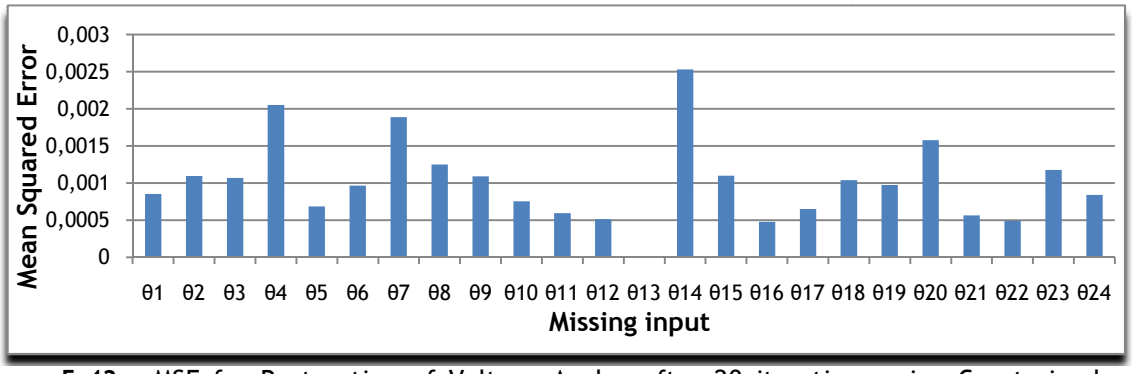

Figure 5.43 - MSE for Restoration of Voltage Angles after 20 iterations using Constrained Search with EPSO.

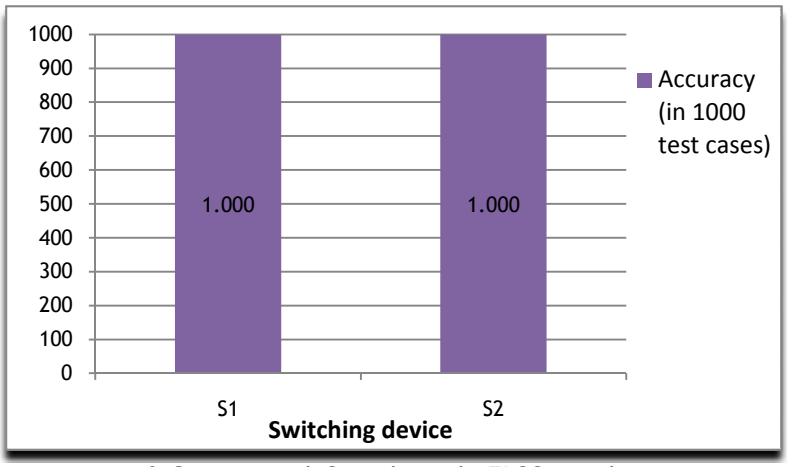

Figure 5.44 - Accuracy of Constrained Search with EPSO in the restoration of switching device status.

5.5.4 MSR using a Hybrid method: POCS+Constrained Search with EPSO

Figure 5.45 - MSE for Restoration of Voltage Magnitudes after 20 iterations using hybrid strategy.

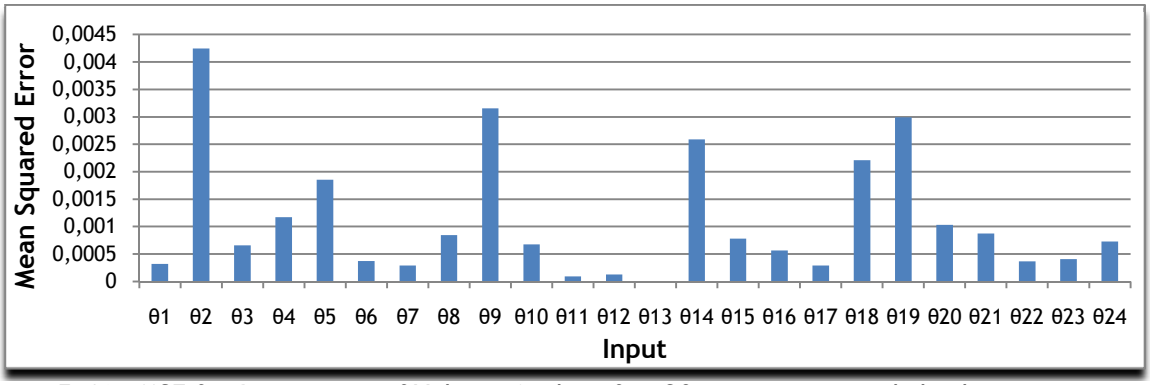

Figure 5.46 - MSE for Restoration of Voltage Angles after 20 iterations using hybrid strategy.

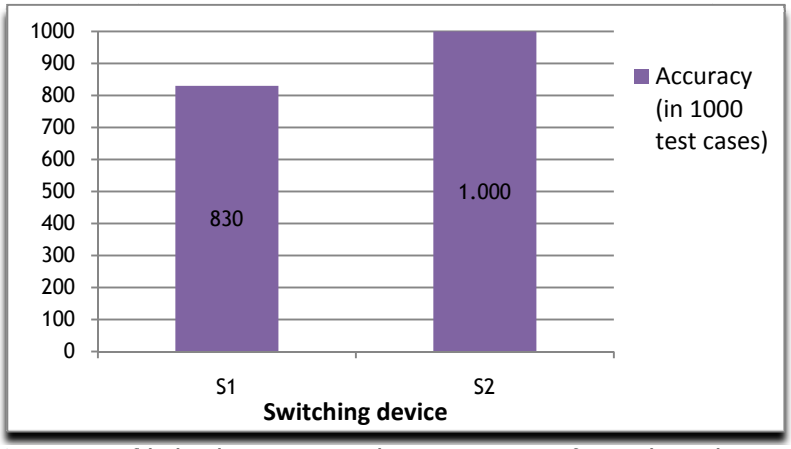

Figure 5.47 - Accuracy of hybrid strategy in the restoration of switching device status.

5.5.5 Comparison of the Recovery algorithms

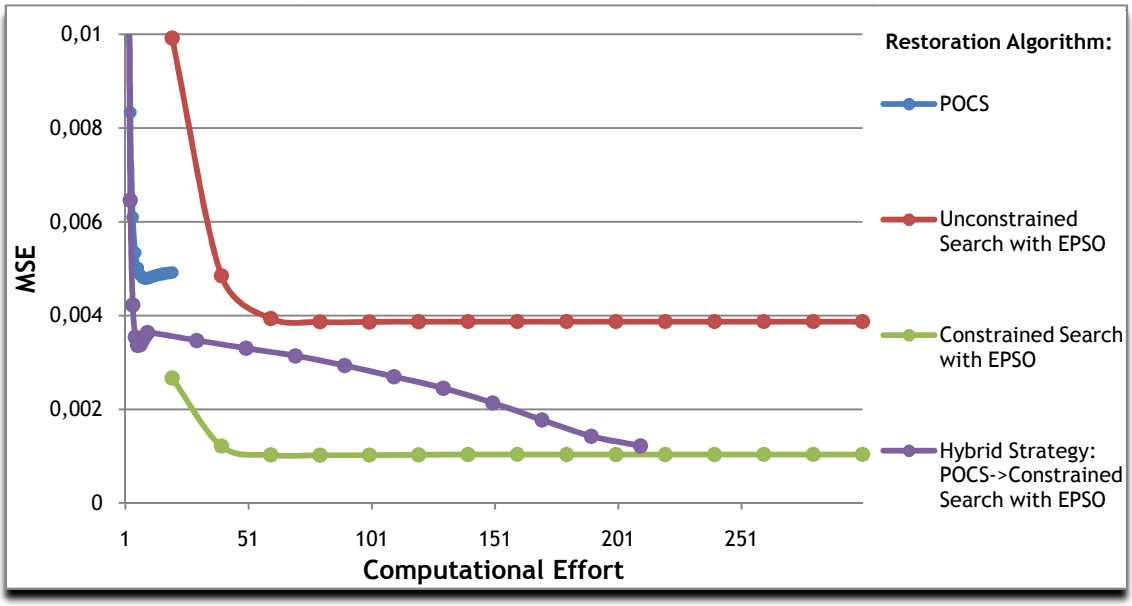

Figure 5.48 - MSE for the various restoration algorithms in function of the computational effort.

Table 5.3 - Comparison of the recovery performance for 10 missing inputs (destandardized values).

		Restoration algorithm							
Missing Inputs		POCS		Unconstrained Search with EPSO		Constrained Search with EPSO		Hybrid Strategy: POCS->Constrained Search with EPSO	
Name	Correct Value	Restored Value	Error	Restored Value	Error	Restored Value	Error	Restored Value	Error
V1 (p.u.)	0,94911	0,927	0,02	0,927	0,02	0,950	0,00	0,9544	-0,01
θ1 (degrees)	-13,7416	-14,900	1,16	-14,900	1,16	-11,403	-2,34	-11,0654	-2,68
V2 (p.u.)	0,950552	0,961	-0,01	0,961	-0,01	0,954	0,00	0,9613	-0,01
θ2 (degrees)	-13,5449	-13,565	0,02	-13,565	0,02	-11,695	-1,85	-11,3515	-2,19
V3 (p.u.)	0,947846	0,928	0,02	0,928	0,02	0,945	0,00	0,9428	0,01
θ3 (degrees)	-18,2561	-19,959	1,70	-19,959	1,70	-18,871	0,61	-18,7591	0,50
V4 (p.u.)	0,937219	0,957	-0,02	0,957	-0,02	0,940	0,00	0,9472	-0,01
θ4 (degrees)	-18,2981	-17,309	-0,99	-17,309	-0,99	-17,963	-0,34	-17,5297	-0,77
S1	1	1	0,00	1	0,00	1,000	0,00	1	0,00
S2	0	0	0,00	0	0,00	0,000	0,00	0	0,00
		MAE	0,39		0,39		0,51		0,62

The results confirm the efficiency of restoration scheme. The method has been able to provide correct answers even when several measurements are missing.

5.6 Restoring injected Active and Reactive Power and Branches Status

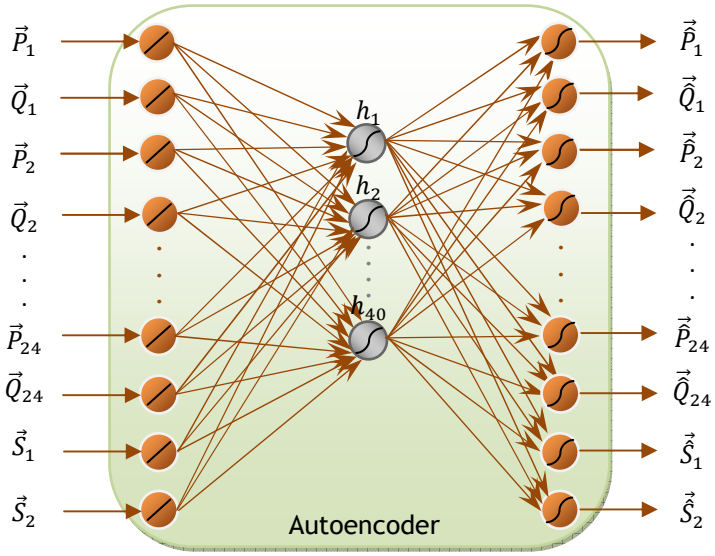

Figure 5.49 - Autoencoder structure used to restore injective Active and Reactive Power and switching device status.

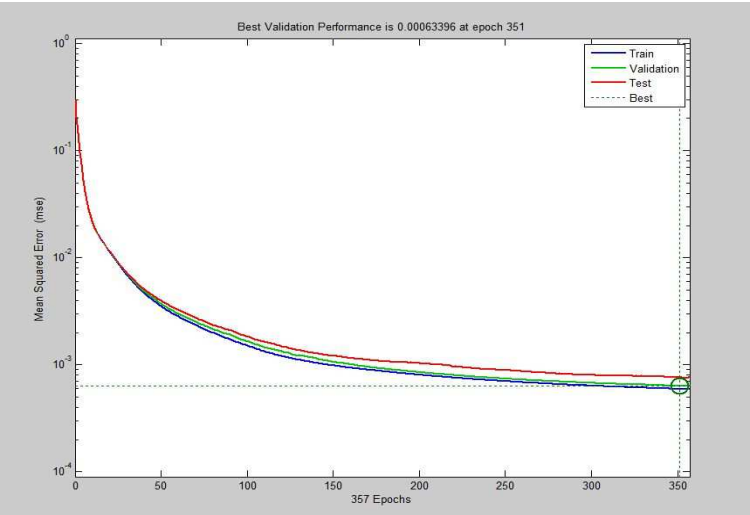

Figure 5.50 - Evolution of the training procedure for the autoencoder used to restore injected Active and Reactive Power and switching device status.

5.6.1 MSR using POCS

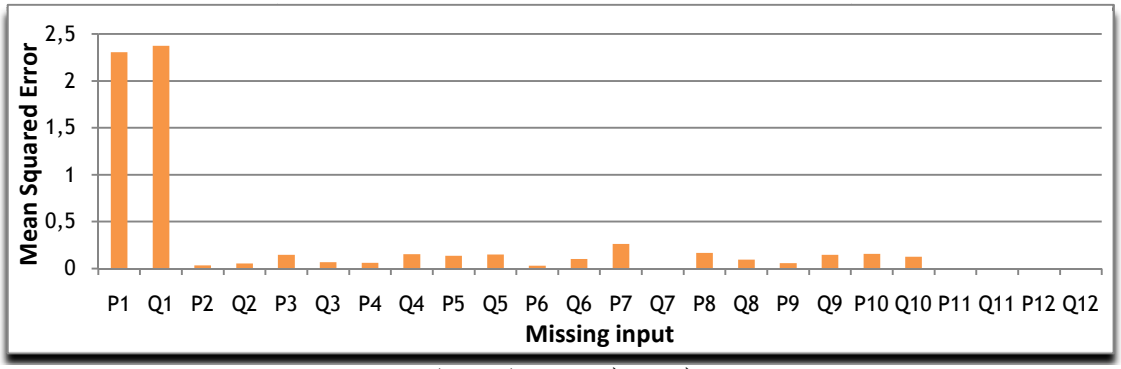

Figure 5.51 - MSE for Restoration of $\vec{P}_1$ to $\vec{P}_{12}$ and $\vec{Q}_1$ to $\vec{Q}_{12}$ after 20 iterations using POCS.

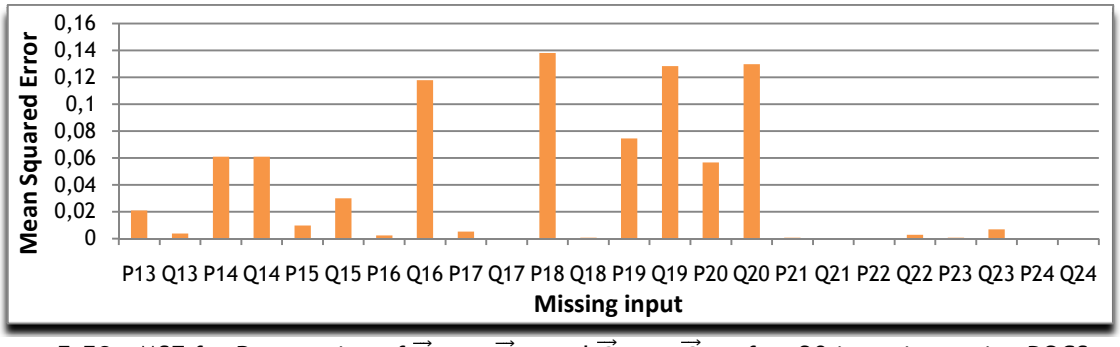

Figure 5.52 - MSE for Restoration of $\vec{P}_{13}$ to $\vec{P}_{24}$ and $\vec{Q}_{13}$ to $\vec{Q}_{24}$ after 20 iterations using POCS.

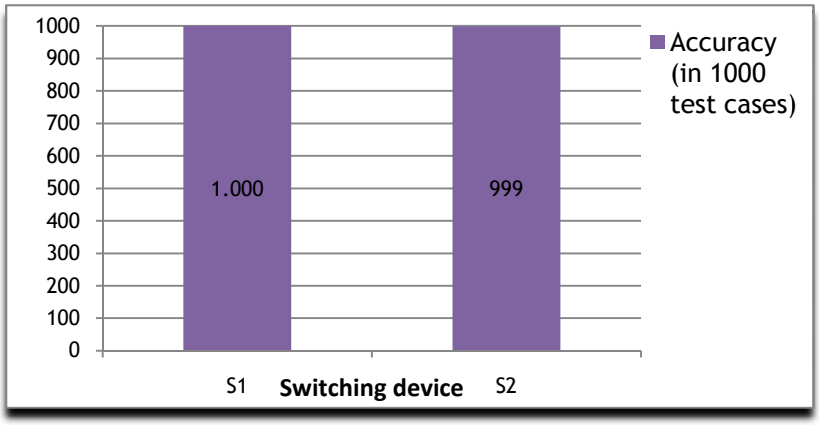

Figure 5.53 - Accuracy of POCS in the restoration of switching device status.

5.6.2 MSR using Unconstrained Search with EPSO

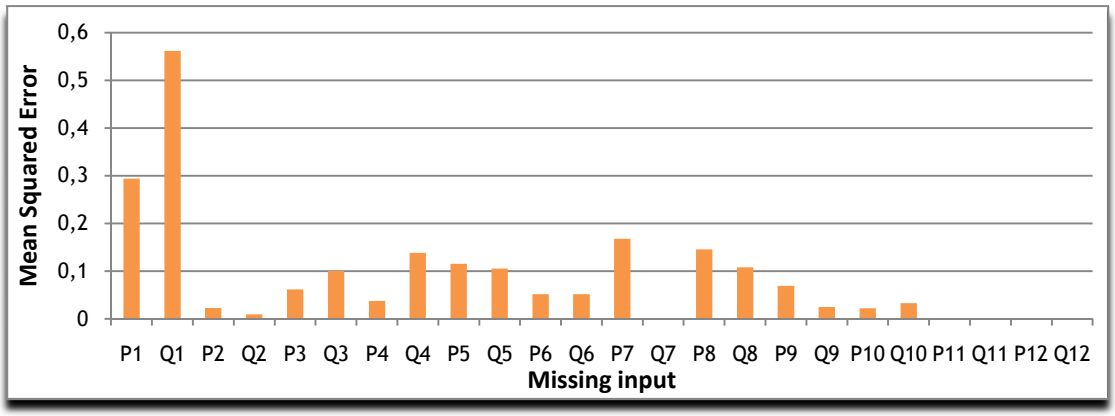

Figure 5.54 - MSE for Restoration of $\vec{P}_1$ to $\vec{P}_{12}$ and $\vec{Q}_1$ to $\vec{Q}_{12}$ after 20 iterations using Unconstrained Search with EPSO.

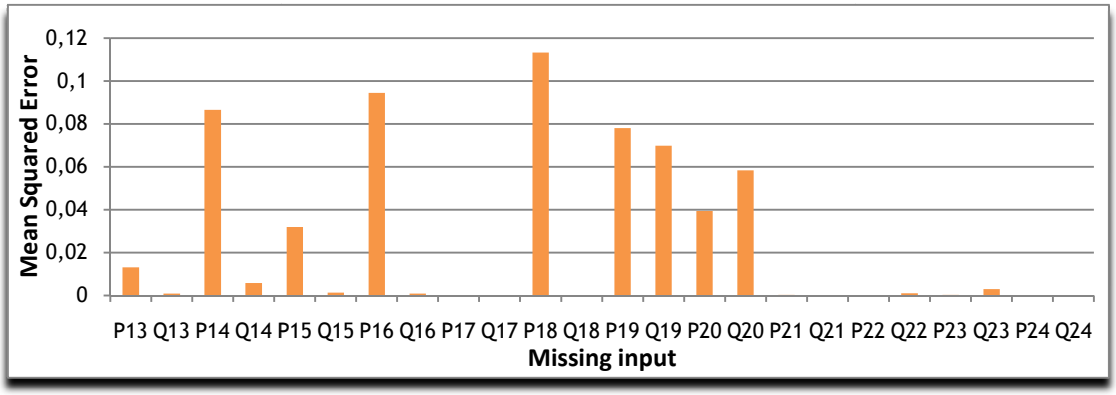

Figure 5.55 - MSE for Restoration of $\vec{P}_{13}$ to $\vec{P}_{24}$ and $\vec{Q}_{13}$ to $\vec{Q}_{24}$ after 20 iterations using Unconstrained Search with EPSO.

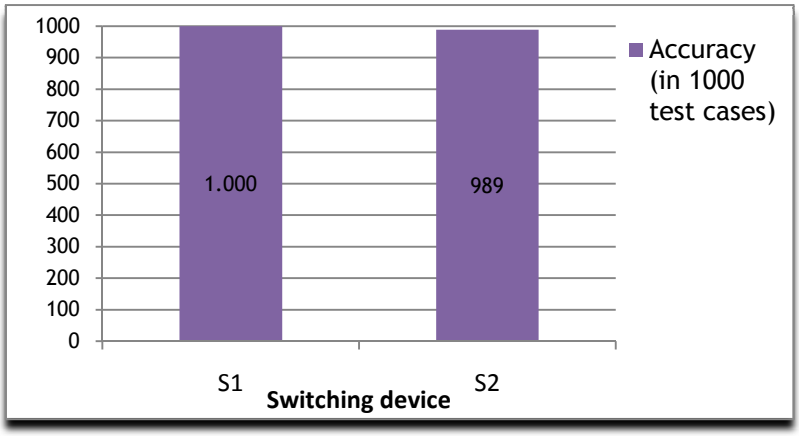

Figure 5.56 - Accuracy of Unconstrained Search with EPSO in the restoration of switching device status.

5.6.3 MSR using Constrained Search with EPSO

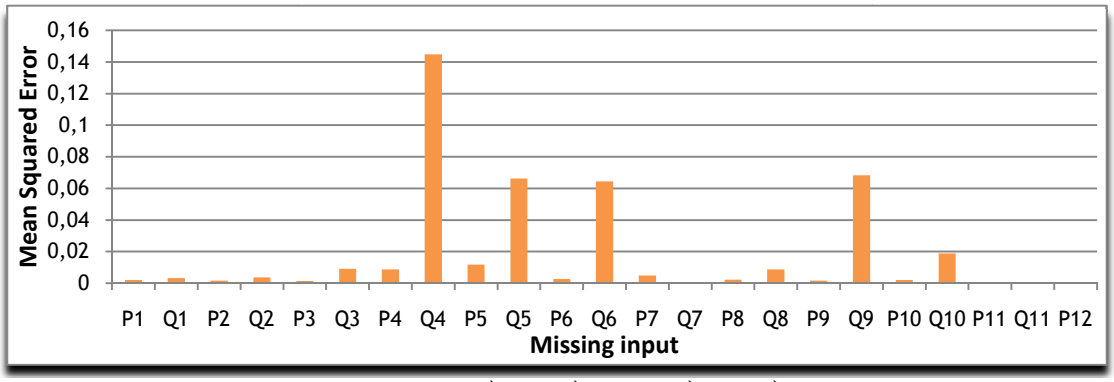

Figure 5.57 - MSE for Restoration of $\vec{P}_{1}$ to $\vec{P}_{12}$ and $\vec{Q}_{1}$ to $\vec{Q}_{12}$ after 20 iterations using Constrained Search with EPSO.

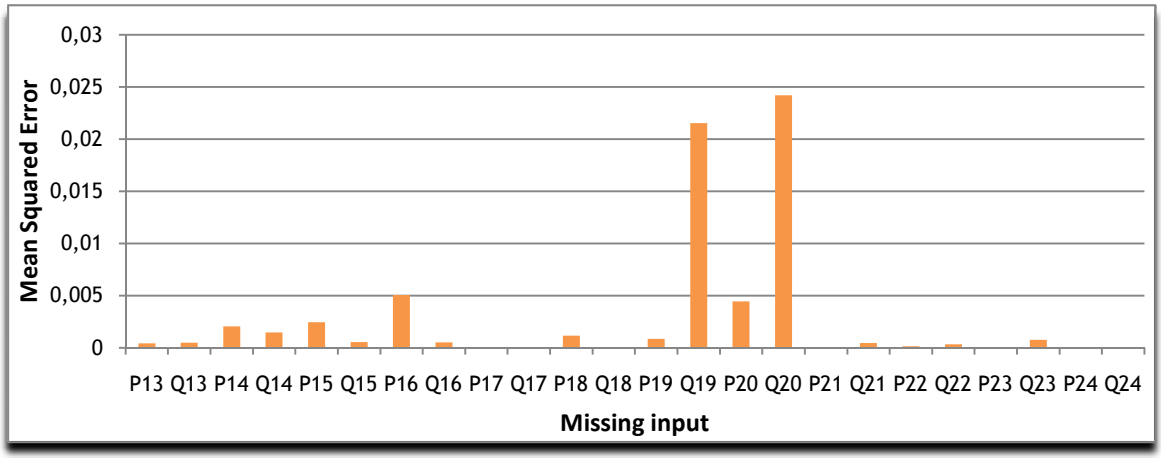

Figure 5.58 - MSE for Restoration of $\vec{P}_{13}$ to $\vec{P}_{24}$ and $\vec{Q}_{13}$ to $\vec{Q}_{24}$ after 20 iterations using Constrained Search with EPSO.

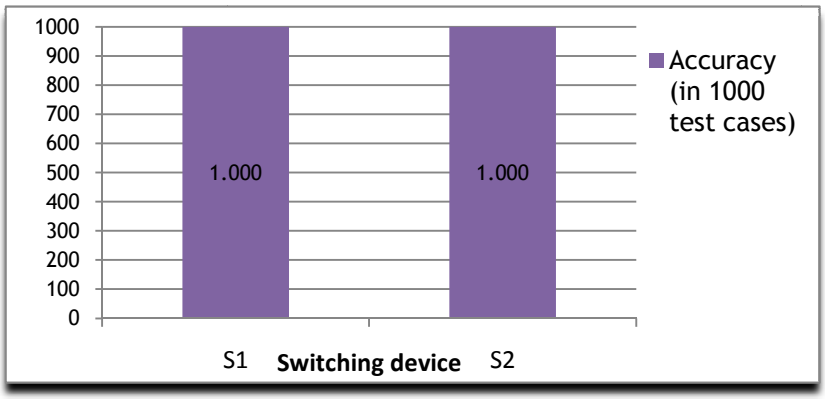

Figure 5.59 - Accuracy of Constrained Search with EPSO in the restoration of switching device status.

5.6.4 MSR using a Hybrid method: POCS+Constrained Search with EPSO

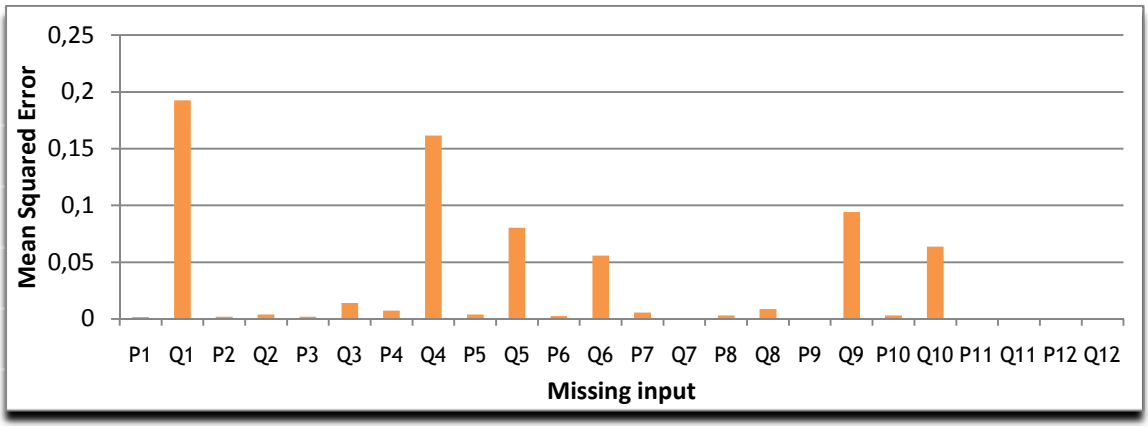

Figure 5.60 - MSE for Restoration of $\vec{P}_1$ to $\vec{P}_{12}$ and $\vec{Q}_1$ to $\vec{Q}_{12}$ after 20 iterations using hybrid strategy.

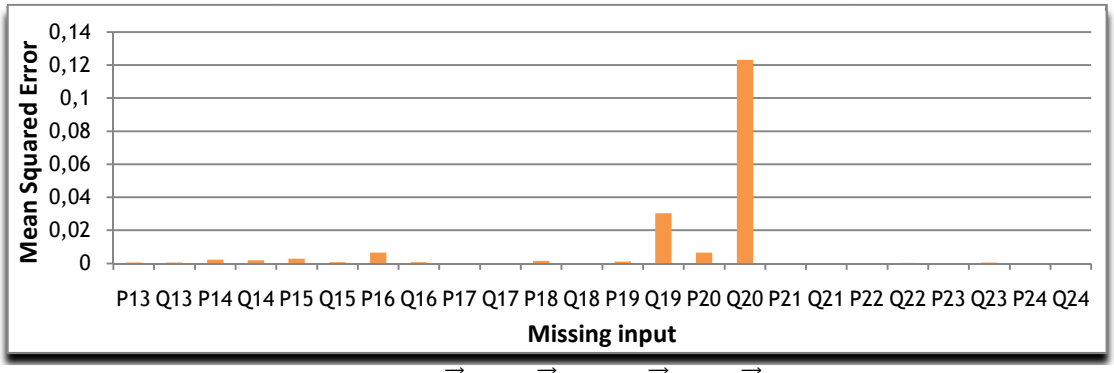

Figure 5.61 - MSE for Restoration of $\vec{P}_{13}$ to $\vec{P}_{24}$ and $\vec{Q}_{13}$ to $\vec{Q}_{24}$ after 20 iterations using hybrid strategy.

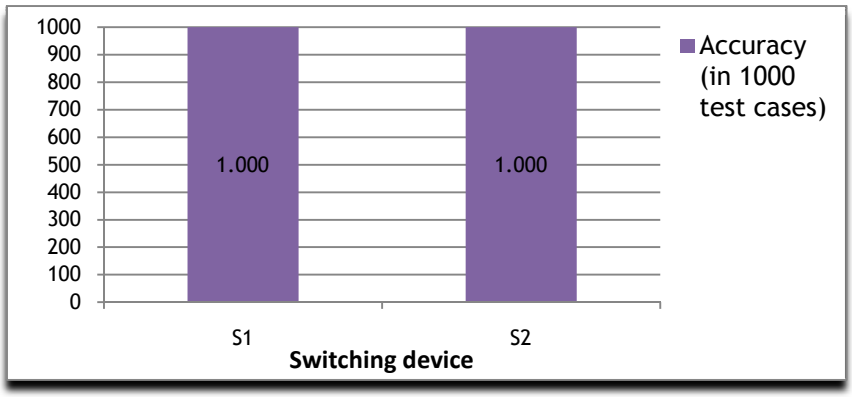

Figure 5.62 - Accuracy of hybrid strategy in the restoration of switching device status.

Table 5.4 - Comparison of the recovery performance for 10 missing inputs (destandardized values).

		Restoration algorithm							
Missing Inputs		POCS		Unconstrained Search with EPSO		Constrained Search with EPSO		Hybrid Strategy: POCS->Constrained Search with EPSO	
Name	Correct Value	Restored Value	Error	Restored Value	Error	Restored Value	Error	Restored Value	Error
P1 (MW)	88,35323	-63,414	151,77	75,141	13,21	-63,526	151,88	-63,4136	151,77
Q1 (MVAR)	-62,6982	4,947	-67,64	4,828	-67,53	4,898	-67,60	4,9466	-67,64
P2 (MW)	135,1849	124,514	10,67	124,757	10,43	124,554	10,63	124,5143	10,67
Q2 (MVAR)	-67,4636	-69,782	2,32	-66,422	-1,04	-69,762	2,30	-69,7822	2,32
P3 (MW)	157,5014	168,877	-11,38	178,691	-21,19	169,377	-11,88	168,8778	-11,38
Q3 (MVAR)	29,82855	29,702	0,13	29,663	0,17	29,789	0,04	29,7016	0,13
P4 (MW)	57,32578	60,505	-3,18	60,576	-3,25	60,471	-3,15	60,5048	-3,18
Q4 (MVAR)	13,55458	14,664	-1,11	13,962	-0,41	14,653	-1,10	14,6638	-1,11
S1	1	0	1,00	0	1,00	0,000	1,00	0	1,00
S2	0	0	0,00	0	0,00	0,000	0,00	0	0,00
		MAE	24,92		11,82		24,96		24,92

5.7 Chapter conclusions

At this chapter, one was able to confirm that missing restoration with autoencoders can be applied to power systems. Therefore, we were able to successfully recover measurements of voltage amplitude and angle, injected active and reactive power and switching device status. By being able to successfully restore missing measurements in a much more complex AC 24-bus network than the parabola examples of Chapter 4, the recovery scheme proved that it can be an effective way to improve SCADA system reliability.

In the restoration of voltage magnitude and angle measurements, all the proposed recovery algorithms were tested to see their performance. All the algorithms have provided good results but the best is constrained search with EPSO. This overall good performance relies essentially on the strong correlation between voltage magnitudes/angles at each bus. The hybrid strategy has proven to be a good alternative but, when the number of missing measurements is high, POCS doesn't converge and has a bad influence in the hybrid approach. So for the cases of several missing measurements the best algorithms are constrained and unconstrained search with EPSO.

For the experiment of restoration of injected active and reactive power the constrained search with EPSO has provided the best results followed by the hybrid strategy. When several measurements are missing, unconstrained and constrained search with EPSO are also the best. The weak correlation existent between the measurements of active and reactive injected power explains why the MSE on the recovery of these measurements are, in general, superiors to the ones verified in the recovery of voltage magnitude and angles values.

In the experiment of restoring voltage magnitudes and angles and switching devices status the result were good, especially in the recovery of switching devices status and using constrained search with EPSO.

Finally, in the test involving the recovery of injected active and reactive power and switching device status, it was achieved a high accuracy in the determination of branches status. However, the inclusion of lines status came to harm the restoration on injected active and reactive power. As seen by the high MSE obtained in the recovery of this measurements, in this test. Maybe the inclusion of transmission line power flows can increase the correlation among the measures and improve the performance of the method.

Conclusions

6.1 General Conclusions

The results obtained in this thesis prove that an autoencoder is capable of learning the correlations existent between the variables in a power system. Then, by the means of a restoration algorithm, is possible to use the trained autoencoder to restore missing measurements.

The strong correlation between voltage amplitudes and voltage angles through the various buses, allow the recovery algorithms to properly restore missing measurements from the remaining measures. However, an increase in the number of missing measurements makes the recovery process more complex and the errors in the restored measures will be increased as well.

6.2 Future studies and developments

One of the first future studies than can be made relatively to the application of missing sensor restoration to power systems is an increase in the number of switches devices. I believe that the autoencoder will be able to perform well in the detection of system topology considering large number of branches status.

Another important future study that can be made is inclusion of both the injected power on the buses and the power flow on the transmission lines. In spite of increasing the number of measurements, the suggested study, allows the autoencoder to gather more information about the system.

Other significant future study is using different types of autoencoder neural networks such as the stacked autoencoder without Restricted Boltzmann Machine (RBM), autoencoder with RBM and stacked autoencoder with RBM as described in [24-25].

The recovering strategy developed using EPSO and autoencoders can be applied to other problems, however that a strong correlation among the measurements must exist.

References

[1] E. Manitsas, *et al.*, "Modelling of pseudo-measurements for distribution system state estimation," 2008.

[2] A. Ukil, *Intelligent Systems and Signal Processing in Power Engineering*: Springer Publishing Company, Incorporated, 2007.

[3] J. D. Cowan, "Neural networks: the early days," in *Advances in neural information processing systems 2*, ed: Morgan Kaufmann Publishers Inc., 1990, pp. 828-842.

[4] S. A. P. Marvin Minsky, *Perceptrons : an introduction to computational geometry*: MIT Press, 1969.

[5] V. Miranda, "Redes Neuronais - treino por retropropagação," *Engineering University of Porto,* June 2007.

[6] G. E. Hinton and R. R. Satakhutdinov, "Reducing the dimensionality of data with neural networks," *5786,* 2006.

[7] I. T. Jollife, *Principal component analysis*. New York: Springer, 2002.

[8] K. Hornik, *et al.*, "Multilayer feedforward networks are universal approximators," *5,* 1989.

[9] B. B. Thompson, *et al.*, "On the Contractive Nature of Autoencoders: Application to Missing Sensor Restoration," 2003.

[10] S. Suzuki and H. Ando, "A modular network scheme for unsupervised 3D object recognition," *NEUROCOMPUTING,* vol. CCCT, pp. 15-28, 2000.

[11] A. Bernieri, *et al.*, "A neural network approach to instrument fault detection and isolation," vol. 1, 1994.

[12] Q. Wei, *et al.*, "Robust neuro-identification of nonlinear plants in electric power systems with missing sensor measurements," *4,* 2008.

[13] S. Narayanan, *et al.*, "Set constraint discovery: Missing sensor data restoration using auto-associative regression machines," Honolulu, HI, United states, 2002, pp. 2872-2877.

[14] H. K. Vladimiro Miranda, Álvaro Jaramillo Duque, "Stochastic Star Communication Topology in Evolutionary Particle Swarms (EPSO) " *International Journal of Computational Intelligence Research,* vol. 4, pp. 105–116, 2008.

[15] V. Miranda and N. Fonseca, "EPSO - Best-of-two-worlds meta-heuristic applied to power system problems," *CEC'02: PROCEEDINGS OF THE 2002 CONGRESS ON EVOLUTIONARY COMPUTATION, VOLS 1 AND 2,* vol. 1, pp. 1080-1085, 2002.

[16] J. C. Pereira, *et al.*, "Comparison of approaches to identify topology errors in the scope of state estimation studies," vol. 3, 2001.

[17] Louren, *et al.*, "Bayesian-based hypothesis testing for topology error identification in generalized state estimation," *2,* 2004.

[18] S. Lee, *et al.*, "Conformal radiotherapy computation by the method of alternating projections onto convex sets," *6,* 1997.

[19] J. Park, *et al.*, "Recovery of image blocks using the method of alternating projections," *4,* 2005.

[20] H. Sun and W. Kwok, "Concealment of damaged block transform coded images using projections onto convex sets," *IEEE Transactions On Image Processing: A Publication Of The IEEE Signal Processing Society,* vol. 4, pp. 470-7, 1995.

[21] F. F. Wu and W.-H. E. Liu, "Detection of topology errors by state estimation," *1,* 1989.

[22] C. Grigg, *et al.*, "The IEEE reliability test system - 1996," *IEEE TRANSACTIONS ON POWER SYSTEMS,* vol. 14, pp. 1010-1018, 1999.

[23] T. Kristiansen, "Utilizing MATPOWER in optimal power flow," *MODELING IDENTIFICATION AND CONTROL,* vol. 24, pp. 49-59, 2003.

[24] C. C. Tan and C. Eswaran, "Performance comparison of three types of autoencoder neural networks," *Proceedings - 2nd Asia International Conference on Modelling and Simulation, AMS 2008,* p. 213, 2008.

[25] C. C. Tan and C. Eswaran, "Reconstruction of handwritten digit images using autoencoder neural networks," *2008 CANADIAN CONFERENCE ON ELECTRICAL AND COMPUTER ENGINEERING, VOLS 1-4,* pp. 442-446, 2008.

[26] J. Kennedy and R. Eberhart, "Particle swarm optimization," Perth, Aust, 1995, pp. 1942-1948.

Annexes

ANNEX A

EPSO Algorithm

To solve complex optimization problems, one has nowadays the possibility to use meta-heuristics such as evolutionary algorithms.

To find the optimum solution of a given problem in the context of a search algorithm, two main mechanisms are necessarily considered: a *movement* generator to produce new possible solutions and an *evaluation* procedure to evaluate the alternatives in the search space and classify them in agreement with the optimization objective.

The least efficient search method is a purely random search in the solution space, because all the possible solutions are randomly generated with no information whatsoever of other sampled solutions.

It is important to have smart strategies for searching in the solution space. Three of these cleverer strategies are the "Genetic Algorithms" (GA), the "Particle Swarm Optimization" (PSO) and finally the algorithm used in this thesis, the "Evolutionary Particle Swarm Optimization" (EPSO), which gathers the best qualities of GA and PSO.

In the next two sub-chapters, a brief explanation of the GA and PSO algorithms will be made in order to explain the reader on the advantages that EPSO gathers from both of those algorithms.

A.1 Genetic Algorithms

The implementation of Genetic Algorithms is based on a generation of an initial population (with random values, but, whenever possible, optimized previously to obtain faster and better convergence) upon which the evolution mechanisms are applied. Each individual of this population represents a solution for the problem. The evolutionary mechanisms that are here applied are inspired on biological species evolution like the survival of the fittest and the inter-individual communication to obtain better results for the group.

The main evolutionary operators applied to the population are:

a) Selection;
b) Recombination;
c) Mutation.

The selection is based on the competition among alternatives to identify which will become the departing points or seeds to generate new points. As a result of the evaluation of each solution (individual), two types of selection techniques can be used: elitist and stochastic. The first one guarantees that only the best solutions survive to the next generation. The stochastic "tournament" uses a (usually small) probability that an unfit individual is able to survive. The latter usually gives best results because even the "bad" individuals carry important genes that can help create diversity, thus a better individual in the future like happens in the species evolution.

Mutation is a unary operator, acting on a single individual and responsible for generating a new solution (new individual), by applying random modifications to it. This is a procedure used to ensure higher diversity in solutions, thus helping the algorithm to avoid excessively similar solutions, or in other words, lack of diversity which helps avoiding local optimums.

Recombination (also referred as crossover) generates new individuals (new points in the search space) by randomly mixing characteristics from more than one parent (solutions previously found).

Figure A.1, below, illustrates how the Genetic Algorithm operation procedure works, including the previously detailed procedures: Selection, Mutation and Recombination (crossover). In a first step, a randomly initial population is generated. Then, all the individuals of the initial population are evaluated. After the evaluation procedure, if the convergence test is positive, the best individual is presented as the final solution, otherwise it proceeds to the selection, recombination and mutation mechanisms, returning then to the evaluation procedure, until the convergence test is positive or the maximum number of generations (iterations) is reached.

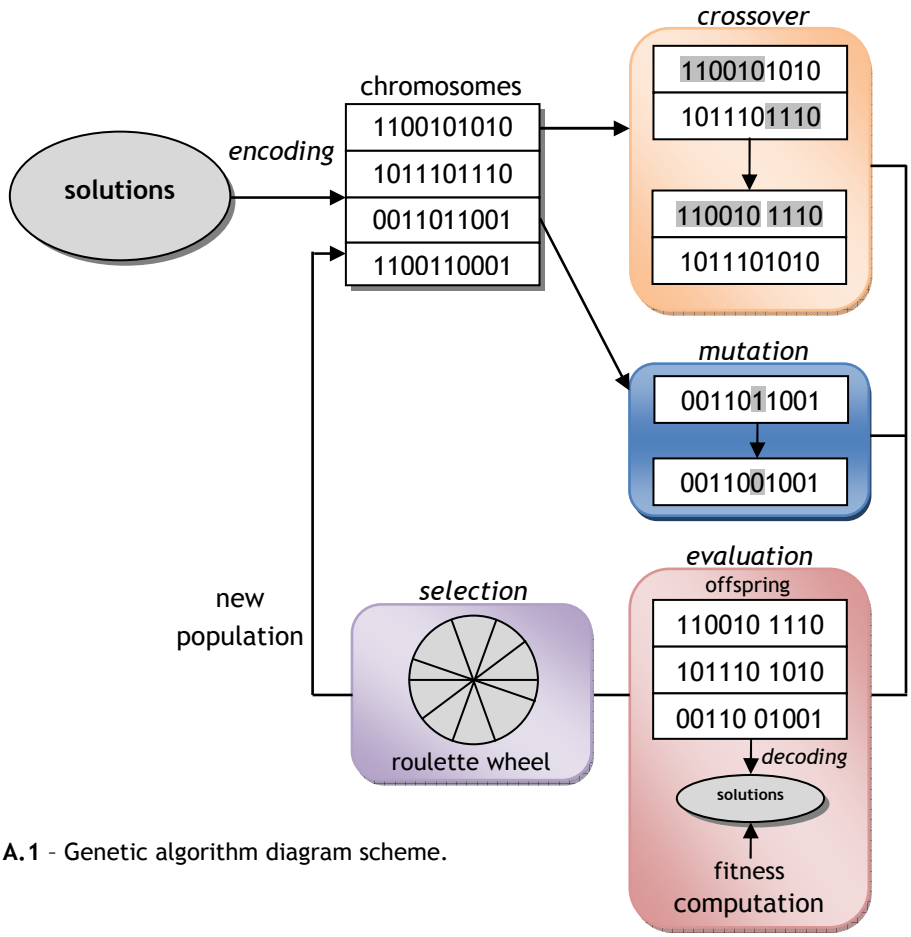

Figure A.1 – Genetic algorithm diagram scheme.

A.2 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) is a relatively new algorithm. It was developed in 1995 by James Kennedy and R. C. Eberhardt [26], revealing competitive characteristics when compared with other meta-heuristic algorithms.

This algorithm is based on the social behavior of living organisms such as swarms of bees, birds or schools of fish, in the way that the group (swarm) and each individual "respond" to the movement as an all.

Each individual or solution is represented as a particle that will move in the search space accordingly to the influence of three vectors:

1. Inertia;
2. Memory;
3. Cooperation.

The first vector (Inertia) will impel the particle in the same direction that it was going. It can be understood like a physical movement inertia that makes it impossible for an object to change instantaneously the speed and direction of the movement.

The second vector (Memory) will attract the particle towards the best position achieved so far by the particle.

The third vector (Cooperation) will attract the particle towards the best position achieved so far by the swarm, regardless what particle have accomplished that position.

Each of these three components will be multiplied by a weight that determines the influence of each vector in the definition of the new position taken by the particle.

The movement equations are thus given by:

$$\boldsymbol{v_i(t+1) = w_{i,0} \times v_i(t) + R(w_{i,1}) \times \left(p_i(t) - x_i(t)\right) + R(w_{i,2}) \times \left(g(t) - x_i(t)\right)} \quad \text{(A.1)}$$

$$\boldsymbol{x_i(t+1) = x_i(t) + v_i(t+1)} \quad \text{(A.2)}$$

Where:

$v_i(t)$ –Velocity on instant t for particle i;

$x_i(t)$ – Position on instant t for particle i;

$p(t)$ – Particle best position on instant t;

$g(t)$ – Global best position on instant t;

$w_{i,0}$ – Inertia coefficient;

$w_{i,1}$ – Memory coefficient;

$w_{i,2}$ – Cooperation coefficient;

R – Randomly distribution of the weights;

By observing the equation (C.1) we can conclude that the inertia term is composed by $w \times v(t)$, the memory term is composed by $R(w) \times \left(p(t) - x(t)\right)$ and the cooperation term is composed by $R(w) \times \left(g(t) - x(t)\right)$.

As we can see, on PSO, unlike other evolutionary algorithms, there is not competition among particles or self adaptation of their characteristics. In reality, if wasn't for the

cooperation term, each particle would completely ignore the other particles on the definition of her new position.

So the particle's new position will be defined by the vector sum represented in figure A.2:

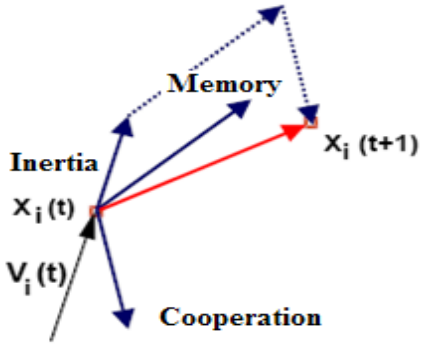

Figure A.2 - Vector diagram of the determination of the particle's new position.

The operation procedure of the PSO algorithm is schematized in figure A.3, below.

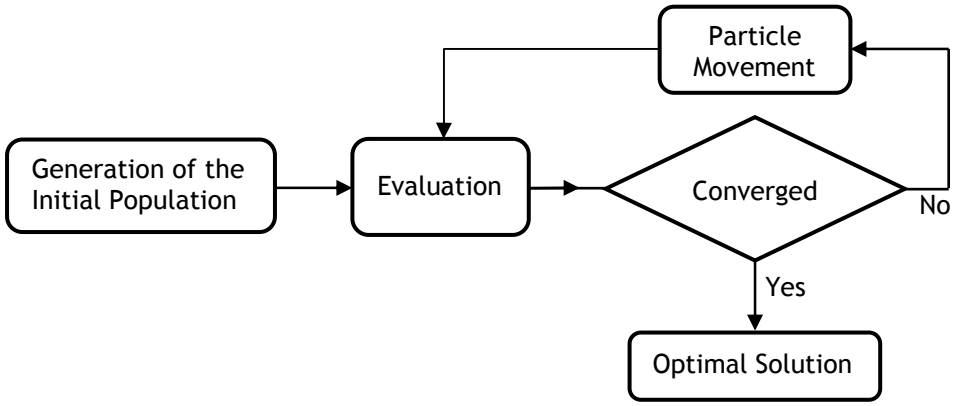

Figure A.3 - Particle Swarm Optimization diagram.

As we can observe in figure A.3, in a first stage a randomly initial population is generated. Then, all the individuals of the initial population are evaluated. After the evaluation procedure, if the convergence test is positive, the best individual is presented as the final solution, otherwise it proceed to the movement mechanism, returning then to the evaluation procedure, until the convergence test is positive or the maximum number of iterations is reached.

A.3 Evolutionary Particle Swarm Optimization (EPSO)

As it was previously demonstrated, the main disadvantage of PSO is that it's not a self adaptive algorithm. Some parameters need to be defined externally by the user, which will determine "fixed weights" for the inertia, memory and cooperation terms in the process. This means that the movement mechanism will not evolve accordingly to the experience acquired in the past iterations, in other words this is a less efficient search method.

In order to suppress these fragilities, the Evolutionary Particle Swarm Optimization (EPSO) was developed which is a hybrid optimization algorithm, developed at INESC – Porto, which gathers the best qualities of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO).

One important difference between the PSO and EPSO algorithms includes the adoption of a stochastic star communication topology, instead of the deterministic scheme usually adopted in PSO. This has the advantage of sharing the knowledge of each particle's knowledge of the global best position, controlled by a communication probability p, which is self-adaptive throughout the algorithm run and also externally defined. The effect produced by the adoption of a stochastic star communication topology is that a particle will ignore the global best on some iterations and include it in other iterations. This not only allows more local search by each particle, but also allows the elimination of disturbing noise, by allowing the dynamics of particle movement to be more stable and avoiding premature convergence.

The other main differences between EPSO and PSO are the selection, replication, weight mutation and movement (reproduction) procedures, as we can see comparing figure A.3 with figure A.4.

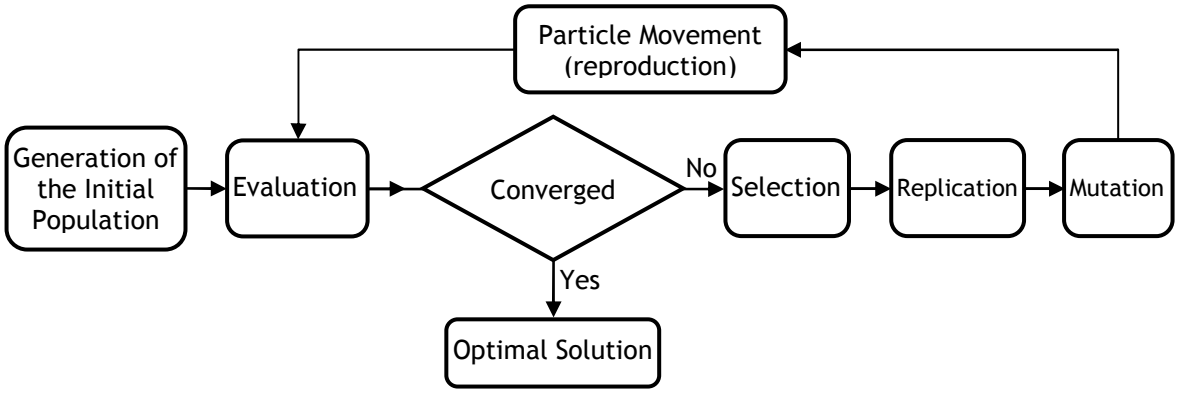

Figure A.4 – Evolutionary Particle Swarm Optimization algorithm diagram.

The selection mechanism applies the competition between particles. This way only the best particles (evaluated previously) "survive" into the next generation (iteration).

The replication mechanism makes a pre-determined number of copies of each particle, which will be mutated afterwards, obtaining in this way more diversity in the population, which helps to avoid local optimums.

The mutation mechanism is one of the main improvements made by the EPSO. In this step the weights from the last iteration (w) are mutated in order to obtain "new" weights (w*) for the coming iteration.

The "new" weights are obtained by multiplying the weight w with random number from lognormal distributions:

$$\boldsymbol{w_i^* = w_i[logN(0,1)]^{\tau}} \quad \text{(A.3)}$$

or by random number Gaussian distributions:

$$\boldsymbol{w_i^* = w_i + \sigma N(0,1)} \quad \text{(A.4)}$$

The τ and σ learning (mutation) parameters are externally defined by the user.

In the most effective EPSO applications not only the weights affecting the components of movement rule are mutated but also the global best position is randomly disturbed to give:

$$\boldsymbol{g^*(t) = g(t) + w^*_{i,3} N(0,1)} \quad \text{(A.5)}$$

where $w^*_{i,3}$ will determine the size of the search space around $g(t)$ where is most likely to find the real best global solution. This helps particles to avoid getting "stuck" in local optimums, in cases where the cooperation term evolves to become dominating while the other terms fade out.

The EPSO movement equations are:

$$\boldsymbol{x_i(t+1) = x_i(t) + v_i(t+1)} \quad \text{(A.6)}$$

$$\boldsymbol{v_i(t+1) = w^*_{i,0} \times v_i(t) + w^*_{i,1} \times \big(p_i(t) - x_i(t)\big) + w^*_{i,2} P\big(g^*(t) - x_i(t)\big)} \quad \text{(A.7)}$$

Where:

$p_i(t)$ - Best point found by particle i in its past life;

$g^*(t)$ - Best overall point found by the swarm so far;

$x_i(t)$ - Location of particle i at generation t;

$v_i(t)$ - Velocity of particle i at generation t;

$w_{i,0}$ - Weight conditioning the inertia term;

$w_{i,1}$ - Weight conditioning the memory term;

$w_{i,2}$ - Weight conditioning the cooperation term;

P - Diagonal matrix with each element in the main diagonal being a binary variable equal to 1 with a given communication probability p and 0 with probability (1-p).

Note: the symbol * indicates that the weights were mutated.

So the particle's new position will be defined by the vector sum represented in figure A.5:

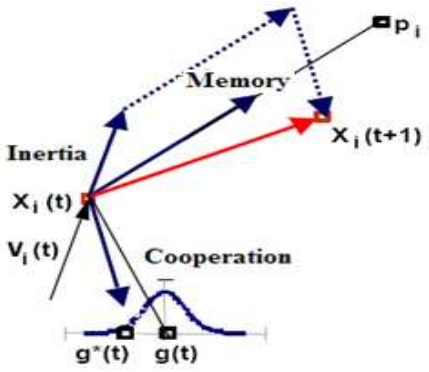

Figure A.5 - Particle's movement with EPSO algorithm

As already said, this is the optimization algorithm adopted on this thesis since it is more efficient and, above all, more easily adaptable.

ANNEX B

The IEEE Reliability Test System

B.1 Network topology

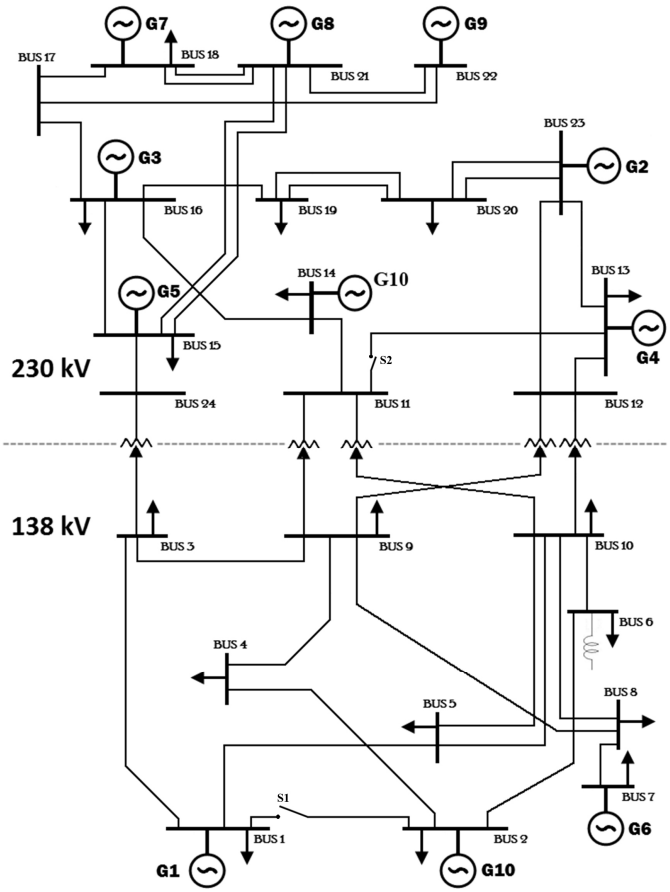

Figure B.6 – Network topology of the IEEE One Area RTS with two switching devices added.

B.2 Network parameters

Table B.1 – IEEE RTS 24 Bus Data.

Bus #	Bus type	Load (MW)	Load (MVAR)	GL	BL	Sub Area	Base kV	Zone #
101	2	108	22	0	0	11	138	11
102	2	97	20	0	0	11	138	12
103	1	180	37	0	0	11	138	11
104	1	74	15	0	0	11	138	11
105	1	71	14	0	0	11	138	11
106	1	136	28	0	1	11	138	12
107	2	125	25	0	0	11	138	12
108	1	171	35	0	0	11	138	12
109	1	175	36	0	0	11	138	13
110	1	195	40	0	0	11	138	13
111	1	0	0	0	0	11	230	13
112	1	0	0	0	0	11	230	13
113	3	265	54	0	0	12	230	14
114	2	194	39	0	0	12	230	16
115	2	317	64	0	0	12	230	16
116	2	100	20	0	0	12	230	16
117	1	0	0	0	0	12	230	17
118	2	333	68	0	0	12	230	17
119	1	181	37	0	0	12	230	15
120	1	128	26	0	0	12	230	15
121	2	0	0	0	0	12	230	17
122	2	0	0	0	0	12	230	17
123	2	0	0	0	0	12	230	15
124	1	0	0	0	0	12	230	16

Bus Type:

1 - Load Bus (no generation);
2 - Generator or plant bus;
3 - Swing Bus.

Load (MW): load real power.
Load (MVAR): load reactive power.
GL: real component of shunt admittance to ground.
BL: imaginary component of shunt admittance to ground.

Table B.2 – IEEE 24 Bus network generator parameters.

Bus #	Pgmax (MW)	Pgmin (MW)	Qgmax (MVAR)	Qgmin (MVAR)	Vs (p.u.)
101	192	0	80	-50	1,035
102	192	0	80	-50	1,035
107	300	0	180	0	1,025
113	591	0	240	0	1,020
114	0	0	200	-50	0,980
115	215	0	110	-50	1,014
116	155	0	80	-50	1,017
118	400	0	200	-50	1,050
121	400	0	200	-50	1,050
122	300	0	96	-60	1,050
123	660	0	310	-125	1,050

Pgmax & Pgmin: are limits of generator active power output.
Qgmax & Qgmin: are limits of generator reactive power output.
Vs: is the generator voltage set-point.

Table B.3 – IEEE 24 Bus network Branch and Transformer Data.

From Bus	To Bus	R (p.u.)	X (p.u.)	B (p.u.)	Con (MVA)	LTE (MVA)	STE (MVA)	Tr (p.u.)
101	102	0,003	0,014	0,461	175	193	200	0
101	103	0,055	0,211	0,057	175	208	220	0
101	105	0,022	0,085	0,023	175	208	220	0
102	104	0,033	0,127	0,034	175	208	220	0
102	106	0,05	0,192	0,052	175	208	220	0
103	109	0,031	0,119	0,032	175	208	220	0
103	124	0,002	0,084	0	400	510	600	1,015
104	109	0,027	0,104	0,028	175	208	220	0
105	110	0,023	0,088	0,024	175	208	220	0
106	110	0,014	0,061	2,459	175	193	200	0
107	108	0,016	0,061	0,017	175	208	220	0
108	109	0,043	0,165	0,045	175	208	220	0
108	110	0,043	0,165	0,045	175	208	220	0
109	111	0,002	0,084	0	400	510	600	1,03
109	112	0,002	0,084	0	400	510	600	1,03
110	111	0,002	0,084	0	400	510	600	1,015
110	112	0,002	0,084	0	400	510	600	1,015
111	113	0,006	0,048	0,1	500	600	625	0
111	114	0,005	0,042	0,088	500	600	625	0
112	113	0,006	0,048	0,1	500	600	625	0
112	123	0,012	0,097	0,203	500	600	625	0
113	123	0,011	0,087	0,182	500	600	625	0
114	116	0,005	0,059	0,082	500	600	625	0
115	116	0,002	0,017	0,036	500	600	625	0
115	121	0,006	0,049	0,103	500	600	625	0
115	121	0,006	0,049	0,103	500	600	625	0
115	124	0,007	0,052	0,109	500	600	625	0
116	117	0,003	0,026	0,055	500	600	625	0
116	119	0,003	0,023	0,049	500	600	625	0
117	118	0,002	0,014	0,03	500	600	625	0
117	122	0,014	0,105	0,221	500	600	625	0
118	121	0,003	0,026	0,055	500	600	625	0
118	121	0,003	0,026	0,055	500	600	625	0
119	120	0,005	0,04	0,083	500	600	625	0
119	120	0,005	0,04	0,083	500	600	625	0
120	123	0,003	0,022	0,046	500	600	625	0
120	123	0,003	0,022	0,046	500	600	625	0
121	122	0,009	0,068	0,142	500	600	625	0

$S_b = 100\ MVA$

Con: Continuous rating.

LTE: Long time emergency rating (24 hour).

STE: Short-time emergency rating (15 minute).
Tr: transformer off-nominal ratio.
Transformer branches are indicated by Tr ≠ 0.

ANNEX C

Submitted paper to publication in IEEE transactions

As a result of the advantages achieved by the developed strategy in this work, it was decided to write a paper where the main achievements and ideas of such strategy are presented. In this annex, one will do a brief and general presentation of the main topics that are focused in the referred paper. However it is important to underline that this paper is, at the moment, confidential, having been submitted for future publication in IEEE Transactions. Therefore only an initial version of two pages is reproduced below.

Reconstructing Missing Data in State Estimation with Autoencoders

Vladimiro Miranda[1], Cristiano Moreira[2] and Jorge Pereira[3]

***Abstract* – This paper presents a new solution to the problem of recomposing missing information at the SCADA of EMS/DMS (Energy/Distribution Management Systems), through the use of off-line trained autoencoders.**

***IndexTerms* – State Estimation, Autoencoders, Neural Networks, Energy Management Systems, Distribution Management Systems.**

I. Introduction

THIS paper presents a new approach to restoring missing data in the context of an Energy Management System or a Distribution Management System, or still of a Micro-Grid or Smart Grid. This approach is based on the properties of autoencoders, which are neural networks with a special architecture so that they have as many inputs as outputs – and trained to produce an output vector equal to the input vector.

When properly trained, an autoencoder stores in the weights of the neural network the information representing the real system whose data was used to train it. Any input pattern coherent with the real system will produce a similar output with negligible error. However, an input with some data inconsistent with the real system will generate a significant error between the input and the output vector. This property has been used to reconstruct missing sensor signals and will be used in this paper to reconstruct voltage values, as well as switching device status.

II. Autoencoders

Autoencoders are feedforward neural networks that are trained to reproduce the input space S in the output.

[1]V. Miranda (vmiranda@inescporto.pt), [2]C. Moreira (ee02245@fe.up.pt), and [3]J. Pereira (jpereira@inescport.pt) are with INESC Porto, Instituto de Engenharia de Sistemas e Computadores do Porto, Portugal.

V. Miranda and C. Moreira are also with FEUP, Faculty of Engineering of the University of Porto, Portugal. J. Pereira is also with FEP, Faculty of Economy of the University of Porto, Portugal.

In an autoencoder, the first half of the neural network approximates the function f that maps the input space to the space of compressed encoding S' while the second half approximates the inverse function f^{-1} (see Fig. 1). The quality of this encoding and decoding process depends on the quality of the training.

An autoenconder with non-linear activation functions and multiple layers chart the input space on a non-linear manifold in such a way that an approximate reconstruction is possible.

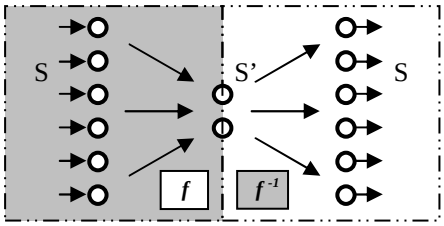

Fig. 1. An autoencoder neural network, with a *bottleneck* inner layer input and output layers of the same dimension.

Publications [1][2] describes some useful properties of autoencoders in restoring missing sensor signals. Three basic approaches are examined:

Alternating Linear Projections Onto Convex Sets (POCS): the first model uses alternating linear projections on the input and output space to converge to the assumed value of a missing sensor. If a sensor signal is missing, its input variable can be set to zero (or to a value outside the valid sensor range) and this will produce a mismatch between input and output. Iteratively reintroducing the output value in the input will converge to a value that minimizes the input-output error.

Unconstrained and constrained search: The second and third models require an optimization algorithm to minimize the input-output error. The unconstrained search controls convergence by the error on the missing signals; the constrained search controls the error on all the outputs of the autoencoder. In [3], for instance, a Genetic Algorithm is used to optimize the error function.

III. Autoencoder preparation in State Estimation

A. Creation of a database

As test system, the IEEE 24 BUS RTS network [4] was selected; then, minor modifications were introduced by defining a set of hypothetical measuring devices providing information on active and reactive power, as well as on voltage and angle, in buses and branches.

For all experiments to be conducted:

a. A large set of load and generation scenarios was prepared, by randomly varying those injections.
b. For each scenario, an AC power flow was run and its results added to a database.
c. A set S of locations and measured values was defined.
d. Training and test sets were defined, by extracting from the database vectors corresponding to the measurements defined.
e. An autoencoder was trained to learn the training set with generalization verified in the test set. Each autoencoder was composed of a single hidden layer, input neurons had linear activation functions, the remainder neurons had activation functions of the type 2arctan()/π and the performance criterion was MAE (Minimum Absolute Error).

B. Algorithmic strategy in restoring missing signals

We have developed and tested the three methods from [5] referred to above. For the *unconstrained and constrained search*, we selected as the optimization algorithm a meta-heuristic denoted EPSO [6][7].

We applied all three methods to a set of sampled cases from the database; some components of the input vector were specified as missing and initialized at a random value.

Using EPSO, the constrained search method has achieved the best results in recomposing missing signals, despite requiring more effort than POCS and *unconstrained search*.

The results in the following sections are, therefore, based on the application of *constrained search* with EPSO.

IV. Voltage pseudo-measurements

In this section (experiment A) we will show that reconstruction of the actual missing values of voltage and angles may be done with a high degree of accuracy.

An autoencoder has been trained using 10000 patterns and the input vector composed of 24 nodal voltages and 24 nodal angles. The middle layer had 40 neurons.

Then, 400 new vectors were generated with random missing signals (between 1 and 10 missing values), voltage and/or angle, and submitted to the autoencoder.

In voltage signals, the experiment led to an MAE of 1.95×10^{-3} p.u. relative to the missing values; in angle signals, the MAE was 2.52×10^{-1} degrees. Figure 5 represents a probability density function of the error distribution in voltage signal reconstruction calculated from the results using the Parzen windows method [8] with Gaussian kernels.

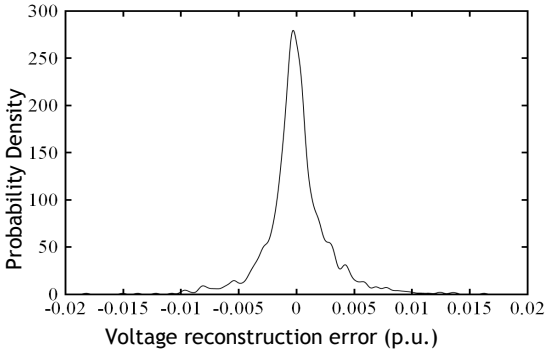

Fig. 5. Probability density function estimate of the error distribution for voltage reconstruction errors in all the 400 scenarios.

V. Network topology

A. Preparing the autoencoder

The database referred to in Section III A was enlarged with fields related to switches admitted in a set of 5 branches (1-2, 3-9, 11-13, 12-23 and 21-22) and enriched with the results from power flow analysis over different system topologies randomly selected (including the case of all switches closed).

Then, an autoencoder with 81 inputs and a middle layer of size 60 has been trained with a training set of 3000 patterns and a test set of 1000 patterns.

B. Discovering the network topology

Experiment B was conducted by sampling a set of 5000 scenarios with diverse active and reactive power measurements and switch status data. Some of the latter were then randomly removed. Five sets of trials have thus been defined, each with 1000 scenarios; in each set a fixed number of switches (1 to 5) with status unknown have been tested.

The results from experiment B are illustrated in Table I.

TABLE I – RESULTS FROM APPLYING AN AUTOENCODER TO RECOMPOSE MISSING SWITCH STATUS SIGNALS IN THE IEEE 24 BUS RTS

Missing signals	Scenarios tested	Correct network topologies	Correct signal reconstructions	Wrong signal reconstructions
1	1000	996	996	4
2	1000	990	1990	10
3	1000	993	2993	7
4	1000	992	3992	8
5	1000	991	4990	10

VI. CONCLUSIONS

The results obtained from a large number of tests give a clear indication that autoencoders do indeed allow the correct reconstruction of the missing signals. Furthermore, the underlying network topology was discovered in a vast majority of cases, even if a considerable number of switch status information was missing.

The confirmed properties of autoencoders allow their use not only in the reconstruction of missing signals but also in the generation of new *virtual measurements* in network locations where no measuring device is installed.

REFERENCES

[1] B. B. Thompson, R.J. Marks and M.A. El-Sharkawi, “On the Contractive Nature of Autoencoders: Application to Missing Sensor Restoration”, Proceedings of the International Joint Conference on Neural Networks, Vol. 4, pp. 3011- 3016, July 2003

[2] Narayanan, S., R.J. Marks I1, II. L. Vian, 1.1. Choi, M.A. El-Sharkawi & B. B. Thompson, "Set Constraint Discovery: Missing Sensor Data Restoration Using Auto-Associative Regression Machines", Proceedings of the Intemational Joint Conference on Neural Networks, Honolulu (Hawaii), USA, pp. 2872-2877, May12-17, 2002.

[3] Abdella, M. and Marwala, T., “The use of genetic algorithms and neural networks to approximate missing data in databases”, *Comput. Inf.*, 2005, 24, 577–589.

[4] IEEE RTS Task Force of APM Subcommittee, "IEEE Reliability Test System', IEEE PAS, Vol-98, No. 6, pp 2047-2054, Nov/Dec. 1979.

[5] G. E. Hinton and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks”, *Science*, Vol. 313, no. 5786, pp. 504 – 507, July 2006

[6] V. Miranda and N. Fonseca, “EPSO – Best-of-two-worlds metaheuristic applied to power system problems,” *Proceedings of WCCI 2002 - World Congress on Computational Intelligence - CEC - Conference on Evolutionary Computing*, Honolulu, Hawaii, USA, May 2002.

[7] Vladimiro Miranda, Hrvoje Keko and Álvaro Jaramillo Duque, “Stochastic Star Communication Topology in Evolutionary Particle Swarms (EPSO)”, *International Journal of Computational Intelligence Research*, vol. 4, no.2, 2008

[8] E. Parzen, “On the estimation of a probability density function and the mode”, *Annals Math Statistics*, vol 33, 1962