ANALYSIS OF THE VOLTAGE STABILITY PROBLEM IN ELECTRIC POWER SYSTEMS USING ARTIFICIAL NEURAL NETWORKS

HERNÁN PRIETO SCHMIDT
BSc (Eng), MSc (Eng)

Thesis presented in the University of London
for the degree of Doctor of Philosophy

Department of Electronic Engineering
Queen Mary and Westfield College
University of London

September 1994

ABSTRACT

The voltage stability problem in electric power systems is concerned with the analysis of events and mechanisms that can lead a system into inadmissible operating conditions from the voltage viewpoint. In the worst case, total collapse of the system may result, with disastrous consequences for both electricity utilities and customers. The analysis of this problem has become an important area of research over the past decade due to some instances of voltage collapse that have occurred in electric systems throughout the world.

This work addresses the voltage stability problem within the framework of artificial neural networks. Although the field of neural networks was established during the late 1940s, only in the past few years has it experienced rapid development. The neural network approach offers some potential advantages to the solution of problems for which an analytical solution is difficult. Also, efficient and accurate computation may be achieved through neural networks.

The first contribution of this work refers to the development of an artificial neural network capable of computing a static voltage stability index, which provides information on the stability of a given operating state in the power system. This analytical tool was implemented as a self-contained computational system which exhibited good accuracy and extremely low processing times when applied to some study cases.

Dynamic characteristics of the electrical system in the voltage stability problem are very important. Therefore, in a second stage of the present work, the scope of the research was extended so as to take into account these new aspects. Another neural network-based computational system was developed and implemented with the purpose of providing some information on the behaviour of the electrical system in the immediate future.

Examples and case studies are presented throughout the thesis in order to illustrate the most relevant aspects of both artificial neural networks and the computational models developed. A general discussion summarises the main contributions of the present work and topics for further research are outlined.

ACKNOWLEDGEMENTS

The author wishes to express his deepest gratitude to his supervisor, Dr. R. N. Adams, for suggesting the topic for study and for his constant advice and support throughout the development of the present work.

The author is also grateful to *CNPq -Conselho Nacional de Desenvolvimento Científico e Tecnológico-* and to *EPUSP -Escola Politécnica da Universidade de São Paulo-*, both Brazilian institutions, for the financial support and the release from academic activities during the present course.

Special appreciation is expressed to Nelson Kagan, long-standing friend whose comments and opinions were most valuable.

The support from friends and colleagues at EPUSP and Queen Mary and Westfield College is acknowledged, with particular reference to Prof. E. J. Robba for his incentive in many occasions throughout the career of the author.

Kevin O'Brien is also acknowledged for the careful revision of the text.

Finally, but by no means least, the author would like to thank Viviane, for everything.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 - General considerations

An electric power system consists of a large set of generators, transmission lines, transformers, protective devices, and other associated equipment. The main purpose of a power system is to produce, transport and distribute energy in the form of electricity. The end users can be connected to the system at various voltage levels (e.g., subtransmission, primary distribution, and secondary distribution) and they dictate, through their continuously changing demand, the necessary generation requirements. Therefore, system operators must be able to operate the system in a myriad of different conditions, and they must do so in the most economic and reliable way. For example, under normal conditions cost minimisation can be the main objective to be pursued, and this can be achieved through the preferential operation of those generating units that are most economical. Under emergency conditions, on the other hand, the main objective can be switched to operational reliability, when more equipment can be brought into operation so as to increase available reserves.

There exist many operational problems that system operators are faced with. For example, the classical Transient Stability Problem (TSP) is concerned with whether the system will remain stable after a major contingency occurs in the system, such as generator outage or three-phase short-circuit. In this work, the focus will be placed on the so-called **Voltage Stability Problem** (simply referred to as VSP throughout this work), in which situations occur where system operators are unable to keep the voltage profile across the system within adequate operational limits.

The VSP is a relatively new problem and is not yet fully understood. Its long-term causes lie in the imbalance between the growth in demand and the system expansion. The latter may not accompany the former, for example, due to unpredicted economic constraints. This can cause the system to operate close to its stability limits. If a minor contingency occurs in an already stressed system, stability may be lost, leading to the most critical outcome of a voltage instability process: the so-called *voltage collapse*. After a voltage collapse occurs, the system becomes dismantled due

to the widespread operation of protective devices.

The classical TSP and the VSP share, to a large extent, some important features. But, while the TSP is "fast" (lasting no more than several seconds), the VSP can be considered as a "slow" phenomenon, with a time frame of minutes or even hours. Also, the TSP is more concerned with the relationship between active power and system frequency, while the VSP is associated with reactive power patterns in the system.

The VSP is a multi-faceted problem in the sense that it can be analysed through many different viewpoints, such as assessment of proximity to voltage collapse, study of sensitivities with respect to changing parameters, and study of suitable corrective actions. In the present work the focus will be placed on the issue of the assessment of proximity to voltage collapse, and this will be done within the framework of **Artificial Neural Networks** (simply referred to as ANN throughout this work). ANNs belong to the field of Artificial Intelligence, and will be briefly introduced in the next section.

1.2 - Artificial Intelligence and power systems

Artificial Intelligence is the branch of Computer Science that deals with computational systems which present features normally associated with human intelligence. These features include the ability to learn, to reason, to understand, to solve problems, etc.

The field of *Expert Systems* is an important part of Artificial Intelligence. Its main purpose is the construction of automatic systems that resemble both the reasoning process and the solutions proposed by human beings who are experts in specific areas of human knowledge.

The application of expert systems in power systems has already achieved a mature stage of development. Numerous successful applications are reported in the technical literature [1]. In general terms, two major components can be identified in a typical expert system: the knowledge base and the reasoning mechanism. The knowledge base is the location where all knowledge (facts) about a specific domain is stored. The reasoning mechanism is the process used to inter-relate the facts in the knowledge base so as to produce useful answers for given problems. The set of functions and subroutines responsible for the reasoning mechanism is normally referred to as *inference engine*. One of the most complex and difficult tasks in the making of an expert system is the transfer of knowledge from the human expert to the automatic system. There are important issues related to this process that still remain unresolved. For instance, what "language" does the human brain utilise for interconnecting all the internal processes when solving a problem? How can this information be translated into a known computing language?

Acknowledging the questions above, the ANN approach attempts to produce the same answers but following a different track. Instead of converting all the known facts about a specific problem into machine language, which sometimes proves to be intractable, the ANN approach relies on learning from past experience and on the construction of internal representations from adequate sample patterns.

The field of ANN (also known as *connectionist* or *learning-by-example* approach) also belongs to the general subject of Artificial Intelligence. Although around since the late 1940s, only in the past few years has it experienced rapid development, as will be seen in this work.

Applications of ANN in power systems are not as developed as expert system applications, which makes the former a very interesting research subject. This is the path followed in the present work.

1.3 - Organisation of the thesis

The text is intended to be self-contained throughout the thesis. Cross-references are provided in order to facilitate the search of specific topics between different chapters.

Chapter 2 introduces the electrical problem under consideration, namely the VSP. The problem is described from a qualitative viewpoint, along with an analysis of its main causes and most common corrective actions practised today by system operators. An example of a simulated voltage collapse is included with the purpose of illustrating some mechanisms that can be observed during a voltage instability process. The VSP shares some important features with the classical TSP, and for this reason a comparative analysis stressing the similarity between the two problems is included. At the end of this chapter it will be possible to specify the scope of the present research work in greater detail.

Chapter 3 contains the main literature review that was carried out during the whole research program. It is divided in two main sections: the first one contains a comprehensive survey of the approaches to the VSP that have been proposed so far, and the second section presents the current state-of-the-art in the applications of ANN to electric power systems. The section on the VSP is intended to be self-contained in the sense that most of the necessary concepts, required to understand the current approaches, are discussed in some level of detail. The same is valid for the ANN section, because a number of different problems (such as load forecast, harmonic analysis, etc.) had to be introduced before actually starting the description of the ANN framework.

Chapter 4 also presents the results of a literature survey, this time on the vast field of ANN. It begins with a discussion on some important issues such as the correspondence between biological neural networks and their artificial counterparts. The training of an ANN is a crucial step, and is also discussed in some detail. The remainder of the chapter is dedicated to the description of the main architectures of ANNs (five in total). All models were implemented using the *C* language, and their operation is always described through an example at the end of the corresponding section. During the implementation of the *Backpropagation* training algorithm (for use on *Multi-Layer Perceptron* networks), a modification to the basic procedure was devised and implemented. This modification improved the training and is also described.

Chapter 5 contains the first set of results of the present research. The VSP was analysed through a static approach, using the mathematical framework of *Singular Value Decomposition.* The voltage stability index was chosen to be the Minimum Singular Value of a Jacobian-related matrix, and a Multi-Layer Perceptron network was trained so as to compute this index from the knowledge of some electrical input variables. A self-contained computational system was developed for this specific problem. The system was written using the *C* language and its main components include a load-flow program, a Multi-Layer Perceptron/Backpropagation program, and an interface program (acting between the load-flow and the neural network program) for automatically creating complete training sets. The most relevant methodology aspects concerning this computational system are described in detail, as well as its utilisation in three different case studies.

Chapter 6 represents the natural extension of the work developed in Chapter 5. In that chapter, the static approach did not allow for either the dynamic nature or the non-linear aspects of the VSP. Also, the results obtained in Chapter 5 were encouraging enough so as to consider the application of a Multi-Layer Perceptron in conjunction with a more sophisticated methodology for the VSP. The new framework that was chosen for the VSP is the *Dynamic Simulation* approach, which not only removes the constraints of the static approach but also allows a detailed modelling of the most relevant components in the power system. Another major reason for considering the coupling between the dynamic simulation and the Multi-Layer Perceptron was precisely the main drawback of the simulation technique: the fact that it is very CPU-intensive. The central idea was to remove the main disadvantage of the simulation technique (through the use of an ANN) while retaining its most valuable features. A new computational system was developed, also using the *C* language. This system consists of a load-flow program (the same as in Chapter 5), a dynamic simulation program, two interface programs, and a Multi-Layer Perceptron program (again, the same as in Chapter 5). The two interface programs correspond to two different techniques that were devised and developed for mapping the VSP onto the ANN problem language. The computational system and the results obtained in some case studies are described in detail.

Chapter 7 presents the conclusions of the thesis. First, a general discussion critically summarises the research work. Then the major contributions of the thesis are made explicit. Finally,

topics for further improvement and development are outlined.

CHAPTER 2

THE VOLTAGE STABILITY PROBLEM

2.1 - Introduction

This chapter introduces the Voltage Stability Problem, which is the electrical problem under consideration in the present work.

Initially this problem will be described from a qualitative viewpoint. Some considerations on the causes and possible corrective actions will be made. An example of a simulated voltage collapse is then included with the purpose of illustrating some important physical aspects of the problem.

The VSP shares, to a considerable extent, the same background as the Transient Stability Problem (TSP). A comparative analysis between the two problems can be very useful for improving the VSP formulation and understanding. This is particularly important here because, in the present work, one of the approaches that was considered is the simulation technique, which is directly derived from the classical TSP analysis (see Chapter 6). A comparative analysis between the VSP and the TSP follows in this chapter.

It will become clear that the VSP is a complex problem that can be analysed from many different viewpoints. Thus, at the end of this chapter it will be possible to further specify the objectives that will be pursued throughout the present work.

2.2 - Description, causes and preventive controls for the VSP

Although the probability of occurrence of voltage collapse is very low, the consequences of such a phenomenon are so serious that they justify the great efforts that have been made in the investigation of the problem. Practical instances of voltage collapse have been observed in France,

Sweden, Japan, USA and many other countries [2-6].

The VSP can manifest itself in various ways, but, in general, two distinct periods can be observed. First, there is a slow decline in the voltage of buses within a specific area. During this period, no clear indicators are available to alert the operators that the system may be undergoing a voltage collapse process. This first stage can last up to several minutes, and it has been observed that some automatic voltage-controlling devices, such as reactive power compensators and on-load tap changers of transformers, can even worsen the situation. The second period is fast (lasting no more than several seconds) and represents the spreading out of the local deteriorating conditions throughout the remainder of the system.

The causes of the VSP are also varied. As a long-term cause, it is possible to mention the fact that, due to economic and operational constraints, the expansion of the electrical system has not accompanied the growth in demand. Thus, interconnected power systems are frequently operating closer than ever to their maximum admissible limits, very often in conditions for which they were not designed. Immediate causes include line outages and random load variations. The outage of a minor transmission line in an already stressed system, for example, may be sufficient for triggering a cascade-like series of events which ultimately leads to the collapse of the system. A voltage collapse process may also start due to load variations only. These facts distinguish the VSP from the classical TSP, where the focus is placed on the balance of active power after a critical contingency. This comparison between the VSP and the TSP will be further developed later in this chapter.

The VSP has also been associated with inadequate reactive support. If this condition is true, at least at a local level, then reactive power must be transported over the network from somewhere else. High flows of reactive power induce high reactive losses and low voltages, and can make generators reach their maximum reactive power generation. Collapse may occur when imbalance between supply and load cannot be corrected by a limited voltage decrease.

In some practical instances of voltage collapse it was observed that bus angles and system frequency remained nearly constant during the voltage decay, and in some other cases they accompanied that decay. In the case of disturbance-induced collapses, it has also been observed that the system had already settled down to a stable state with lower voltages, but the stability was eventually lost due to the action of load controllers and generator reactive power limiters.

System loads play an essential role in a voltage instability process, especially those loads known as *stiff loads*. This term designates the loads which possess automatic control devices, which tend to keep the power absorbed by the load at a constant level regardless of the voltage at which the load operates. The action of these controlling devices can be very detrimental to a system at a time when some level of load alleviation may be required in order to allow the system to

reach a new stable operating point. Stiff loads usually exhibit dynamic behaviour with time constants in the order of minutes, and therefore they can be considered as "slow" devices. Induction motors, on the other hand, have a characteristic response much faster than stiff loads, but their contribution to a voltage instability process is also important due to their behaviour at low operating voltages, because in that situation they absorb large quantities of reactive power and can even stall. As a final remark on the load problem, it can be said that proper load modelling is an essential issue in voltage stability studies, and yet is a topic which is very controversial and far from being mature. This will be further developed in Chapter 3 (Literature Review).

With respect to the controls available to improve the voltage stability of a system, perhaps the most important are tap changers, generator reactive overload, and undervoltage load shedding. It should be pointed out that the use of all these resources for improving the voltage stability has not yet been fully established. Tap changers can sometimes contribute to a voltage instability process and sometimes can improve the stability, so a thorough understanding of their role is required. Generator reactive overload can make the difference between voltage stability and voltage collapse, especially if the overload is not large and is to be maintained for short periods of time. Much research is needed in this field, especially in order to establish quantitative results. Finally, undervoltage load shedding also appears to be a promising technique. More research is also needed so as to establish the amount, location and duration of preventive load shedding.

2.3 - Example of a simulated voltage collapse

For the purpose of illustrating some physical aspects of the VSP, an example of voltage collapse is presented in this section. It should be noted that this example was extracted from reference [6].

Figure 2.1 shows a simple, heavily loaded power system, which will be used in the following explanation.

In this case, the instability process is triggered by the loss of a small capacitor bank at bus 5, on the 115-kV level. After this outage the deficit in reactive power is now transported over the network. This increases the transmission losses and reduces the voltage in the load area. This voltage reduction also reduces the reactive power generated by the remaining capacitors and line chargings, thus further aggravating the reactive power deficit. Immediately after the outage, the active power flow on the line decreases because the active demand of the residential load also decreases due to the new, lower voltage at bus 6. The industrial load, mostly made up of induction motors, does not change significantly during this period.

Supposing that the residential substation is equipped with an automatic on-load tap changer

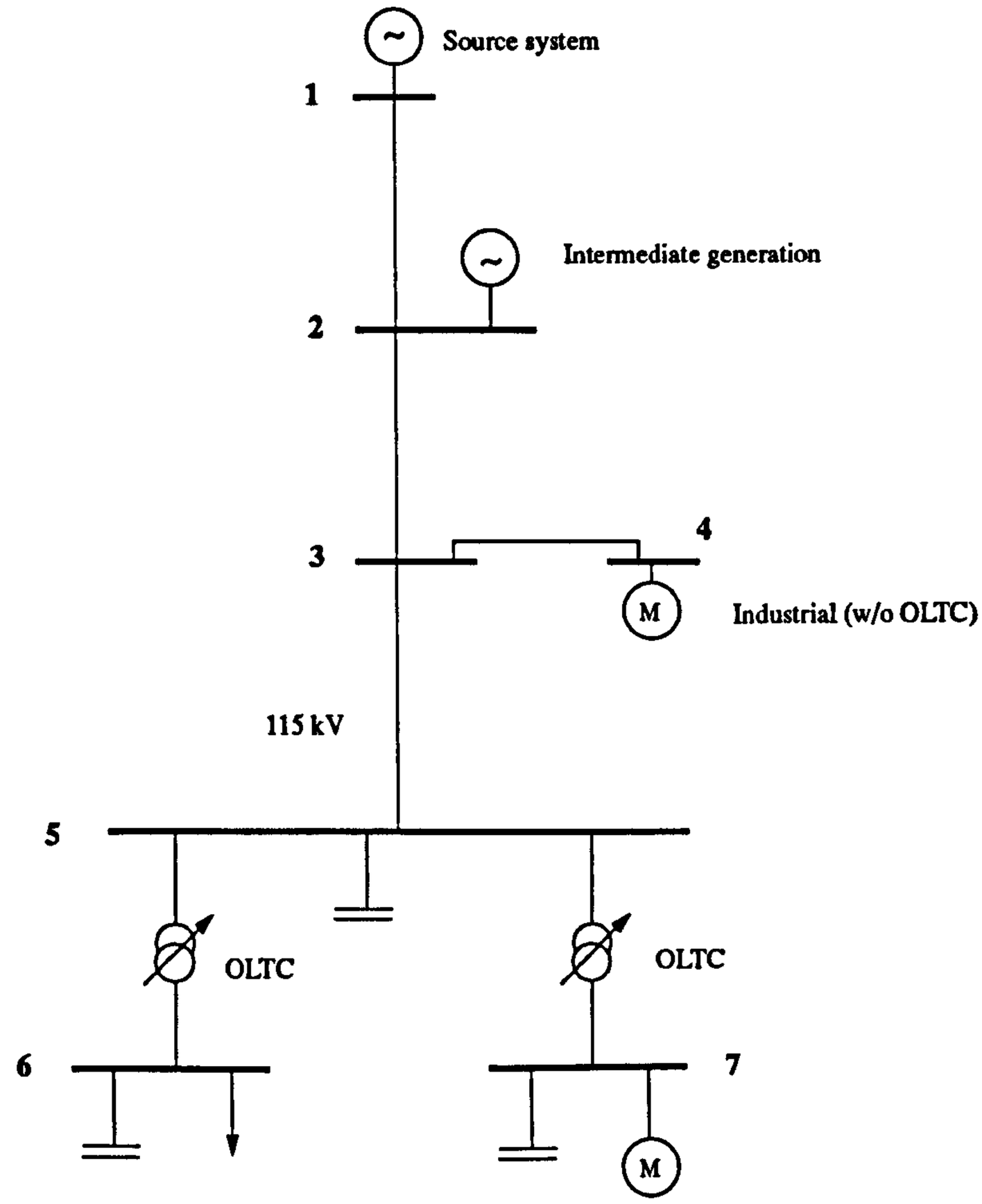


Figure 2.1 - Simple power system for example of voltage collapse

(OLTC), after some time this OLTC will start restoring the voltage of the residential load. This has the effect of increasing the residential active power (the reactive power is not much affected due to the high load factor of the residential load). This increase on active power will be transported over the 115-kV line (branch 3-5). With more active power flowing on the line, the voltage drop across the line, required to supply the initial reactive power deficit, also increases. This reduces the voltage at bus 5 and increases the reactive losses on the 115-kV line, making the deficit of reactive power even more severe. As the residential OLTC continues to compensate for its low primary voltage, the situation becomes more and more critical. During this process, reactive power reserves at the intermediate generator (bus 2) may become exhausted, thus propagating the reactive power deficit to remote generators further up the 115-kV line.

With the tap at its limit, the residential OLTC can maintain the residential voltage at 1.00 pu

with its primary voltage at 0.90 pu. If the 115-kV line cannot maintain 0.90 pu at bus 5, a new, stable operating point may be reached, with residential voltage in the range 0.95 - 1.00 pu and voltage at bus 5 in the range of 0.85 - 0.90 pu. With residential voltages below 1.00 pu, local load controllers will start to restore loads to their rated values. This has the effect of transforming constant impedance loads into constant power.

If, during this process, the voltage at bus 7 (industrial loads) reaches a value of around 0.90 pu or less, it is quite possible for a motor to stall, thus reducing the voltage in the area. Voltage collapse at this point is likely to occur.

Protective devices of motors actuate when the motor voltage is in the range of 0.50 - 0.70 pu. Also, fluorescent lamps may be extinguished at these low voltage levels. If a large amount of load is removed in this way, voltages will rise. The voltages in the residential area will be particularly high due to the tap position of the OLTC, which will be lowered only after some time has elapsed. In the following minutes, industrial load will be restored, as well as air-conditioning equipment, for instance. This will have the effect of dropping voltages again. The whole process of voltage instability may be repeated at regular intervals over the next thirty minutes or so.

2.4 - Stability in electric power systems

Stability in electric power systems is usually classified into *steady-state stability* and *transient stability.*

Steady-state stability seeks to determine the stability limit of a system when "small" changes are made in the system (for example, in the form of load variations). A steady-state stability study can be carried out with a conventional load-flow program, and its main results correspond to the maximum power transferable across transmission lines and transformers, so as not to violate thermal limits as well as to guarantee adequate voltage levels.

Transient stability studies, on the other hand, address the question of whether a system will preserve stability in the face of a structural change such as line or generator outage. Stability implies, in this case, that no generator will lose synchronism with respect to the rest of the system. Traditionally, transient stability studies have been carried out using dynamic simulation programs, and, more recently, through the use of so-called *direct methods.* Direct methods assess the stability without actually simulating the temporal behaviour of the system, in an attempt to overcome the main drawback of classical simulation methods, which is their high computational cost.

Both steady-state and transient stability studies are classically associated with patterns of

active power and frequency in the electric system because the main concern is the question of loss of synchronism. Also, the decoupling of active power and angle from reactive power and voltage allows some degree of independence between angle stability and voltage/reactive power problems. For instance, in a classical stability study, voltages are assumed to be close to the value 1.0 pu. But in the more modern phenomenon of voltage instability, hypotheses like this are no longer true. Another major difference between classical stability and voltage stability is the time scale. Secondary voltage controls, such as on-load tap changers, are much slower than primary controls associated with generators, namely the excitation and governor subsystems. Thus, voltage instability problems can be seen as "slow" dynamic processes, as opposed to the dynamics of classical stability.

There is not as yet a well established methodology for studying voltage stability problems. This will be discussed in detail in Chapter 3, where the main approaches that were found in the literature will be presented. Every approach has its own advantages and disadvantages. The main compromise among all methodologies lies between simulation accuracy and the computational cost involved in the analysis, as it occurs in many other research areas.

2.5 - Scope of the present work

The first two questions that arise in the study of voltage instability in power systems refer to the proximity and to the mechanisms of voltage collapse. The issue of proximity addresses the establishment of an indicator of how close to collapse a system is operating. An ideal indicator should fulfil the following basic requirements:

- *meaning*: the information delivered by the indicator should be physically related to electrical magnitudes of the problem (voltages, reactive power injections, etc.);
- *reliability*: the indicator should be reliable so as to avoid misclassification of operating states (either "misses" or "false alarms");
- *efficiency*: the indicator should be evaluated inexpensively (especially for on-line applications).

With regard to the issue of voltage collapse mechanisms, it is essential to identify the main contributing factors in a voltage collapse process, in order to (*i*) understand the phenomenon, and (*ii*) devise corrective actions to be taken, preferably within the time frame of on-line applications.

As will be seen in Chapter 3, the current approaches to the VSP can be broadly classified according to their static or dynamic nature. The VSP is dynamic in nature, but it can be analysed through a static viewpoint since the dynamic processes are slow. Within a static approach, use is

made of the load-flow algebraic equations describing the power system. The Jacobian matrix, used for solving the load-flow equations through the Newton-Raphson method, can provide some information on proximity to voltage collapse, but its main merit is, perhaps, the information it can deliver about sensitivity of an operating point with respect to varying parameters such as system load. The Jacobian matrix represents a linearisation around an operating point, and therefore the validity of the results obtained through it is limited to the validity of the linearisation itself. In this way, the non-linear aspects of the VSP cannot be taken into account.

Dynamic approaches, on the other hand, can take into account these non-linear aspects. They also allow the representation of those system components whuch present dynamic behaviour. These components are modelled using differential equations in the time domain. Perhaps the most developed dynamic approach is the simulation technique, which is strongly based upon the simulation methods of the classical transient stability problem. The dynamic simulation approach allows a detailed modelling of the system and its components, but it is very time-consuming. Also, it does not readily provide sensitivity information, which is important when considering corrective actions.

In the present work, the emphasis will be put on the issue of a voltage stability index (issue of proximity), and this goal will be pursued within the framework of both static and dynamic viewpoints (cf. chapters 5 and 6).

2.6 - Summary

It has been seen that nowadays the VSP is becoming an important concern in electric power systems, at both planning and operational levels. Voltage collapse is the most critical consequence of a voltage instability process, and it means the system disintegration due to the action of protective devices. The consequences of a voltage collapse are so serious that they offset its probability of occurrence, which is usually very low. Electric utilities need accurate indicators of proximity to voltage collapse so they can operate the system with maximum economy and reliability.

The VSP is in itself a very complex problem, and its dynamic mechanisms are not yet fully understood. This makes the establishment of a detailed problem formulation quite difficult. On the other hand, these facts allow the problem to be analysed from many different viewpoints. In the present work, emphasis will be put on the issue of proximity to voltage collapse. This means the definition and computation of a scalar, global indicator associated with every operating state in the electric system. Two different indices will be considered in this work, and both will be computed within the framework of Artificial Neural Networks.

CHAPTER 3

LITERATURE REVIEW

3.1 - Introduction

This chapter describes the results of the literature survey that was carried out during the development of the present research work. This survey was conducted on two separate subjects: the voltage stability problem and the application of artificial neural networks in Power Systems. Both subjects will be discussed in the following sections. At the end of the chapter, a summary with general conclusions is presented.

3.2 - The voltage stability problem

3.2.1 - Introduction

The relatively high number of different approaches to the voltage stability problem that were found in the literature is in itself an indication that this field is still in an early stage of development. When analysing these various approaches, the first criterion which naturally emerges for classification is their static or dynamic nature. In static analysis, steady-state operation of the system is assumed, and therefore only algebraic equations are considered. On the other hand, in dynamic analysis, transient operation of the system is considered, and thus either pure-differential or differential-algebraic equations are established.

Table 3.1 presents the main approaches to the voltage stability problem, which will be discussed in the following sub-sections.

3.2.2 - Steady-state stability limit

Approach	
Nature	**Procedure**
Static	Steady-state stability limit
	Load-flow analysis
	Optimisation techniques
	Sensitivity methods
Dynamic	*M*-matrices
	Transient *PV* curves
	Energy methods
	Bifurcation theory
	Singular perturbation theory
	Simulation approach
	Device analysis

Table 3.1 - Main approaches to the voltage stability problem

The analysis of the steady-state stability limit of a power system addresses the maximum power that the system is able to transfer to its loads under normal operating conditions. Figure 3.1 shows a simple system where a load is fed by a generator through a transmission line.

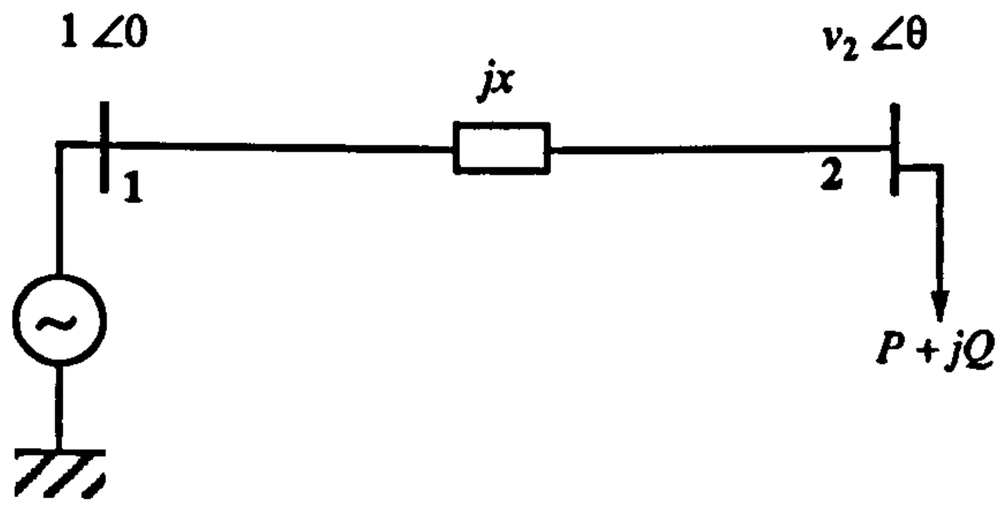


Figure 3.1 - 2-bus power system

For the system of Figure 3.1, it can be shown [2] that the relationship between the load voltage v_2 and the power delivered to the load is given by the family of curves of Figure 3.2, also known as "*PV* curves" or "nose curves".

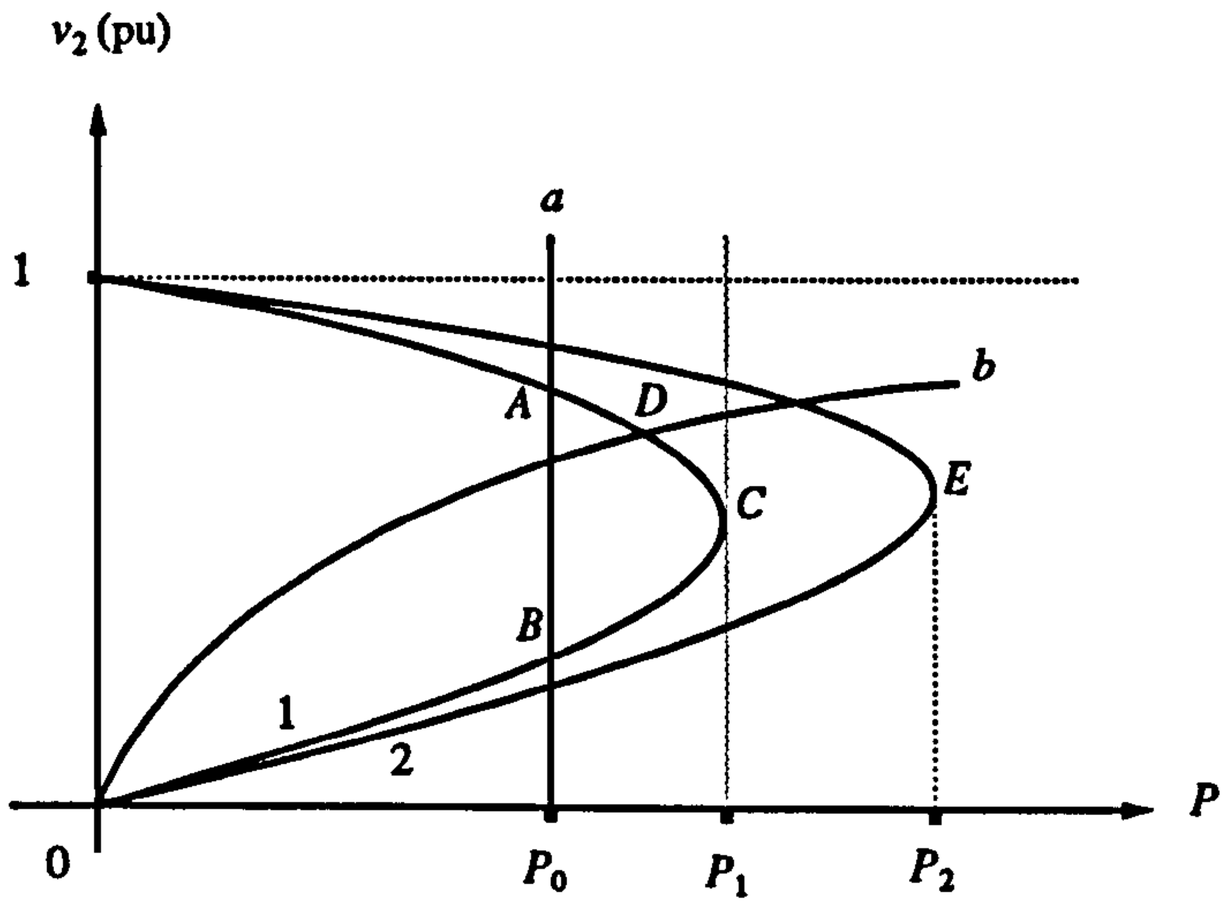


Figure 3.2 - *PV* curves

In Figure 3.2, curve *a* (vertical straight line) represents the behaviour of a constant-power load and curve *b* represents a constant-impedance load. Curve 1 (small nose) describes the behaviour of the system of Figure 3.1 and Curve 2 corresponds to the same system with a reactive power compensator connected to bus 2. Curve 1 will be used in the following explanation. It can be seen that when the load is modelled as a constant-impedance device, only one stable solution will always exist (point *D*), regardless of the magnitude of the load and neglecting the trivial case of null load. On the other hand, when the load is modelled as constant-power, either zero, one or two solutions may exist, depending on the magnitude of the load. For a load with active power P_0, two solutions will exist (points *A* and *B*). Point *A* is called the *high-voltage* solution and represents the normal operating point; point *B* is the *low-voltage* solution (see reference [7] for a thorough discussion on the stability of both solutions). Increasing the load up to the value P_1, both solutions become the same (point *C*). This is the maximum power deliverable by the system, and for this reason the value P_1 is called the *steady-state stability limit*. With this system, it is impossible to feed a constant-power load of which the demand is above the value P_1. It is interesting to notice that adequate reactive power compensation at bus 2 can increase the steady-state stability limit (point *E* in curve 2). An important conclusion of this analysis is that proper load modelling is essential in the study of the steady-state stability limit.

The next step is concerned with how the previous result can be generalised for an *n*-bus power system. Venikov et al. [8] were the first to suggest the analysis of the singularity of the Jacobian matrix of the load-flow equations for evaluating the steady-state stability limit:

$$\begin{bmatrix}\Delta P\\ \Delta Q\end{bmatrix} = J \cdot \begin{bmatrix}\Delta\theta\\ \Delta V\end{bmatrix} = \begin{bmatrix}F_\theta & F_V\\ G_\theta & G_V\end{bmatrix} \cdot \begin{bmatrix}\Delta\theta\\ \Delta V\end{bmatrix} \tag{3.1}$$

When matrix J becomes singular due to incremental load changes, Eq. (3.1) cannot be solved and the steady-state stability limit is reached. The singularity of the Jacobian matrix of Eq. (3.1) implies that at least one of its eigenvalues is zero. In accordance with this fact, some authors concentrated on the analysis of either the eigenvalues or the singular values of the Jacobian matrix. Tiranuchit et al. [9, 10] proposed the use of the Minimum Singular Value of the Jacobian matrix as a measure of proximity to the steady-state stability limit.

The Singular Value Decomposition (SVD) is a mathematical tool based on the use of orthogonal matrices. Given an n x n real matrix M, its SVD is defined as follows [11, 12]:

$$M = U.S.V^t$$

where U and V are n x n orthogonal matrices and S is an n x n diagonal matrix of which entries σ_1, σ_2, ... , σ_n are non-negative and are called singular values of M. The singular values of M are the positive square roots of the eigenvalues of either matrices $M^t.M$ or $M.(M^t)$, which are symmetric, non-negative definite matrices and hence possess real, non-negative eigenvalues. Usually the singular values are arranged so that $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$, and therefore the element σ_n is called the Minimum Singular Value (MSV). Let r be the rank of M; when $r = n$, all singular values will be strictly positive and the remaining (n-r) singular values will be zero. The MSV measures the l_2-norm distance between M and the set of all the rank-deficient matrices [13]. This, and the fact that the SVD is a well-conditioned decomposition, make the SVD especially attractive for analysing problems related to the singularity of the Jacobian matrix. Another advantage of the SVD lies in the information that can be extracted from the associated singular vectors: the right singular vector v_n, associated with σ_n, indicates sensitive voltages and angles, whereas the left singular vector u_n, associated with σ_n, indicates the most sensitive direction for changes of active and reactive power injections.

Tiranuchit et al. [10] showed that the MSV is more sensitive to load changes than bus voltages in the proximity of singularity, which is an important characteristic of this indicator. They also proposed an optimisation procedure for determining the generation profile which yields the largest MSV.

One disadvantage of using the SVD is that the computing time of a full decomposition increases as the cube of the size of the matrix under consideration. Löf et al. [14] proposed two different algorithms for fast evaluation of the SVD. They applied their technique to two different matrices: (*i*) the complete Jacobian matrix J of Eq. (4.1), and (*ii*) the matrix G_S defined by:

$$G_S = G_V - G_\theta \, . F_\theta^{-1} \, . F_V$$

The matrix G_S relates voltage changes to reactive power changes when there are no changes in the active power ($\Delta Q = G_S.\Delta V$ in Eq. (3.1) for $\Delta P = 0$). The authors also interpreted the meaning of either left and right singular vectors in terms of sensitivity of the MSV with respect to changes in the electrical parameters (voltages, angles and power injections). It should be pointed out that the MSV approach is an important part of the present work and will be further developed in Chapter 5.

Kessel and Glavitsch [15] developed a voltage stability indicator using linear steady-state analysis of the power system. The computation of the indicator ("*L*-index") is straightforward and is based on the knowledge of the state variables of the system (voltage magnitudes and angles) and the nodal admittance matrix. It is shown that theoretical values for this index range between 0 and 1, the former corresponding to the no-load condition and the latter corresponding to the collapse of the system. Although this is a local index (associated with each bus in the system), the paper proposes the definition of a global index as the largest (worst) local index computed throughout the system. The paper also shows how the index can be updated when changes in either nodal powers or topology occur.

In a comprehensive paper, Gao et al. [16] proposed the application of modal analysis (eigenanalysis) for evaluating the steady-state stability limit of a power system. As with the SVD, computing the minimum eigenvalue indicates the distance of the current operating point from the stability limit. The authors consider the m least eigenvalues, making it possible to analyse the m least stable modes instead of the least one alone. Using the associated left and right eigenvectors, bus participation factors are derived. These participation factors allow the identification of areas close to voltage instability. Another interesting idea of the paper is the use of the "snapshot" approach, which basically consists of various static analyses carried out at different instants in order to assess the temporal behaviour of the system. This allows overriding a detailed dynamic analysis in the time domain. The authors show an example in which snapshots were taken immediately after a contingency, then after fast controls had actuated, then after the action of slow controls, and finally after the action of human operators.

Finally, Sauer and Pai [17] presented a thorough analysis in which they compared the system matrix of a dynamic power system model to the standard load-flow Jacobian matrix. They showed how the load-flow Jacobian matrix appears within the dynamic system matrix and established the circumstances under which the analysis of the load-flow Jacobian matrix provides useful information about the dynamic system. They concluded that these circumstances correspond to rather drastic assumptions about the synchronous machines and their control systems, so that the singularity of the load-flow Jacobian matrix should be considered an optimistic upper bound on the maximum loadability of the system.

3.2.3 - Load-flow analysis

As seen in the previous sub-section, the steady-state stability limit of a power system is related to the singularity of the Jacobian matrix *J*. This has important implications in the classical load-flow problem, because the Newton-Raphson (NR) method -by far the most popular technique for solving the problem- requires the inversion of matrix *J* in order to evaluate corrections for angles and voltage magnitudes. Hence, the conventional NR method is inappropriate for the study of the power system in the vicinity of a steady-state stability limit because convergence problems arise as the Jacobian matrix approaches singularity.

Having perceived this important fact, many researchers investigated the problem and proposed some alternative techniques for determining the steady-state stability limit of a power system. Such techniques will be discussed in this sub-section.

Ajjarapu and Christy [18] proposed a continuation technique for automatic evaluation of the steady-state stability limit (or critical point). This technique is based on a predictor-corrector method which evaluates a continuum of solutions, starting from a base case and moving towards the critical point. Divergence due to ill-conditioning around the critical point does not occur with the proposed method. Figure 3.3 illustrates this approach for the *PV* curve of the 2-bus system.

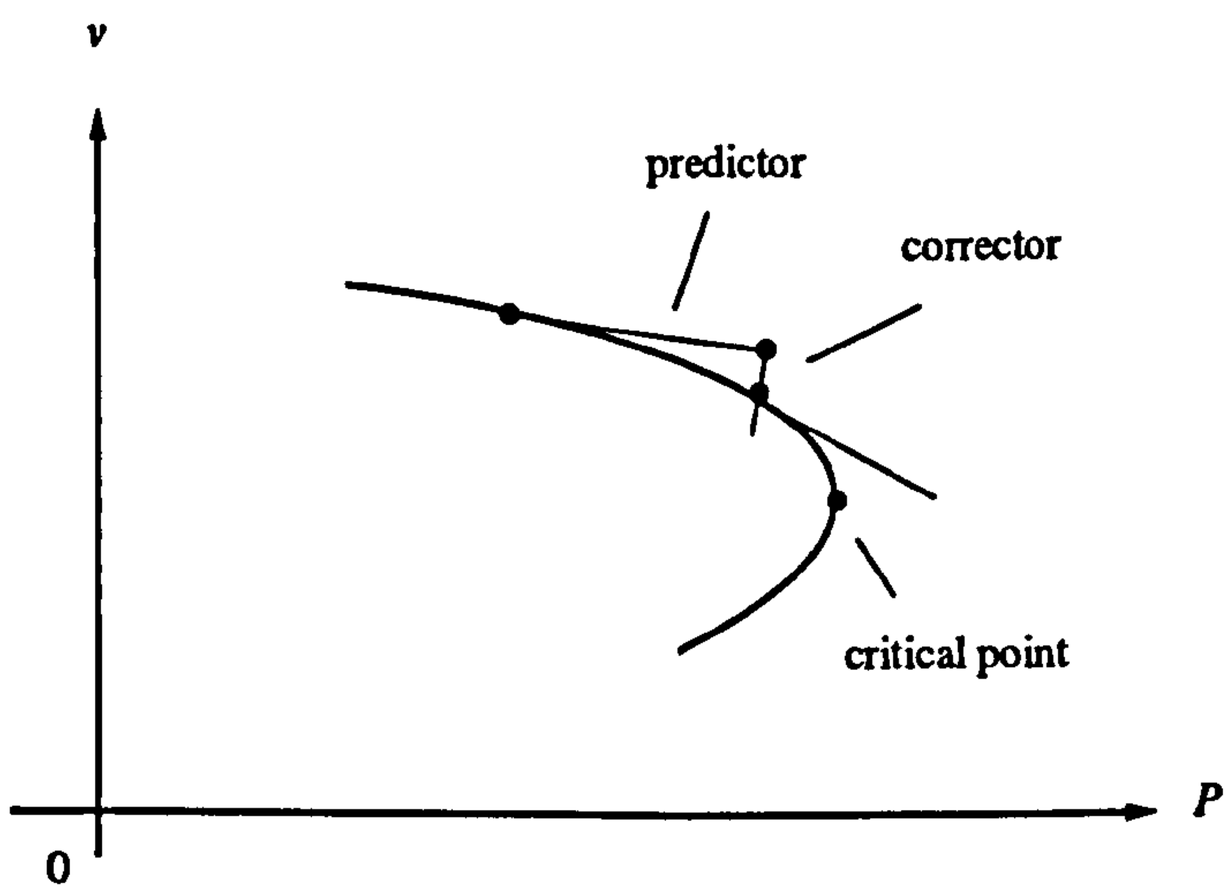


Figure 3.3 - Predictor-corrector method

Iba et al. [19], Semlyen et al. [20] and Kataoka [21] also proposed continuation techniques for evaluating the critical loading condition and also for obtaining relevant *PV* curves.

Tamura et al. [22] studied the relationship between multiple load-flow solutions and voltage instability. They concentrated on the behaviour of the multiple solutions as the load is increased, up to the point where solutions no longer exist (steady-state stability limit in Figure 3.2). They considered a series of different criteria for determining the stability of a particular solution. The first criterion -devised by Venikov et al. [8]- refers to the change of sign of the load-flow Jacobian matrix as the load is increased. The second criterion uses the sign of *V-Q* sensitivity factors, and the third criterion corresponds to the energy stored in both capacitive and inductive components in the network. As these criteria are not free of classification errors, they are organised in a sequential manner: only if a given solution passes the three criteria will it be considered as stable; otherwise it will be labelled as unstable.

Dehnel and Dommel [23] developed a modified load-flow technique for identifying weak areas which prevent convergence of the standard Newton-Raphson method. This technique is a modified version of the damped Newton-Raphson method, which produces "quasi solutions" with minimum mismatches. Sensitivity analysis is then used to determine the weak areas.

Galiana and Zeng [24] proposed a generalised sensitivity relationship between specified injections and resulting voltages in order to analyse the behaviour of the solution in the neighbourhood of a singularity. The proposed relation contains two orthogonal components which behave differently as the operating point moves towards the critical point. The proposed sensitivity relationship extends the conventional load-flow sensitivity analysis because it can be carried out for operating points at or near a Jacobian singularity, which does not occur in the conventional sensitivity analysis. In a second paper [25], the authors apply this generalised sensitivity relationship in the estimation of the maximum loading condition of a system. The proposed technique finds the distance from an initial operating point to the maximum loading point for a given direction of load increase. The maximum loading point can be found in as few as 5 load-flow runs, and the initial operating point need not be near the maximum loading point in order to obtain good accuracy.

Berizzi et al. [26] studied the existence of load-flow solutions for a sequence of load increases at individual buses in the electrical system, in order to determine the maximum admissible load. After stating the disadvantages of using a conventional Newton-Raphson based load-flow, the authors make use of the Hartkopf method, whereby the state-variable vector of the Newton-Raphson method is multiplied by an accelerating factor α at the end of each iteration. This factor α is computed from the sum of squared bus mismatches obtained in the previous two iterations. The authors then developed two Hartkopf-based algorithms for increasing the load and studying the convergence properties of the solution. The results reported in the paper show a reduction of up to 60% in the total CPU time, with respect to the conventional Newton-Raphson method.

3.2.4 - Optimisation techniques

Cutsem [27] proposed the reactive power margin of a group of buses as a measure of proximity to voltage collapse. This margin is the difference between the maximum reactive load and the base case value, and is obtained through an optimisation procedure in which the objective is to maximise the load. Non-optimised loads are treated as equality constraints and generator reactive limits as inequality constraints. The optimisation procedure is based upon Newton's method.

Jung et al. [28] also applied an optimisation technique for obtaining the closest critical point from a given operating point. The objective of the optimisation procedure is to minimise the l_2-norm distance between the current operating point and the closest critical point. The equality constraints are the power flow equations and the singularity of the Jacobian matrix. Generator power limits are treated as inequality constraints.

3.2.5 - Sensitivity methods

Flatabø et al. [29, 30] proposed the use of sensitivity methods for evaluating the voltage stability condition of a power system. In order to determine the voltage instability point, the sensitivity of various parameters with respect to the reactive power injected at any bus is used. The authors propose the MVAr-distance as a local indicator of proximity to voltage collapse. The MVAr-distance of a bus is the reactive load that can be added to the bus before the system becomes voltage unstable, and it is obtained through the sensitivity matrices. In the same way, MW- and MVA-distance to voltage collapse are computed. The sensitivity matrices are used to examine either the consequences of line outages, the effect of voltage dependency of loads on the distance to voltage collapse, or the importance of reactive power reserves on transmission equipment.

Begovic and Phadke [31] proposed the total generated reactive power as an indicator of voltage stability margins. They also derived sensitivity factors of the total generated reactive power with respect to both power injections in load buses and shunt susceptances. These factors were used for evaluating the allocation of reactive support and also for determining the most effective buses for load shedding.

3.2.6 - *M*-matrices

In an early work, Abe et al. [32] developed a thorough methodology for studying the VSP in power systems. The dynamics of the system is taken into account through first-order delay models used for both OLTCs and constant-impedance loads. The authors then applied the mathematics

framework of *M*-matrices for establishing necessary and sufficient conditions for voltage stability using a set of linearised dynamic equations (an *M*-matrix is a matrix such that all of its eigenvalues have positive real part).

3.2.7 - Transient *PV* curves

Ohtsuki et al. [33] proposed the use of "transient *PV* curves" for analysing the voltage stability of a power system. These curves are conceptually the same as the static curves presented in sub-section 3.2.2. In the present case, a *PV* curve is defined for each bus at each time *t* in a dynamic simulation of the system (only one simulation is required for building up these curves). Transient *PV* curves allow the estimation of the critical time delay of emergency control and also the critical point of reverse action for on-load tap changers.

Pal [7] also utilised the concept of transient *PV* curves, although he did not use a dynamic simulation program for their evaluation. Instead, he considered a simple 2-bus system in which the generator was modelled as a fictitious voltage source behind a fictitious reactance.

3.2.8 - Energy methods

The classical transient stability problem in power systems addresses the question as to whether a system will be stable after the occurrence and clearing of a large disturbance, and is essentially concerned about the balance of active power within the system. Traditionally, this problem has been solved through the use of time-domain simulations (or numerical integration methods), which produce accurate results but are computationally very expensive.

More recently, new direct methods have been developed. Direct methods assess the system stability without explicitly solving the differential equations of the dynamic model, thus providing much more economical means of stability evaluation.

The voltage stability problem has also been studied through the use of direct methods (or energy methods). Hiskens and Hill [34] proposed an energy method which preserves the structure of the network (thus allowing the use of sparsity techniques) and also considers non-linear loads. They start with a set of differential-algebraic equations which, in principle, are difficult to deal with. These equations are shown to be equivalent to a set of ordinary differential equations, which are then used to analyse the problem. A method is derived for determining and classifying the equilibrium points of the model.

DeMarco and Overbye [35] proposed an energy-based security measure for assessing vulnerability to voltage collapse. Initially, they represent a simple power system through a set of differential-algebraic equations. Then this model is transformed into a set of pure-differential equations and a Lyapunov-like energy function is devised. A distance to voltage collapse is then defined; this distance provides the maximum load increment (active or reactive) in a bus for a given value of energy before collapse occurs. The main idea is to use the method for efficient determination of low-voltage load-flow solutions with the smallest associated energy measures. The proposed method requires load-flow and energy evaluations, which must be carried out off-line. Thus, interpolation procedures are required for on-line applications. One of the main problems with this approach is the need to evaluate low-voltage solutions. In a more recent paper [36], the authors developed an improved method for overcoming this difficulty. In paper [37] one of the authors proposes the energy method as a local measure: each bus has an associated energy which provides an indication of the voltage security of the area adjacent to the bus (the lower the energy, the more dangerous the situation). It is shown how the number of load-flow solutions needed to analyse voltage security increases exponentially with the size of the system. A new method is then proposed for screening critical buses. The method is called FBBV (Fixed Boundary Bus Voltage) and it classifies buses in the network on different levels (level 1 refers to the first neighbours of a bus; level 2 refers to the second neighbours and so on). It is shown how FBBV can help reduce the total amount of computation in order to determine the set of lowest energy measures.

3.2.9 - Bifurcation theory

Bifurcation theory is a branch of the mathematics field of non-linear systems. When some parameters of a given system are varied, the system may experience a sudden change in its state; both previous and final states may differ qualitatively and quantitatively. The term "bifurcation" (or structural instability) refers to this sudden change. Several types of bifurcation are possible in this situation: saddle-node bifurcations, Hopf bifurcations, etc.

Saddle-node bifurcations occur when only one eigenvalue of the Jacobian matrix describing the system becomes zero due to changes in the system parameters. Hopf bifurcations occur when a complex conjugate pair of eigenvalues crosses the imaginary axis and moves into the right half-plane due to changes in the system parameters. The system may start oscillating with a small amplitude. Thus, a Hopf bifurcation point connects stationary solutions with periodic solutions.

Kwatny et al. [38] applied the bifurcation theory to stability analysis in power systems. The authors developed a rigorous analysis and stated the conditions for the existence of some bifurcation-related phenomena such as loss of steady-state stability and voltage collapse. Loss of steady-state stability means infinite sensitivity of bus angles (or maximum real power transfer) to changes in system parameters, and voltage collapse corresponds to infinite sensitivity of bus voltages.

Cañizares et al. [39] proposed the application of the Point of Collapse (*PoC*) method for the study of saddle-node bifurcations in ac/dc power systems. The *PoC* method basically consists of augmenting the set of system equations with an additional constraint that ensures the existence of a zero eigenvalue (of the original Jacobian matrix) at the point of interest. In doing so, the *PoC* Jacobian matrix becomes non-singular, thus allowing the determination of an acceptable load increase before voltage collapse occurs.

Ajjarapu and Lee [40] applied the Hopf bifurcation theory to dynamic phenomena in power systems. The authors analysed the stability of the resulting periodic solutions and applied the results in a simple power system.

Chiang et al. [41] developed a tool for analysing voltage collapse based on a centre manifold model. They identified three different stages during the operation of the system. Stage 1 corresponds to the quasi-steady state operation of the system before bifurcation occurs. In Stage 2, the system reaches the steady-state stability limit by encountering a saddle-node bifurcation. In Stage 3, the system dynamics after bifurcation is described by a centre manifold model. In Stage 1, the solution curve is obtained through an indirect method (predictor-corrector) in order to avoid convergence problems in the neighbourhood of a bifurcation. For Stage 2, a special algorithm, based on Powell's hybrid method, is proposed for obtaining the bifurcation point. Finally, in Stage 3, assuming that x^* is the point at which the system Jacobian matrix has one zero eigenvalue and that p is the eigenvector associated with this zero eigenvalue, the centre manifold is the curve made up of system trajectories which is tangent to the eigenvector p at point x^*.

Jean-Jumeau and Chiang [42] developed an interesting method for solving the load-flow equations in the vicinity of a saddle-node bifurcation, where convergence problems arise in the conventional Newton-Raphson method due to ill-conditioning of the Jacobian matrix. The methodology is based on the parameterisation of the load-flow equations, which locally removes near-singularities and therefore enlarges the region of convergence around such points. Its implementation requires a simple modification of the standard load-flow equations. The dimension of the system of equations remains unchanged and only a few non-zero elements are added to the Jacobian matrix, thus preserving its sparsity.

Dobson and Lu [43] developed direct and iterative methods for computing the distance from a given operating point to the locally closest saddle-node bifurcation. This distance is measured using an auxiliary vector which is normal to the hypersurface of critical load powers. This surface is the loci of all points in the load power space for which the Jacobian matrix results singular with only one zero eigenvalue (saddle-node bifurcation). The method exploits the fact that this normal vector is easy to compute. The distance between the current operating point and the locally closest saddle-node bifurcation is also called *worst case load power margin* and is proposed as a voltage

collapse index.

3.2.10 - Singular perturbation theory

Yorino et al. [44] proposed the use of singular perturbation theory for analysing voltage instability problems. According to this approach, the dynamic description of the system can be made separately through two different subsystems: a "fast" model and a "slow" model. The slow model includes those devices for which the response occurs in the time-frame of minutes (such as tap changers and thermal units of loads), while the fast subsystem includes much faster devices such as automatic voltage regulators and governors. This approach allowed the grouping of voltage instability problems into the following categories:

I) voltage drop due to slow dynamic factors;
II-1) voltage collapse due to slow dynamic factors;
II-2S) voltage collapse (static bifurcation) due to fast dynamic factors;
II-2D) voltage collapse (dynamic bifurcation) due to fast dynamic factors;

Furthermore, the authors claim that types I and II-2S can be analysed through the determinant of the load-flow Jacobian matrix; type II-2D through eigenvalue analysis of the dynamic system; and type II-1 through direct non-linear techniques.

Nwankpa and Shahidehpour [45] proposed the mean first passage time (MFPT) as an indicator of proximity to voltage collapse. The first passage time is the time before the stable operating point disappears beyond the stability boundary. The MFPT is the average taken over all initial conditions. The authors use a stochastic approach for formulating a boundary value problem, for which an asymptotic solution is obtained through the singular perturbation theory. This solution allows the first passage time to be evaluated.

3.2.11 - Simulation approach

Begovic and Phadke [46] proposed the study of voltage collapse through a dynamic simulation approach. The model of the power system is composed of differential equations for generators and load-flow equations for the network, which are numerically solved in the time domain. They analysed cases in which the loading condition at $t = 0$ was near to the critical value (where the Jacobian matrix becomes singular) and slowly increased the loads in order to lead the system into voltage collapse. They showed that in some cases early rescheduling of loads can avoid the later occurrence of voltage collapse.

A similar approach was proposed by the same authors [47], where the focus was placed on the use of a reduced state vector for estimating the actual state of the power system. The idea is to group the system buses into clusters of electrically similar buses so that each cluster is represented by one of its buses only, thus reducing the dimensionality of the problem.

Lachs and Sutanto [48] also proposed a simulation approach for studying voltage instabilities. They used a static model (load-flow) which was run at strategic instants in order to capture the dynamic nature of some automatic controls (tap changers and generator excitation) ("snapshot" approach). They showed the importance of timely countermeasures, which can effectively avoid the occurrence of voltage collapse.

Cutsem [49] also studied emergency voltage situations through the simulation technique. The main contribution of this paper is perhaps the establishment of different time frames for the phenomena involved in voltage instability processes. These time frames correspond to (*i*) fast components (network, generator AVRs, governors, etc.), (*ii*) mid-term controls (OLTCs, maximum excitation limiters for generators), and (*iii*) components exhibiting long-term behaviour (such as thermostatic loads). The relative independence among these three time-frames allows, in some cases, the dynamic simulation to be carried out without performing numerical integration, i.e., the simulation is done through the evaluation of a sequence of equilibrium points using algebraic equations only. This technique leads to a computationally effective simulation of the system in the time domain.

Morison et al. [50] developed a thorough comparison between a static technique and a dynamic simulation program. This paper represents an extension of an earlier work reported by the same authors [16], in which they had proposed the use of modal analysis for estimating the static voltage stability of a system and for identifying critical areas and appropriate corrective actions. In the more recent paper, the Extended Transient/Midterm Stability Program (ETMSP) [51] is used for studying various load scenarios. The main contribution of the paper is perhaps the attempt to bring together both static and dynamic approaches. The authors obtained consistent results and showed that the static approach presents some important advantages, such as less computational cost and the possibility of developing sensitivity analysis. The dynamic simulation approach offers, however, unique capabilities for executing detailed dynamic analyses and studying the coordination of controls and protections.

3.2.12 - Device analysis

It has now been recognised that some automatic voltage control devices, such as on-load tap changers (OLTCs) and static VAr compensators (SVCs), play an important role in the voltage instability problem. This is especially true for OLTCs, whose dynamics is relatively slow with

respect to that of other voltage control devices. Some attempts have been made to develop detailed models for such devices, and they will be presented in this sub-section.

Medanic et al. [52] developed discrete and continuous dynamic models for OLTCs and showed how poorly coordinated devices can lead to either transformers hunting one another or voltage collapse. They also established theoretical conditions for assuring OLTC operation which maintains voltages within acceptable limits.

Liu and Vu [53] developed and extremely detailed analysis on OLTCs, which is based on a non-linear dynamic model. The main results of the paper are the establishment of a stability criterion for equilibrium points and a method for obtaining voltage stability regions in a system with several OLTCs.

Ohtsuki et al. [54] studied the reverse action of an OLTC, i.e., the situation where the secondary voltage of a transformer drops as a consequence of raising the tap position. Their system model included a dynamic model of the OLTC. They showed how the occurrence of OLTC reverse action may lead to voltage collapse in a heavily loaded system.

Hiskens and McLean [55] developed an analysis of the SVC behaviour under voltage stability conditions. They showed how reverse action may also occur in voltage collapse conditions when using SVCs (the bus voltage decreases when increasing the capacitive susceptance). Therefore, proper coordination between the SVC control system and the local PV characteristic is required.

3.3 - Artificial neural networks in power systems

3.3.1 - Introduction

This section presents the results of the literature survey on the application of neural networks in Power Systems. The publications that were found are classified in several groups according to the electrical problem under consideration. These groups are as follows:

- security assessment and dynamic stability;
- load forecast;
- diagnosis;
- contingency analysis;
- voltage harmonics;
- unit commitment;
- control and observability analysis;

- modelling;
- economic load dispatch;
- Distribution systems.

In the following sub-sections, each problem will be briefly addressed and some comments will be made on the corresponding publications. The neural network models mentioned here will be discussed in detail in Chapter 4.

3.3.2 - Security assessment and dynamic stability

Security assessment analysis addresses the problem of classifying a given operating state of a power system as either secure or insecure, according to a pre-specified criterion. This security criterion is often related to the dynamic transient stability of the power system, and for this reason neural network applications to both security assessment and dynamic stability are discussed together in this section.

Typically, security assessment involves off-line analysis and on-line assessment. A number of contingencies are studied off-line, classified according to their security level, and finally stored in a security database. During on-line operation the power system continuously changes between different states (or operating points), and each state must be assessed as to its security level. In general terms, given a system state, the operator must find a similar state in the database. When this search is not successful, the operator must interpolate among the best matches in the database. This task must be executed very quickly due to the on-line nature of the problem. Depending on the number of variables involved, a human operator may not be able to produce an answer in an adequate amount of time. Therefore, efficient pattern recognition and interpolation techniques are very good candidates for application in the security assessment problem.

El-Sharkawi et al. [56-58] have dealt with the security assessment problem through the use of a Multi-Layer Perceptron (MLP) network trained with the backpropagation algorithm. They addressed both the static and the dynamic aspects of the problem. In static assessment the load-flow equations are used so as to check for violations in voltage or power flow constraints given a set of contingencies. In dynamic assessment the non-linear power system model is linearised around an operating point and linear analysis techniques (e.g., eigenvalue analysis) are applied for assessing the local stability of the system. The authors of these papers used both techniques to set up a training set for the neural network and showed the classification and interpolation capabilities of the MLP.

Pao and Sobajic [59,60] proposed the use of neural networks for estimating the critical clearing time (CCT). The CCT is an important parameter associated with the transient stability of

a power system. It is the maximum time after which a fault must be cleared if dynamic stability of the system is to be preserved, and it can be obtained through exhaustive simulations with a transient stability program. In the first paper, the authors used an MLP for estimating the CCT. In the second paper, they proposed a very interesting technique which combines unsupervised and supervised learning. According to this technique, the first step is to classify all the training vectors into clusters using an unsupervised-type classifier, in order to reduce the dimensionality of the training set. This exploits the fact that similar vectors can be represented by an average exemplar of the class. The second step is to train an MLP network with the backpropagation algorithm and using the reduced training set, thereby reducing the overall training time.

Niebur and Germond [61] proposed the use of the self-organising map of Kohonen (unsupervised learning) for assessing the static security of a power system. Their electrical criterion is the loading state of transmission lines (normal or overloaded). The neural network clusters all the training vectors in fewer categories, and the authors show how new operating states are accommodated in the already existing categories so as to estimate the security level of these new states.

Mori et al. [62] applied the self-organising map of Kohonen to the dynamic stability problem in a power system. They chose the most critical eigenvalue of the system matrix (obtained through the S-matrix method) as the dynamic stability index. They proposed two methods for inferring the value of the stability index. This is a major issue when using unsupervised training for evaluating parameters that were not present in the training set, because the neural network cannot build up an internal representation of the relationship between the input variables and the desired indices.

Fidalgo et al. [63] developed an MLP-based tool for predicting transient stability margins. Inputs for the MLP are given by active power, *emf* and inertia constants for generating units. The output corresponds to an estimation of ΔV, the transient energy function margin (for $\Delta V < 0$ a given contingency is classified as unstable; for $\Delta V > 0$ it is labelled as stable). The MLP was trained using the so-called Adaptive Backpropagation algorithm (ABP), whereby the learning rate of individual neurons is adjusted during the training according to the behaviour of the corresponding derivative. The ABP algorithm is reported to substantially reduce the overall training time. Another interesting feature of this work is the use of MLPs for suggesting corrective actions whenever an unstable situation is detected. In this case, MLPs are used to compute derivatives of the output variable with respect to the input variables (i.e., the sensitivity of the transient energy margin to the control variables).

Finally, Fouad et al. [64] developed an MLP-base tool for assessing the *vulnerability* of power systems in the transient stability problem. Vulnerability is defined as a combination of the transient energy margin ΔV and the sensitivity $\partial \Delta V / \partial p$, where p is a critical system parameter (in this case, p is defined as the power flow in a stability-limited transmission line). Depending on

the value of these two variables, the system is classified as either vulnerable or non-vulnerable, for pre-defined faults. The inputs to the MLP are the energy margin ΔV, the distribution factors for the flow in the transmission line of interest, and the sensitivity of the energy margin with respect to changes in the power of critical generators. The output of the MLP is the vulnerability status. The authors reported good classification results with this technique and also proposed the use of the MLP in on-line environments.

3.3.3 - Load forecasting

Knowledge about the future behaviour of the load is essential in many problems in Power Systems. Depending on the amount of time involved in the load prediction, load forecasting can be broadly classified as short-term, mid-term and long-term forecasting.

Short-term forecasting (few minutes to one hour ahead) allows system operators to adjust the various controls for meeting the demand of the system (for example, the rate of change of generator output). Mid-term load forecasting (from one hour to one week) is needed for optimal generator unit commitment, start-up and shut-down of thermal plants, and control of power exchanges in interconnected systems. Long-term load forecasting (up to 10 or 15 years) is used for planning the system expansion (purchase of new generator units and other facilities).

Traditionally, load forecasting has been carried out within two basic frameworks, namely time series analysis and regression analysis, although other methods exist that do not fall into either category.

The time series approach does not normally take into account weather information (temperature, humidity, wind speed, etc.), which is an important parameter, especially for residential loads, and thus it often produces inaccurate predictions. On the other hand, the regression approach attempts to associate weather variables with the load demand, and a functional relationship (model) must be stated *a priori*. Usually, a linear model is adopted. As the relationship between these variables depends on time, conventional regression models fail to represent this temporal variation.

The first attempt at tackling the load forecasting problem through the use of neural networks was by Dillon et al. [65]. Furthermore, their 1975 paper is probably the first attempt at applying neural networks in Power Systems. The authors developed a self-organising learning machine capable of representing the non-stationarity and the seasonality of load data.

Most of the remaining publications studied use the Multi-Layer Perceptron (MLP) model

trained with the backpropagation algorithm [66-70]. Paper [68] proposes an adaptive learning algorithm which allows the momentum coefficient α (see sub-section 4.4.5) to be updated during training. Paper [69] proposes a modified neural network which adds a set of linear terms to the normal non-linear computation of the basic MLP architecture. Finally, paper [70] describes the combination of several basic MLP networks producing a more complex, not fully connected overall network. Research using the MLP model reports very low predicting errors, in the order of 1 - 2%.

Peng et al. [71] proposed an adaptive neural network approach to one-week ahead load forecasting. The technique is based on the Adaline model, and the load data is separated into 3 components (base, low frequency, and high frequency) through the use of digital filters. Each component is then forecasted by one Adaline unit. Each Adaline has an input sequence, an output sequence (forecasted load), a desired response-signal sequence, and a set of trainable weights. The weight vector is designed to make the output sequence follow the actual load sequence. The forecasted errors are reported to be less than 3.4%.

Lu et al. [72] carried out a detailed study for evaluating the effectiveness of the artificial neural network approach on the short-term load forecasting, also using the MLP model. They used real data from 2 different companies in order to predict both next hour load forecasts (1 value only) and next 24-hour load forecasts (a sequence of 24 values). The main conclusions of the paper can be summarised as follows: (*i*) the neural networks are system dependent, in the sense that systems with different load characteristics require different neural network architectures; (*ii*) in general, the neural network model is case independent, except for season changing periods and abrupt changes in temperature and load conditions; and (*iii*) the neural networks are sensitive to bad data, thus requiring data pre-processing by intelligent filters.

Baumann et al. [73] carried out an interesting comparison among four different approaches to the short-term load forecasting problem: (*i*) Kohonen learning, (*ii*) Backpropagation learning, (*iii*) Multiple regression analysis, and (*iv*) Kalman filters. The testing period spanned over two weeks, and the Kohonen model was the one that best responded to the forecasting problem, especially with respect to adapting to a new situation (such as the transition from summer season to winter season). The MLP model produced the worst results, but this was probably due to the small size of the training set (only one year was represented). The Kalman filter technique, and especially the multiple regression method, reacted slowly to the seasonal transition.

3.3.4 - Diagnosis

In this sub-section, published research in power system diagnosis is presented. These include alarm processing, operating conditions monitoring, fault diagnosis and predictive mainte-

nance in gas-insulated equipment.

Chan [74] proposed the use of a multi-layer perceptron for implementing an intelligent alarm processor (IAP). The main purpose of the IAP is to help the operator identify a fault in a system given a set of active alarms. The author reported successful identification of the problem in cases where there was no noise (missing alarms), and partial success when there was noise. One of the best achievements of this IAP is the ability to produce a single message from many active alarms (for instance, a bus fault which activated 15 alarms could be identified and reported with only one message).

Sobajic et al. [75] proposed a neural network-based system for on-line monitoring and diagnosis of power systems. This paper is a rather general description of the principles of such a system. They also proposed the combined use of both unsupervised and supervised learning and also the use of a hybrid approach, by which an expert system would share some specific tasks with a neural network.

Tanaka et al. [76] used a multi-layer perceptron to implement a fault diagnosis engine. Given the current status of protective devices (circuit breakers and relays), the network was able to indicate the faulty component (bus, line or transformer). They set up a training set containing situations with zero or one malfunction (missing circuit breaker or relay information) and applied the trained network to a different testing set with double malfunctions. A very interesting feature of their work is that the network, once trained, was able to produce additional meaningful "rules" (training vectors) which were not present in the original training set.

Kandil et al. [77] implemented three uni-flow counterpropagation networks for fault identification in an ac-dc transmission system. Input variables for the networks are voltages (dc values and *rms* and instantaneous ac values) and the outputs are the system state (fault/no fault) and the type of the fault if one exists.

Finally, Ogi et al. [78] applied the self-organising map of Kohonen in the detection of abnormalities in gas-insulated switchgear. Assessing the condition of internal components in this kind of equipment is usually difficult, because inferences have to made from indirect, external measurements. This characteristic has prompted the development of predictive maintenance for gas-insulated equipment. In this case, an acceleration sensor is attached to the external surface of a gas-filled tank. This sensor is capable of detecting partial discharges that occur in the gas due to small pieces of metal ("junk"). The sensor signal is sampled at a rate of 853 samples per cycle. The samples undergo a pre-processing stage whereby a Fast Fourier Transform (FFT) is performed and the resulting spectrum is averaged in time (in order to cancel noise), followed by an amplitude-normalisation process. The result from this pre-processing stage is presented to the neural network, which is then capable of delivering 6 different diagnostics: (*i*) normal condition, (*ii*) existence of a

conductor-fixed junk, (*iii*) existence of a tank-fixed junk, (*iv*) existence of a floating junk, (*v*) existence of a small gap between conductor ends at a junction, and (*vi*) insufficiently-fixed metal fixing. The authors show examples of the sensor signal for each one of the diagnostics above, and from them the relevant features of each diagnostic cannot be easily distinguished. However, after the pre-processing step is applied, the signals are very different from one another, and this is the information that is passed on to the neural network. The authors report a success rate of 80% when diagnosing abnormalities using this methodology.

3.3.5 - Contingency analysis

The main purpose of contingency analysis is to classify each contingency in the system (line and generator outages) as secure or insecure. A secure contingency is one which does not lead to line overloads, bus-voltage or generator-VAr violations. Insecure contingency is one in which at least one of these limits is violated. The number of contingencies to be analysed in a power system is usually extremely high, and therefore a contingency selection procedure -or contingency screening- is needed in order to reduce the dimensionality of the problem. The screening strategy must be adequate so as to guarantee that no insecure contingency is missed, and also to avoid false alarms (a contingency being classified as insecure when it is not).

There are several methods for solving the contingency screening problem. One of them uses linear approximations around a given operating point in order to take into account the modifications in the operating conditions (for instance, distribution factors for reflecting topology changes). Another method is the ranking of the contingencies according to a suitable performance index (PI). A threshold value for this PI must also be provided for classifying the contingencies as either secure or insecure.

Fischl et al. [79, 80] addressed the contingency screening problem through the use of neural networks. In the first paper they developed a multi-layer perceptron for which the inputs are the system susceptance matrix and active power injections at buses. The outputs are power flows through selected lines and a binary flag for indicating the occurrence of overflows. During normal operation of the MLP, they reported only a few cases of misclassification.

In the second paper the authors utilised a more pattern recognition-oriented approach. They devised a Hopfield network for detecting the limiting contingencies in a power system. They defined what a limiting contingency is and showed how this attribute may change during on-line operation. An interesting feature is that they trained the network through a linear programming-based method, instead of the sum-of-outer-products (SOOP) algorithm commonly associated with Hopfield networks. They also showed how the new training method attained a larger stability margin with respect to the SOOP algorithm.

3.3.6 - Harmonic analysis

Harmonic current injection in power systems is caused by either nonlinear industrial loads or semi-conductor controlled loads, such as TV sets, air conditioning equipment, etc. The problems that may arise from this phenomenon range from capacitor banks overheating and relay misoperation to harmonic resonance in the worst case.

One technique for dealing with the problem of harmonic measurement is the state estimation approach. At the fundamental frequency, the state estimation technique allows obtaining accurate estimates of voltages and power flows from redundant, noisy measurements. This technique can be extended to the measurement of harmonic quantities. In this case, a large number of harmonic detectors must be placed throughout the network. Hartana and Richards [81] studied the possibility of substituting measured values for a set of initial estimates (*pseudomeasurements*), thereby allowing the use of less permanent harmonic instrumentation. Furthermore, they investigated the possibility of obtaining these pseudomeasurements through the use of a multi-layer perceptron. Having obtained the pseudomeasurements, they applied the state estimation technique to find additional unknown harmonic sources. The authors applied their proposed technique to the IEEE 14-bus system.

Mori et al. [82] presented a brief review of current methods for predicting voltage harmonics in a power system. Based upon one of these methods, namely Recursive Least Squares, they set up a multi-layer perceptron for dealing with this problem. Generally speaking, the neural network is able to predict the output value at time $t+1$ given a time series containing p previous values of voltage harmonics. They also carried out an analysis of the number of input- and hidden-layer units and its influence on the overall performance. The authors reported almost the same accuracy as the most effective conventional techniques.

3.3.7 - Unit commitment

The unit commitment problem seeks to allocate generating units with minimum operational cost in order to meet a forecasted load pattern. The variables normally involved in this problem are the production cost of generating units, minimum start-up and shut-down times, forecasted load, and so on. For example, short-term unit commitment deals with hourly unit allocation within a time span of 24 hours.

The unit commitment problem has so far been studied through various approaches, namely

Dynamic Programming (DP), Branch and Bound techniques, Mixed Integer Programming and Lagrangian Relaxation. The DP technique is probably the most popular approach to the problem, and it normally involves very large amounts of computation.

Very often the changes in the load pattern between corresponding scenarios are very small, but the DP approach does not take any advantage of this fact; a complete new case must be re-run even if the changes are small. Based on this important observation, Ouyang and Shahidehpour [83] proposed a new approach to the unit commitment problem, a hybrid neural network-dynamic programming technique. The neural network receives a load forecast pattern (24 inputs in the case of short-term unit commitment) and produces a schedule table which contains the status (on/off) of each generating unit in each time interval. The neural network -a multi-layer perceptron (MLP)- may produce either "determined" values (generating units on or off) or "undetermined" values (intermediate values between on and off, due to the interpolative nature of the MLP). When the neural network produces a determined output, no further calculations are performed, because it is assumed that the MLP was presented with a load pattern very similar to one of the vectors in the training set (obviously, the training set was prepared in advance using the mathematical programming techniques). On the other hand, when the neural network produces an undetermined output, the best solution is found by complementing the existing pre-schedule with a feasible selection, using dynamic programming. The authors show that this method can significantly reduce the execution time of the DP approach without degrading the quality of the results.

In a different approach, Sasaki et al. [84] applied the Hopfield paradigm to solve the combinatorial optimisation problem related to unit commitment. However, they reported some discrepancies between the results obtained with their method and the Lagrangian Relaxation (LR) method. Also, the computing times were higher than those of the LR method.

3.3.8 - State estimation and observability analysis

Power system state estimation is responsible for generating a reliable real-time database with information on the current state of the system. This database is used in an energy management system by advanced functions for system security monitoring and control.

Alves da Silva et al. [85] developed an MLP network for state forecasting in power systems. They used a least-square based training method called Optimal Estimate Training 2 (OET2). The authors applied this methodology to the state estimation of the 24-bus IEEE Reliability Test System using a 5-minute estimation cycle for a given day during the winter. Differences between estimated, filtered and true values are reported to be very low (filtered values correspond to the update of estimated values at time k+1 due to the availability of new measurements at time k+1).

Before using a state estimation procedure, it is necessary to assess the topological observability of the network, which address the following major question: is it possible to obtain bus angles and voltages from a given set of measurements? In general terms, the observability of a power system is related to the non-singularity of *observability matrices* which relate a set of measurements to a set of system state variables.

Mori and Tsuzuki [86] presented a description of the topological observability problem and also carried out a brief survey of the current approaches to the problem. Furthermore, they proposed two alternative approaches, namely an integer programming-based method and a neural network-based method. First, the authors established the integer programming approach, which was later transformed into a neural network formulation. The latter is based upon the Hopfield model. They applied the neural network to a 5-bus test system, with different observability conditions (observable/unobservable). They also showed that the Hopfield model is highly influenced by some key parameters, and thus they carried out a detailed analysis of these parameters.

3.3.9 - Modelling

Chow and Thomas [87] reported the use of a multi-layer perceptron network, trained with the backpropagation algorithm, for modelling the dynamics of a synchronous machine.

The authors modelled the dynamic response of the machine to a step change in the field voltage. The variables associated with this analysis were classified into state variables and control variables. All the control variables (e.g., mechanical torque) are assumed to be known during the transient analysis. All state variables (e.g., rotor speed) are assumed to be directly measurable and known at instant k. Control and state variables at instant k constitute the input to the neural network, and the output is defined as the state variables at instant $k+1$. Therefore, the training set is made of training pairs which represent sequential snapshots of the transient period, taken at discrete values of time.

The authors showed that the neural network was capable of capturing the dynamic behaviour of the synchronous machine, although they did not consider evaluating the performance of the neural network with a testing set different from the training set. A brief analysis of the relationship between the number of neurons in the hidden layer and the precision of results was also included.

3.3.10 - Economic load dispatch

The Economic Load Dispatch (ELD) problem is concerned with determining the amount of

power that each plant in a power system must generate in order to satisfy the system's demand, and this has to be done in such a way that the total generation cost is minimum. The problem is complex due to different operating costs for different plants, and also to different fuel costs. It is further complicated by other operational constraints such as minimum and maximum power output for each plant.

Matsuda and Akimoto [88] were the first to study the ELD problem through the neural network technique. They applied a Hopfield model (see section 4.6) and showed the advantages of using this approach, especially with respect to the difficulty of formalising the problem.

Park et al. [89] also used a Hopfield model for solving the ELD problem. The authors did not address the essential issue of the local minima problem, which can seriously affect the operation of Hopfield networks.

3.3.11 - Distribution systems

The number of neural network applications in Distribution systems is considerably lower than that for generation and transmission systems. The problems analysed through this approach include optimum capacitor control, detection of high impedance faults and calculation of eddy currents in electrical conductors.

Iwan Santoso et al. [90] proposed the use of a two-stage multi-layer perceptron for optimal switching of distribution capacitors. The first stage receives information about active and reactive power, voltage magnitude and current capacitor settings; this input information is obtained through measurements at certain key locations in the distribution network. The output provides the load profile for various subsystems (each subsystem represents a set of loads with similar behaviour; different subsystems need not have the same temporal behaviour). The second stage receives the information about these load profiles and outputs new capacitor settings. The main purpose of this two-stage approach is to "break down" the high nonlinearity of the problem, in order to improve the training process of the neural network. The training set of the first stage is obtained through an electrical model of the system (load-flow) with input vectors conveniently generated. The training set of the second stage is obtained with the help of a capacitor control optimisation procedure. The authors tested their method on a 30-bus system and reported accurate results.

Kim et al. [91] implemented a 2-stage Multi-Layer Perceptron with backpropagation training for reconfiguring Distribution feeders in order to reduce losses. Whenever the load level changes, a new configuration is found such that the total loss is reduced. The first-stage MLP determines, from measuring data, the load level by *zone* (a zone is a group of buses located between switching devices). The second-stage MLP selects the best configuration from the zone

load level estimated by the first-stage MLP. After selection of the best configuration, a switching sequence is applied so as to change the configuration of the system from the current one to the selected one. All switching sequences have been defined in advance (off-line mode). Good results are reported in terms of losses when compared to minimum losses.

Ebron et al. [92] applied a multi-layer perceptron (with backpropagation training) for detecting high impedance faults in distribution feeders. Initially, they describe the problem in detail, showing its most difficult aspects and the various techniques that have been used so far to solve it. The neural network processes phase and neutral current over a 10-cycle period. The information gathered is then pre-processed and 200-element input vectors are generated. Each input vector is obtained with the aid of the EMTP program (ElectroMagnetic Transient Program) and contains information such as the peak value of transient currents on the 3 phases, the amount of imbalance between phases, and so on. The output vectors are 1-dimensional; the information contained in them concerns the system status (presence or absence of a high impedance fault). After training, the network is capable of deciding as to whether a high impedance fault exists given a measured input.

Sultan et al. [93] also studied the problem of detecting high impedance faults in distribution systems through the use of a multi-layer perceptron. In this case, the pre-processing stage consists of comparing *rms* and instantaneous values of current between different cycles, which are then fed into the inputs of the neural network. The analysis is carried out over a number of cycles, during which the neural network output is integrated over time for evaluating the response of the detector.

Chow et al. [94] implemented a Multi-Layer Perceptron network with backpropagation training to recognise animal-caused faults in Distribution systems. Faults of this kind account for a considerable percentage in the operational area of the Duke Power Company (USA), and therefore it is important to quickly recognise the cause of a given fault. In this case, the inputs for the MLP are conditional probabilities of causes c given events x, where the description of events x include the circuit identification, weather, season, day of week, time of day, number of phases affected and the actuated protective device; and the set of causes c comprises 11 different fault causes (one of which corresponds to animal-caused faults). The output from the MLP is a flag indicating whether or not the fault described by the inputs is animal-caused. Accuracy of this network is reported to be 99% with the training set and 98% with a testing set not used during training.

Finally, Feria et al. [95] proposed a "cellular neural network" for studying eddy currents in electrical conductors. Their method basically corresponds to a network analogue of the well-known finite difference method, by which the continuous integro-differential equations of the electrical problem are transformed into discrete difference equations. Furthermore, the cellular units do not possess a learning procedure; they only perform algorithmic calculations. This raises questions as to whether the proposed method can be referred to as a neural network approach. It should

be pointed out that this work was strongly criticised by the discussers of the paper.

3.4 - Summary

This chapter has presented an extensive literature survey, where the main purpose was to assess the state-of-the-art of the voltage stability problem and the application of artificial neural networks in electric power systems.

The survey on voltage stability showed that this electrical problem is at an early stage of maturity. Evidence of this is the relatively high number of different approaches that have been proposed, as well as the number of discussions and closures which could be found in many papers, some of them in no agreeing terms. All the methodologies have advantages and disadvantages, and they contribute in one way or another to the solution of the problem.

The dynamic simulation approach appears to be the most thorough technique for studying the electrical problem. It allows a rich representation of those system components which play an essential role in voltage stability, such as induction motors and OLTCs. An inherent disadvantage of this methodology is the fact that it cannot provide answers in the time frame of on-line operations. Even with the computational speeds available today, a dynamic simulation tool is far too slow for this application.

Regarding the application of artificial neural networks, it was possible to find a number of well-developed applications in different areas. This is particularly true in the area of load forecasting, where the learning abilities of Multi-Layer Perceptrons can overcome the main drawbacks of conventional techniques. Most applications are based on the MLP architecture, which exhibits excellent computational speed and interpolation capabilities. It should be noted that the vast majority of these applications are relatively recent (1989-90 onwards).

From the discussion above, as well as the conclusions of Chapter 2, it is now possible to justify the research path that was followed in the present work. On the one hand, it is clear that no attempt has been made to study the voltage stability problem through the neural network approach. On the other hand, the main characteristics of the MLP architecture (speed and accuracy) make it a serious candidate for use in the voltage stability problem in conjunction with a dynamic simulation tool. This subject will be further developed in Chapter 6.

CHAPTER 4

OVERVIEW OF ARTIFICIAL NEURAL NETWORKS

4.1 - Introduction

This chapter presents a general discussion on the subject of neural networks. Initially, some considerations on biological neural networks and the attempts of simulating them that have been made so far are presented. The training phase of an artificial neural network is a crucial step and is also discussed in some detail.

The remainder of the chapter is dedicated to the description of the most important models of artificial neural networks available today. All these models were implemented using the *C* language and are currently operating on Unix-based workstations in the Department of Electronic Engineering, QMW. Examples of applications are always included at the end of each section. Emphasis is put on the *Multi-Layer Perceptron* and its associated *Backpropagation* training algorithm, by far the most popular model of artificial neural networks.

It should be pointed out that the subject of artificial neural networks is very extensive and by no means is it wholly covered here. Also, the literature on neural networks is found among very disparate sources due to the interdisciplinary nature of the field. Much of the research work presented here is based upon Wasserman [96] and Beale and Jackson [97], which the author recommends as introductory texts on neural networks. A minimum set of references is given alongside each model.

4.2 - Biological and artificial neural networks

The idea of simulating biological neural networks stems from the desire to understand the structure and functioning of the human brain. Questions like *How do we think?*, *How do we learn?* and *How does the brain work?* have long since been the subject of great attention in several

branches of science. This has been done with a double purpose: first, to understand the human brain from a physiological and psychological viewpoint; and second, to produce computational systems (artificial neural networks) capable of performing brain-like functions.

Although a great deal of effort has been dedicated to the aforementioned questions, there is still very little understanding of the human brain. Nevertheless, some basic facts about it are known. It is a highly specialised structure where several different regions are responsible for dedicated tasks. The basic unit of the brain is the *neuron*, which is a self-contained processing unit. There are two main types of neurons, namely the *interneuron cells* and the *output cells*. Interneuron cells occur within specific regions of the brain and their connections with neighbouring cells span over about 100 microns. Output cells either connect different regions of the brain to each other, or connect the brain to a muscle, or connect from sensory organs into the brain. The neurons are arranged in layers and it is estimated that their total number in the brain is 10^{10}, with each neuron connecting to 10^4 neighbouring neurons. Figure 4.1 depicts a basic scheme of a biological neuron.

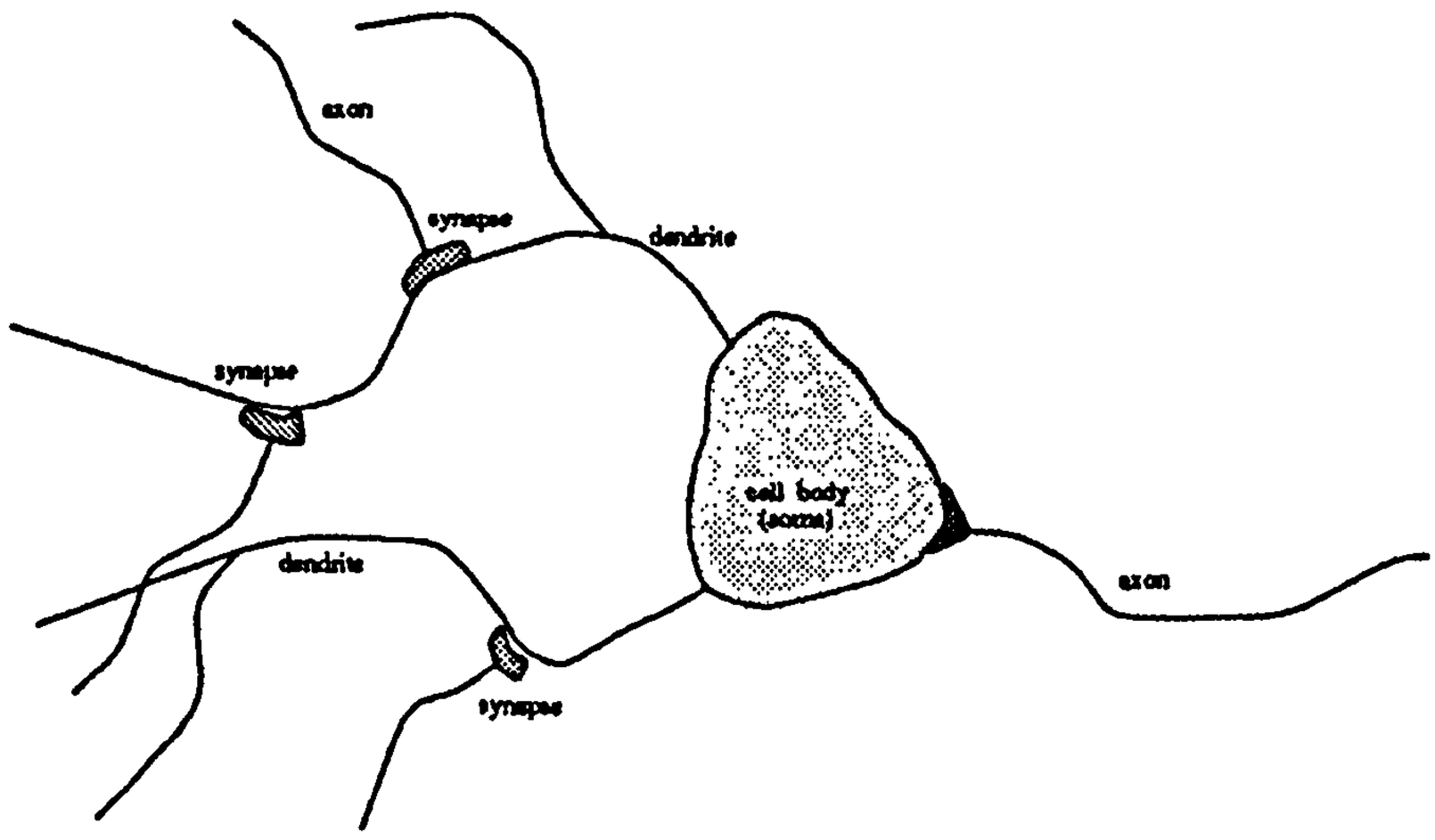


Figure 4.1 - Basic biological neuron (reproduced from [97] with permission)

The body of the neuron is the *soma*, to which long filaments called *dendrites* are attached. The dendrites act as the inputs to the soma. The *axon* can be seen as the output of the soma. Axons always occur in output cells but are often absent from interneurons, in which case they also act as the output of interneurons. Unlike dendrites, axons are electrically active, producing a voltage pulse (action potential) when the potential within the soma (due to the sum of the input dendrites) exceeds a critical threshold. This action potential lasts about 10^{-3} s and is actually a series of voltage spikes. The axon terminates in a special contact called *synapse*, which links the axon to the dendrite of another neuron. An important characteristic of the synapse is that it is not a direct contact; rather, it is a temporary chemical link. Figure 4.2 shows a schematic representation of the

synapse.

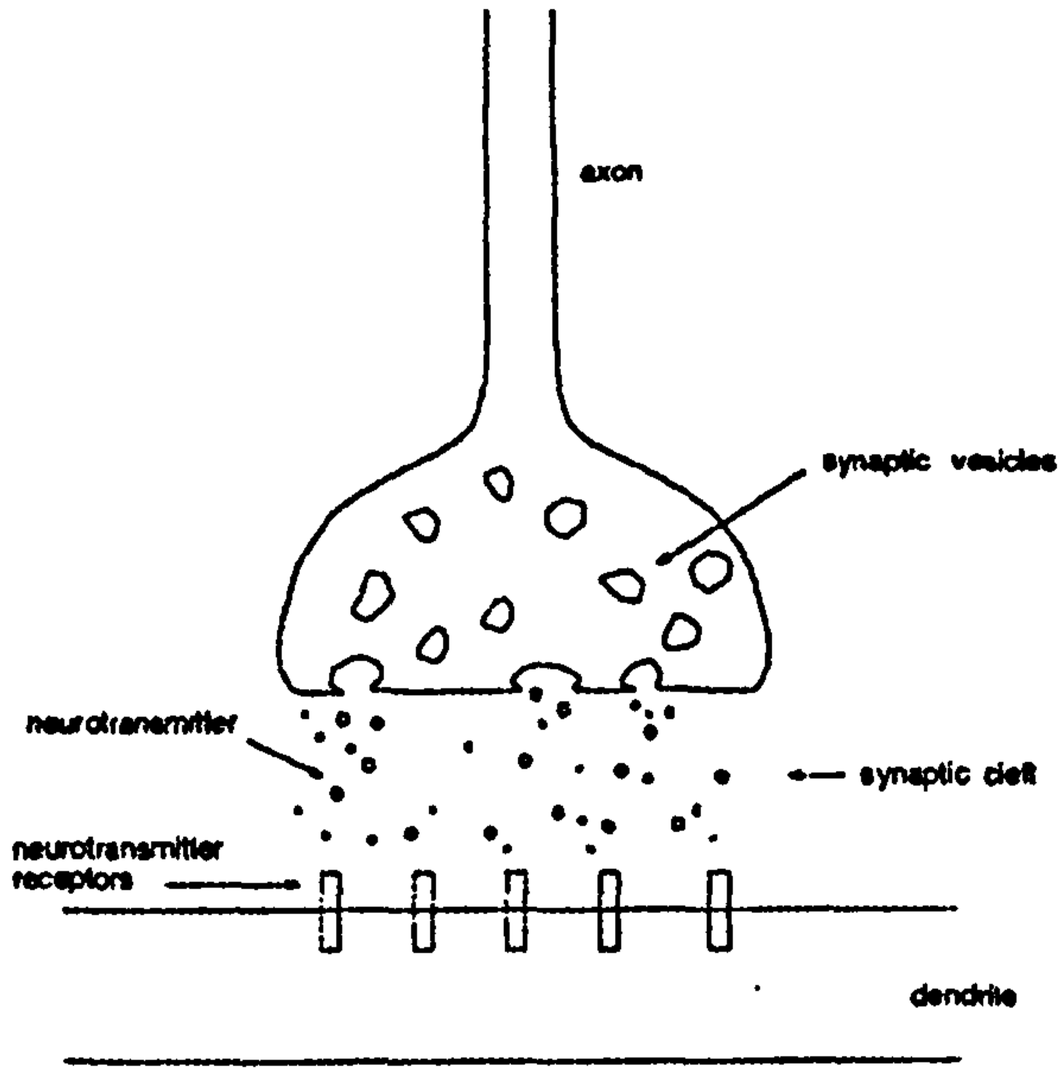


Figure 4.2 - Representation of a synapse (reproduced from [97] with permission)

The synapse releases chemical neurotransmitters whenever the action potential of the axon reaches a certain value. These neurotransmitters activate gates on the dendrites which when open allow charged ions to flow through and thus transmit a voltage pulse down the dendrite to the next neuron. Each neuron may receive several synaptic inputs through its dendrites and may have many synaptic outputs connecting it to other neurons. This is how massive interconnectivity is achieved within the brain.

Learning in the human brain is believed to occur through modifications in the strength of synaptic junctions, allowing more (or less) neurotransmitters to be released and thus providing a better (or poorer) coupling between neurons.

As the understanding of the structure and the functioning of the human brain is still mostly an unanswered question, the general adherence of any artificial model of the human brain is very poor. Therefore, it is essential to emphasise the small equivalence between biological and artificial neural networks. Despite this fact, artificial neural networks have achieved impressive results since the first attempts were made in the late 1940s.

Ideally, an artificial neural network should be capable of performing three main tasks,

namely *learning*, *generalisation*, and *abstraction*. Learning is the ability of an artificial neural network to modify its own behaviour in response to the environment the network interacts with. Generalisation can be seen as the ability to respond correctly, using past experience, to an input which the network has never seen before. Abstraction is the ability to extract the relevant features from an input environment that can be affected by particular variations such as external noise.

Artificial neural networks have long since been closely associated with the field of Pattern Recognition. This is due to the fact that both areas share, to a large extent, the same mathematical background and also deal with the same problems. Pattern recognition is an important area of computer science of which the main objective is classification, which can be formulated as: given any input, a meaningful categorisation for it is required.

The process of classification can be divided into two subproblems: feature extraction and classification. The former deals with the problem of recognising salient features from a complex input environment, i.e., the features that distinguish individual components of this environment. Once this is done, a classification task is performed using these salient features as classification criteria and using a suitable technique. Two well-known examples of pattern recognition applications are speech and image recognition.

Finally, just to mention some early successful applications of neural networks, Sejnowsky and Rosenberg [98] trained a network to convert text to phonetic representations; Burr [99] built up a network that can recognise hand-written characters; and Cottrell, Munro and Zipser [100] developed a neural network-based image compression system.

4.3 - Training artificial neural networks

Training is an essential step when setting up an artificial neural network (which will be simply referred to as "neural network" from now on). It means the procedure by which the neural network learns how to respond correctly to a certain input environment. The terms "training" and "learning" are interchangeable throughout this thesis.

Training can be of either *supervised* or *unsupervised* type. In supervised training, pairs of input and output vectors (i.e., sets of data which contain relevant information) are presented to the network and some internal parameters of the network are adjusted so as to produce an evaluated output as close as possible to the desired output. On the other hand, in unsupervised training, no output vectors are presented to the network; the network self-adjusts its internal parameters as the input vectors are presented.

The purpose and applicability of supervised and unsupervised training are different. Supervised training is well suited to the interpolation problem, when the network is required to produce the output for an input that it has never seen before. A good example of supervised training is the backpropagation algorithm, which is usually applied to the multi-layer perceptron network (MLP). Because of its non-linear nature, the MLP can be very accurate at producing complex mappings between input and output variables. In other words, it can provide a good model of a problem represented by a given training set (a set of input and output vectors that accurately represents the problem at hand). In doing so, the network allows the overriding of the mathematical formulation of the problem. This can be very important in cases where the direct application of mathematical tools is difficult or prohibitively time consuming. As will be seen in this work, the evaluation time required by an MLP network is nearly always negligible.

Networks trained through unsupervised procedures cannot perform interpolative calculations since they never see an example of desired outputs, and hence they do not know the relationship between input and output variables. Examples of these networks are the self-organising map and the ART architecture. The most important characteristic of these networks is perhaps their ability to extract relevant features from the input environment.

4.4 - Perceptrons and Multi-Layer Perceptrons (MLPs)

4.4.1 - Introduction

This section describes the Perceptron and the Multi-Layer Perceptron (MLP), the most popular paradigms of neural networks.

Perceptrons represent the first attempt to simulate biological neural networks, back in the 1940s. They are very limited in the sort of problems they can solve, and it is shown how they evolved into the more general model of Multi-Layer Perceptrons.

The training process for both Perceptrons and Multi-Layer Perceptrons is of the supervised type. The main training procedures available today for the MLP, namely the Backpropagation algorithm and Statistical Methods, are presented in some detail.

During the implementation of these training methods, some modifications to the basic backpropagation procedure were devised and tested. Such modifications are also described.

Finally, an illustrative example on the application of MLP is presented. More complex examples are also presented in chapters 5 and 6.

4.4.2 - Perceptrons and the linear separability problem

During the 1940s, McCulloch and Pitts [101, 102] were the first researchers to publish systematic work on the field of artificial neural networks. Much of their work was based on the perceptron model, Figure 4.3, in which a basic unit, also called neuron, receives many inputs and produces an output value.

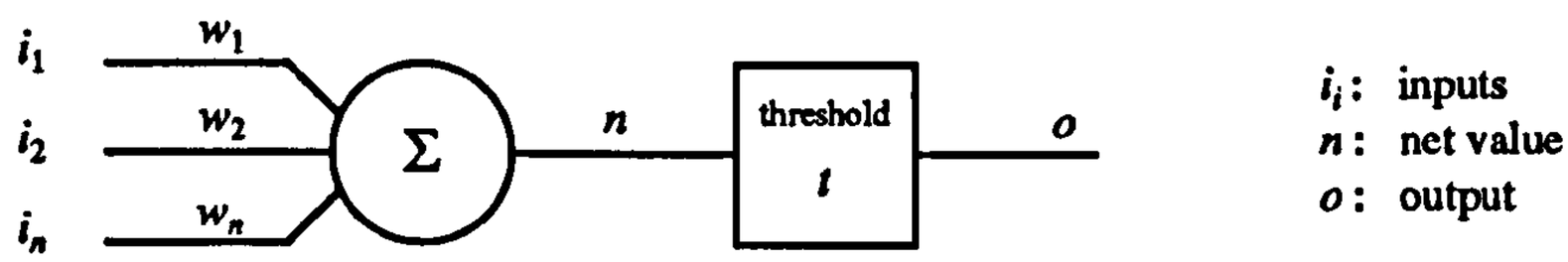


Figure 4.3 - Perceptron

The summation block (Σ) evaluates the weighted sum of all inputs, using weights w_i. This weighted sum, or net value (n), is then passed on to the threshold block. The output o from the threshold block (and from the perceptron itself) is 0 if the net value n is less or equal than a prespecified threshold value t; otherwise it is 1. This simple model sought to mimic some features of the biological neuron. In formal terms:

$$\begin{aligned} n &= \textstyle\sum_i w_i i_i && \text{(net value)} \\ o &= 0 && \text{if} \quad n \le t \\ o &= 1 && \text{if} \quad n > t \end{aligned} \tag{4.1}$$

One important variation of the basic perceptron is the multi-output perceptron, Figure 4.4, in which several basic neurons are arranged in a single layer.

It should be pointed out that in Figure 4.4, the leftmost circles represent fan-out devices, which only distribute the inputs and thus do not execute any mathematical operation. Each perceptron unit executes identical operations to the single output perceptron of Figure 4.3.

In 1962 Rosenblatt [103] proved the perceptron learning algorithm (see sub-section 4.4.4), which states that a perceptron could learn anything it could represent. At this point, it is essential to clarify the precise meaning of *representation* and *learning*. Representation refers to the ability of the perceptron to simulate a given function, whereas learning means the automatic procedure by

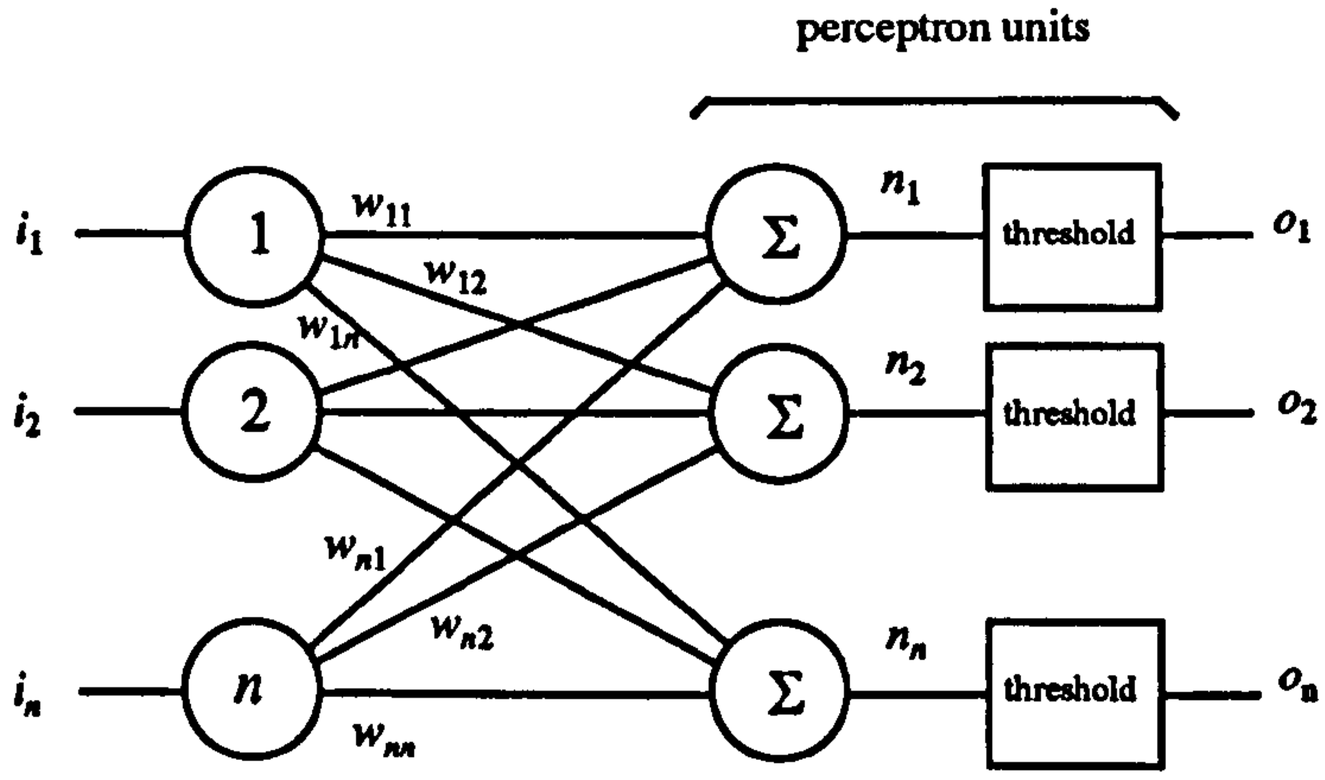


Figure 4.4 - Multi-output perceptron

which the perceptron acquires this ability through weight adjustment. A major goal when designing a training algorithm is to create automatic procedures that do not require the interference of human teachers. The theorem due to Rosenblatt was a milestone in the evolutionary process of the neural network field, and hence it produced great interest among scientific researchers.

Nevertheless, the initial optimism was replaced by disillusionment as perceptrons failed to solve certain simple problems. In 1969 Minsky and Papert [104] analysed this problem and proved that there are several restrictions on what a single-layer perceptron can represent, and therefore learn. They proved, among other things, that perceptrons cannot simulate the simple *exclusive-or* function.

Consider, for instance, the exclusive-or function, of which the truth table is presented in Table 4.1. Consider also, a 2-input, 1-output perceptron as represented in Figure 4.5. The analyti-

Inputs		Output
i_1	i_2	
0	0	0
0	1	1
1	0	1
1	1	0

Table 4.1 - Truth table for the exclusive-or function

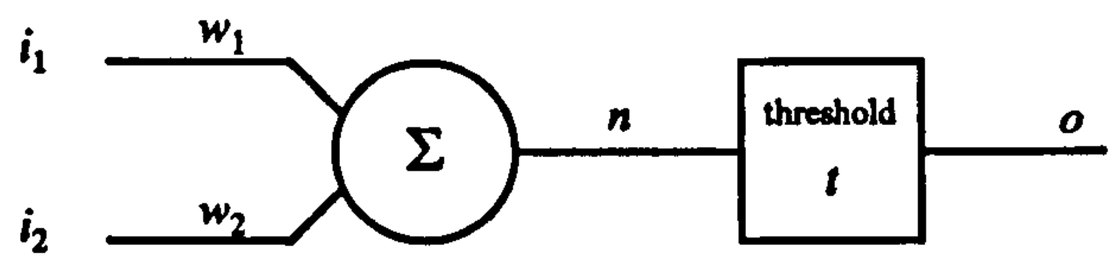


Figure 4.5 - 2-input, 1-output perceptron

cal equation for the perceptron in Figure 4.5 is as follows:

$$\begin{aligned} n &= w_1 i_1 + w_2 i_2 && \text{(net value)} \\ o &= 0 && \text{if} \quad n \leq t \\ o &= 1 && \text{if} \quad n > t \end{aligned} \tag{4.2}$$

Eq. (4.2) can be represented by a straight line which divides the i_1-i_2 plane into two different regions: the region for which the perceptron output is always 0 and the region where this output is always 1. Figure 4.6 illustrates this point.

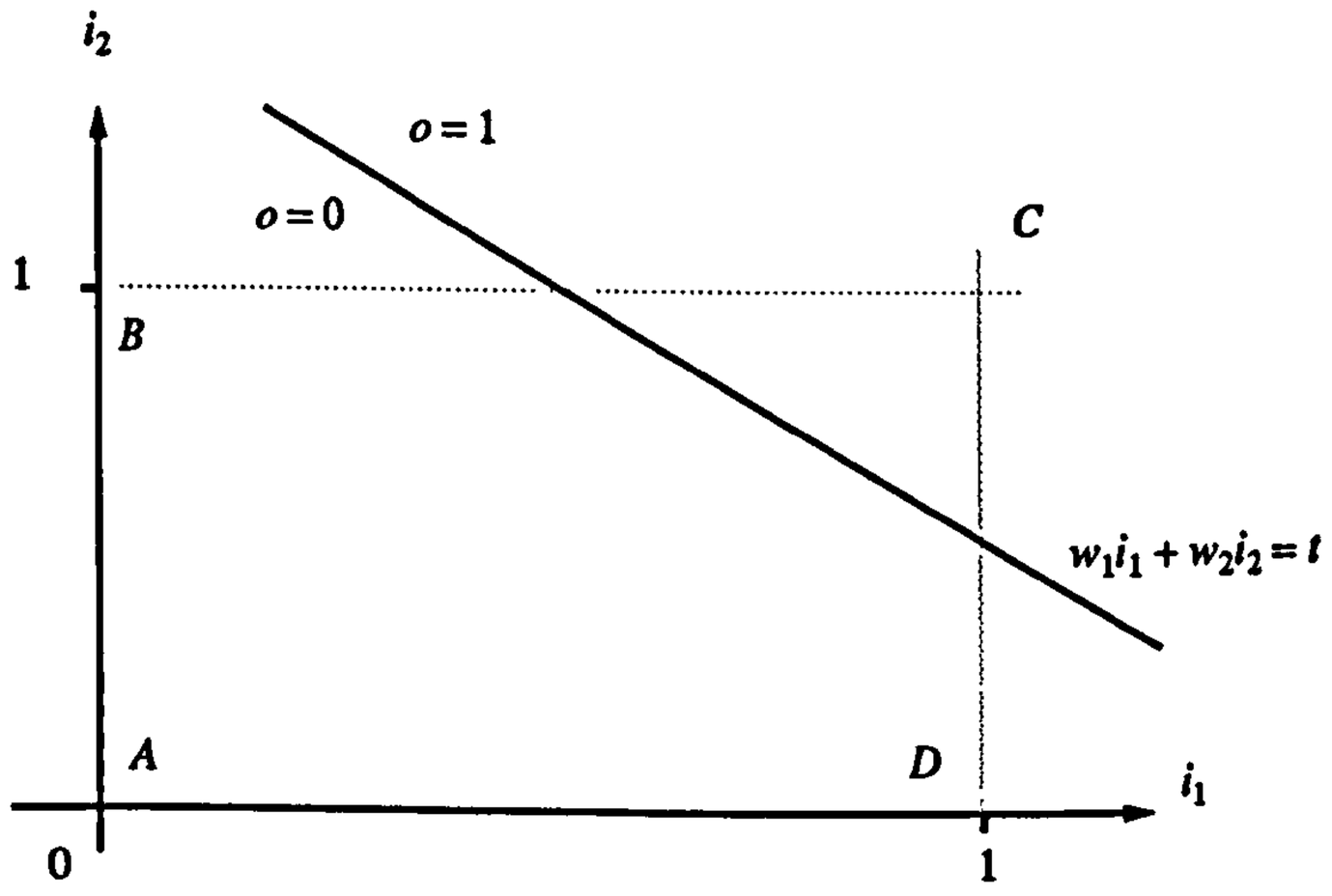


Figure 4.6 - Graphical representation of the output domain of perceptrons

Figure 4.6 also shows points A, B, C and D, which correspond to the definition of the exclusive-or function ($o = 0$ for points A and C, and $o = 1$ for B and D). From this figure it can be seen that the perceptron linearly classifies points in the 2-dimension input space into 2 categories: points below the straight line $w_1 i_1 + w_2 i_2 = t$ and points above this line. It is impossible, through this classifier, to separate points A and C from points B and D, in order to simulate the exclusive-or

function. In other words, the perceptron of Figure 4.5 cannot represent the exclusive-or function, and therefore it cannot learn how to represent this function.

Functions like the exclusive-or are called *linearly non-separable* and constitute an important subset of all functions. As an example, a perceptron with n binary inputs can have 2^n different input patterns. Each input pattern can produce two different binary outputs (0 or 1), and therefore there are 2^{2^n} different functions of n variables. Table 4.2 illustrates how the number of binary functions and the number of linearly separable functions grow as n is increased.

n	2^{2^n}	Nº of linearly separable functions
1	4	4
2	16	14
3	256	104
4	65 536	1 882
5	4.3 E09	94 572
6	1.8 E19	5 028 134

Table 4.2 - Number of binary functions and of linearly separable functions as a function of dimension n [105]

It is clear from Table 4.2 that the number of linearly separable functions relative to the total number of binary functions decreases drastically as the number of inputs is increased. Even the problem of determining how many linearly separable functions exist is a difficult one for large values of n.

4.4.3 - Multi-Layer Perceptrons

By the late 1960s, the linear separability problem associated with single-layer perceptrons was well understood. One solution for overcoming this problem was to construct 2-layer perceptrons. These networks may be formed by cascading two single-layer perceptrons, Figure 4.7.

The 2-layer network is capable of performing more general classifications, separating points which are contained inside and outside convex regions (a convex region is one in which any two points can be joined by a straight line which is completely contained within the region). For

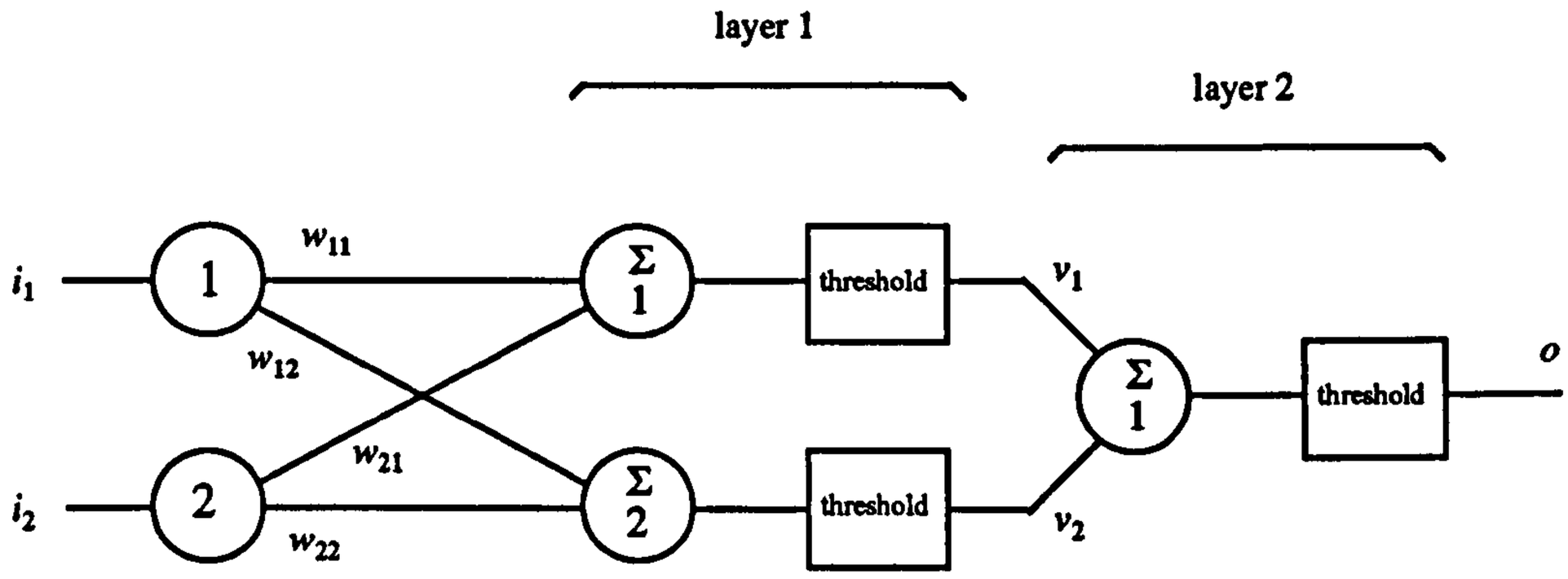
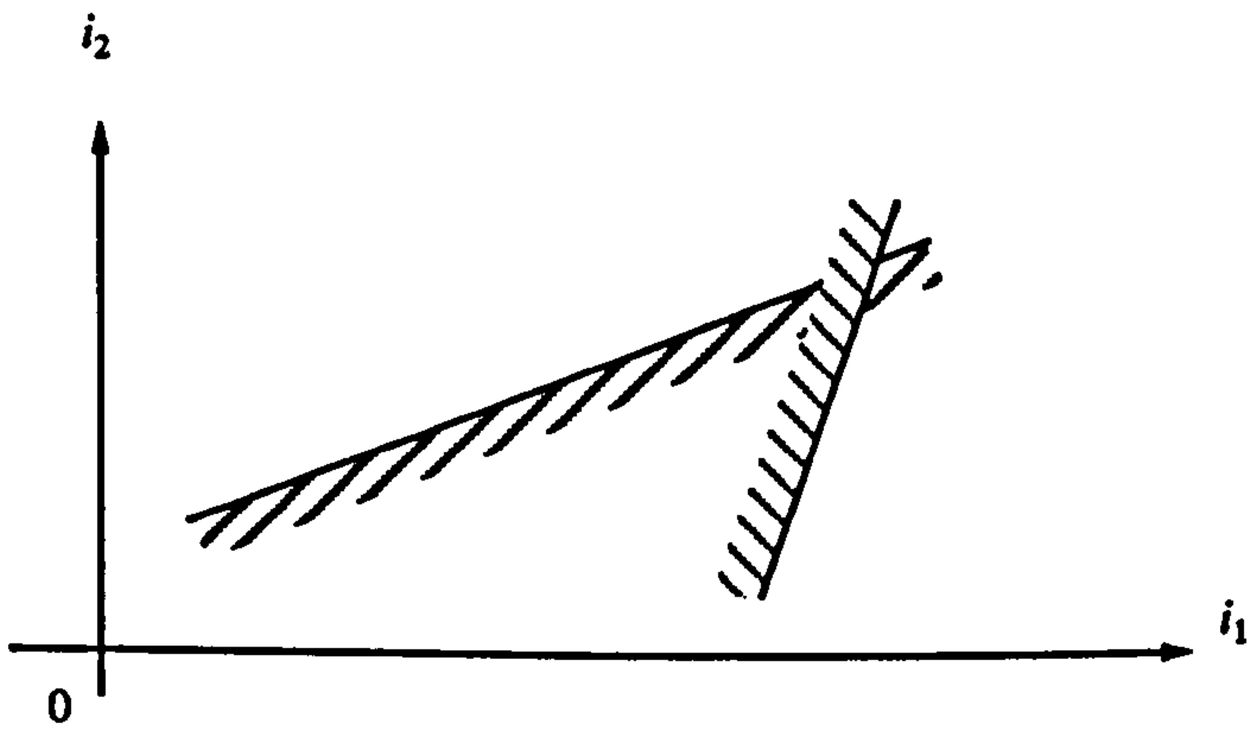


Figure 4.7 - 2-layer perceptron

instance, for the 2-layer perceptron of Figure 4.7, assume that the neurons in layer 1 classify points in the plane according to Figure 4.8 (a). Then, for the only neuron in layer 2, if weights v_1 and v_2 are set to 0.5 and the threshold is set to 0.75, this neuron will perform the logical "*and*" of its two inputs. The whole network is therefore capable of classifying points inside and outside the convex region illustrated in Figure 4.8 (b).

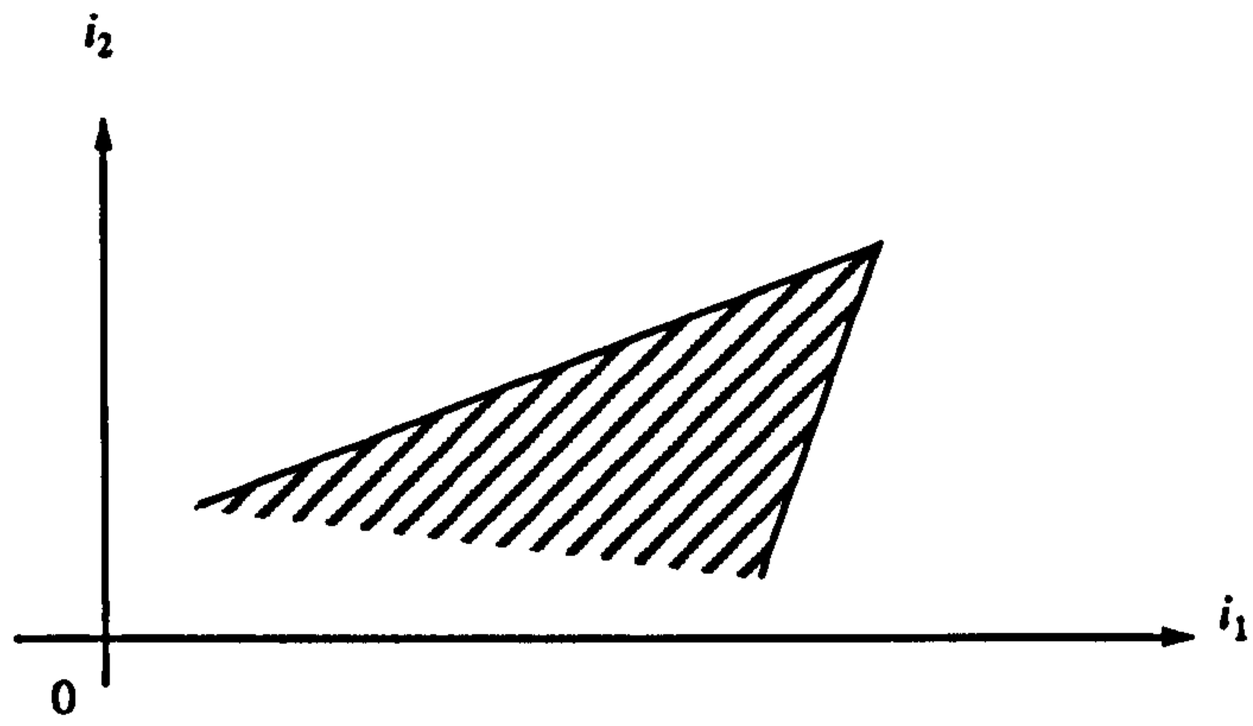


(a) - Classification of layer-1 neurons

Figure 4.8 - Classification capabilities of the 2-layer network of Figure 4.7

The convex region in Figure 4.8 (b) can be made a closed convex region simply by adding a third neuron to the first layer and resetting the weights and threshold of the single layer-2 neuron.

A 3-layer network is even more general. There are no convexity constraints since, for instance, two convex regions can be combined to form a non-convex region, Figure 4.9.



(b) - Convex region recognised by the network

Figure 4.8 - Classification capabilities of the 2-layer network of Figure 4.7

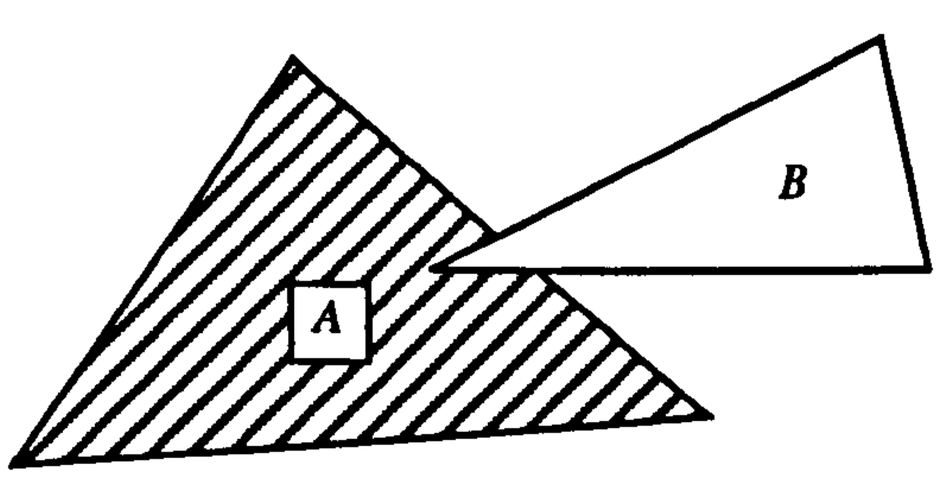


Figure 4.9 - Example of non-convex region (*A* and *not B*)

Despite the fact that multi-layer perceptrons are much more general than single-layer perceptrons, the multi-layered networks were of little use because for many years no training algorithm was available for this sort of networks.

4.4.4 - Perceptron training algorithm

Most of the training algorithms available today have evolved from the training procedure proposed by D. O. Hebb in 1961 [106]. This procedure is of the unsupervised type and it seeks to increase the synaptic link (weight) between two neurons if both neurons are activated. Symboli-

cally,

$$w'_{ij} = w_{ij} + \alpha \,.\, o_i \,.\, o_j \tag{4.3}$$

where w_{ij} = previous value of weight connecting neurons i and j;
w'_{ij} = new value for weight w_{ij};
o_i = output from neuron i;
α = training rate coefficient.

Eq. (4.3) is therefore known as Hebbian learning equation.

Rosenblatt [103] established the single-layer perceptron training algorithm in 1962, along with the proof that a perceptron can learn anything it can represent. Current training algorithms use, in one form or another, elements from this basic algorithm, which is described hereafter.

The perceptron training algorithm is an example of supervised training; output values for corresponding input vectors must be known in advance. The weights of the perceptron must be given initial values prior to training. Consider first of all a binary perceptron (inputs and outputs equal to 0 or 1 only), Figure 4.10.

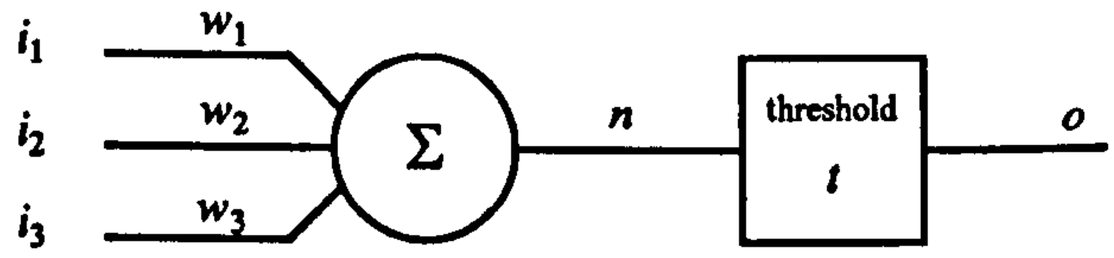


Figure 4.10 - Example of perceptron training

The basic idea is to sequentially present the perceptron with each input vector and evaluate the output using the current weights. This evaluated output may only be correct or incorrect. If it is correct, then nothing is done (because the perceptron has already learned the current input vector) and a new input vector is selected. If the evaluated output is incorrect, only the weights associated with the active inputs (the inputs of which the value is 1) will be modified as follows:

1. if the incorrect output is 0, then add each input to its corresponding weight;
2. if the incorrect output is 1, then subtract each input from its corresponding weight.

The process continues until no weight modifications are made after a complete sweep over the training set. In a finite number of steps, the perceptron will learn the entire training set, provided that it represents a linearly separable function.

Rule 1 above increases the strength of the connections between all active inputs and the perceptron, as the current input vector failed to produce a net value greater than the threshold value. Conversely, Rule 2 decreases the strength because the current input vector excessively excited the perceptron.

For continuous inputs and outputs, the so-called *Delta Rule* provides an important generalisation of the perceptron training algorithm. Defining d as the difference between the desired output (D) and the actual output (A) evaluated by the perceptron, the correction to be applied to a generic weight w_i is given by:

$$\Delta_i = \eta \,.\, \delta \,.\, x_i \qquad (4.4)$$
$$w_i(n+1) = w_i(n) + \Delta_i$$

where
- $w_i(n)$ = value of weight before correction;
- $w_i(n+1)$ = value of weight after correction;
- x_i = input i;
- Δ_i = correction to be applied to weight
- δ = D - A (difference between desired and actual outputs, or evaluation error);
- η = training rate coefficient, which allows the average size of weight changes to be controlled.

The Delta Rule automatically takes into account the polarity of the evaluation error (negative, zero or positive).

The main drawback of the perceptron training algorithm is that it does not indicate the number of necessary steps before convergence occurs. Also, it may be difficult to determine whether or not the training set being used represents a linearly separable function.

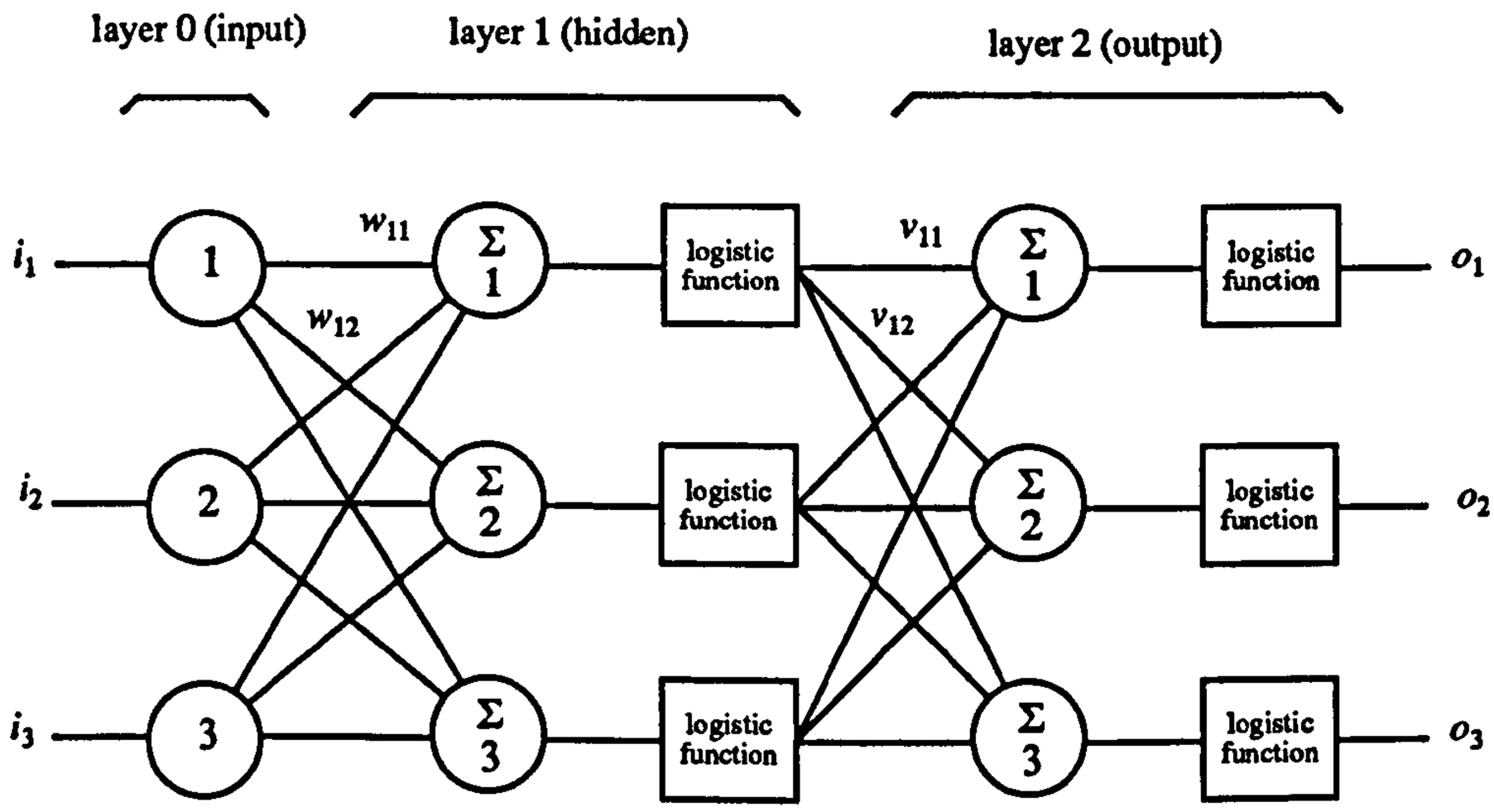
4.4.5 - Backpropagation

The backpropagation algorithm was the first systematic method for training Multi-Layer Perceptrons. The most recognised description of this algorithm was presented by Rumelhart, Hinton and Williams in 1986 [107], but soon afterwards it was found that it had already been described by Parker in 1982 [108] and Werbos in 1974 [109].

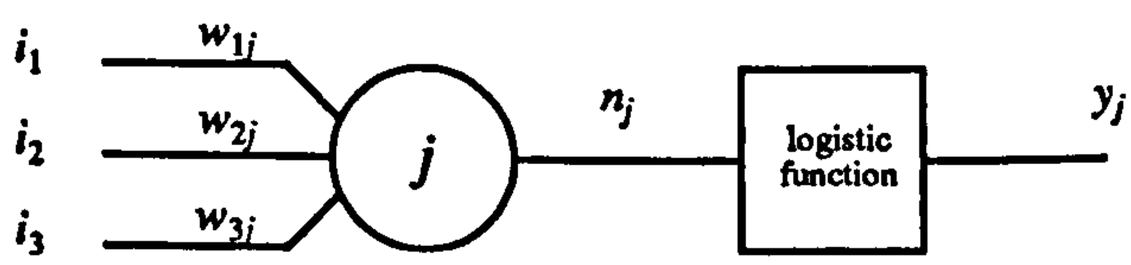
It should be pointed out that the original backpropagation algorithm has been modified and improved by many researchers over the past few years, so that nowadays the name "backpropagation" actually refers to a family of different algorithms. For this reason, only the basic backpropa-

gation procedure will be presented in detail here. A brief discussion on some of the major variations will be presented at the end of this sub-section.

In order to describe the backpropagation procedure, the 2-layer network of Figure 4.11 will be used (generalisation for the case of networks containing more than two layers is straightforward). A generic weight w_{ij} (or v_{ij}) connects the output from neuron i in a given layer to the input of neuron j in the next layer. Layers other than the input and output ones are called hidden layers.



(a) - 2-layer network



(b) - neuron j of layer 1

Figure 4.11 - 2-layer network and basic unit

In this case, neurons in layer 0 (the input layer) act only as fan-out devices, simply distributing the inputs i_1, i_2, ... into layer-1 neurons through the weights w_{11}, w_{12}, ... The basic unit (Figure 4.11 (b)) works in a slightly different manner to that of perceptrons described previously. The threshold function has now been replaced by another non-linear function. This non-linear function allows the constraints of linear separability and convexity to be overcome. Usually this function is chosen to be the logistic function given by Eq. (4.5).

$$y_j = \frac{1}{1 + e^{-\lambda n_j}} \tag{4.5}$$

where y_j = output from neuron j;
n_j = $\sum_i w_{ij} i_i$ (net input value);
λ = control parameter.

The logistic function (also known as *sigmoid* function) is a convenient choice because it is differentiable from $-\infty$ to $+\infty$ and its derivative, which is used by the backpropagation algorithm, is easily evaluated. The control parameter λ allows the steepness of the function to be controlled (for values of λ tending to infinity the function approaches the threshold function). It should be pointed out that layer 2 takes the outputs from layer-1 neurons as its inputs and performs the same operations as layer 1, using weights v_{ij} instead of w_{ij}. Finally, weights w_{ij} are considered to be layer-1 weights, whereas weights v_{ij} belong to layer 2.

The backpropagation algorithm is a supervised-type training procedure. Therefore, pairs of input and their corresponding output vectors are required. The macro-description of the procedure is as follows:

Step 1 Select the next training pair from the training set. Apply the input vector and evaluate the actual output vector;

Step2 Evaluate the error between the actual output vector and the training output vector (evaluation error);

Step 3 Using the evaluation error from Step 2, adjust all weights of the network so as to minimise this error;

Step 4 Repeat steps 1 to 3 for each pair in the training set (cycling over the entire training set more than once, if necessary) until the evaluation error for the whole training set is acceptably low.

Steps 1, 2 and 3 will be explained in greater detail below.

Step 1

The evaluation of the actual output vector is straightforward. An input vector is applied to the inputs of layer 0. The output of each neuron in layer 1 is evaluated as described previously (evaluation of the weighted sum $n_j = \sum_i w_{ij} i_i$ and application of the non-linear function to n_j). The outputs from layer 1 are the inputs to layer 2 and, for this new layer (and any subsequent layer), the outputs are evaluated in exactly the same way as for layer 1. Step 1 can therefore be seen as a "forward" step.

Step 2

For each neuron in the output layer the evaluation error is simply the difference between the actual value and the desired value from the training set.

Step 3

The weights of the network are adjusted according to their layer, starting from the output layer and moving backwards taking one layer at a time, until the input layer is reached. For this reason, Step 3 can be considered as a "backward step".

a) Adjusting the weights of the output layer

The weight w_{ij} between neuron j in the output layer and neuron i in the previous layer is adjusted according to:

$$\Delta w_{ij} = \eta \,.\, \delta_j \,.\, o_i \qquad (4.6)$$
$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}$$

where
- $w_{ij}(n)$ is the value of w_{ij} prior to adjustment;
- $w_{ij}(n+1)$ is the value of w_{ij} after adjustment;
- Δw_{ij} is the adjustment for weight w_{ij};
- o_i is the output from neuron i;
- δ_j = $o_j(1 - o_j)(d_j - o_j)$
- o_j is the actual output from neuron j;
- d_j is the desired output from neuron j (from training set);
- η is the training rate coefficient.

From Eq. (4.6), it can be seen that the weight adjustment for neurons in the output layer is calculated taking into account both the derivative of the logistic function $[o_j(1 - o_j)]$ and the evaluation error $[d_j - o_j]$. The training rate coefficient is usually set at values between 0.01 and 1.0 and it allows the control of the size of weight corrections.

b) Adjusting the weights of hidden layers

For hidden layers, there are no output values available in the training set, so the evaluation error must be calculated in a different way than for the output layer. Weight corrections for hidden layers are evaluated through Eq. (4.6) with the only difference being that the term δ_j is now evaluated according to:

$$\delta_j = o_j \,.\, (1 - o_j) \,.\, \sum_k \delta_k w_{jk} \qquad (4.7)$$

where
- o_j is the output from neuron j in the current hidden layer;
- k indicates any neuron in the subsequent layer (see Figure 4.12).

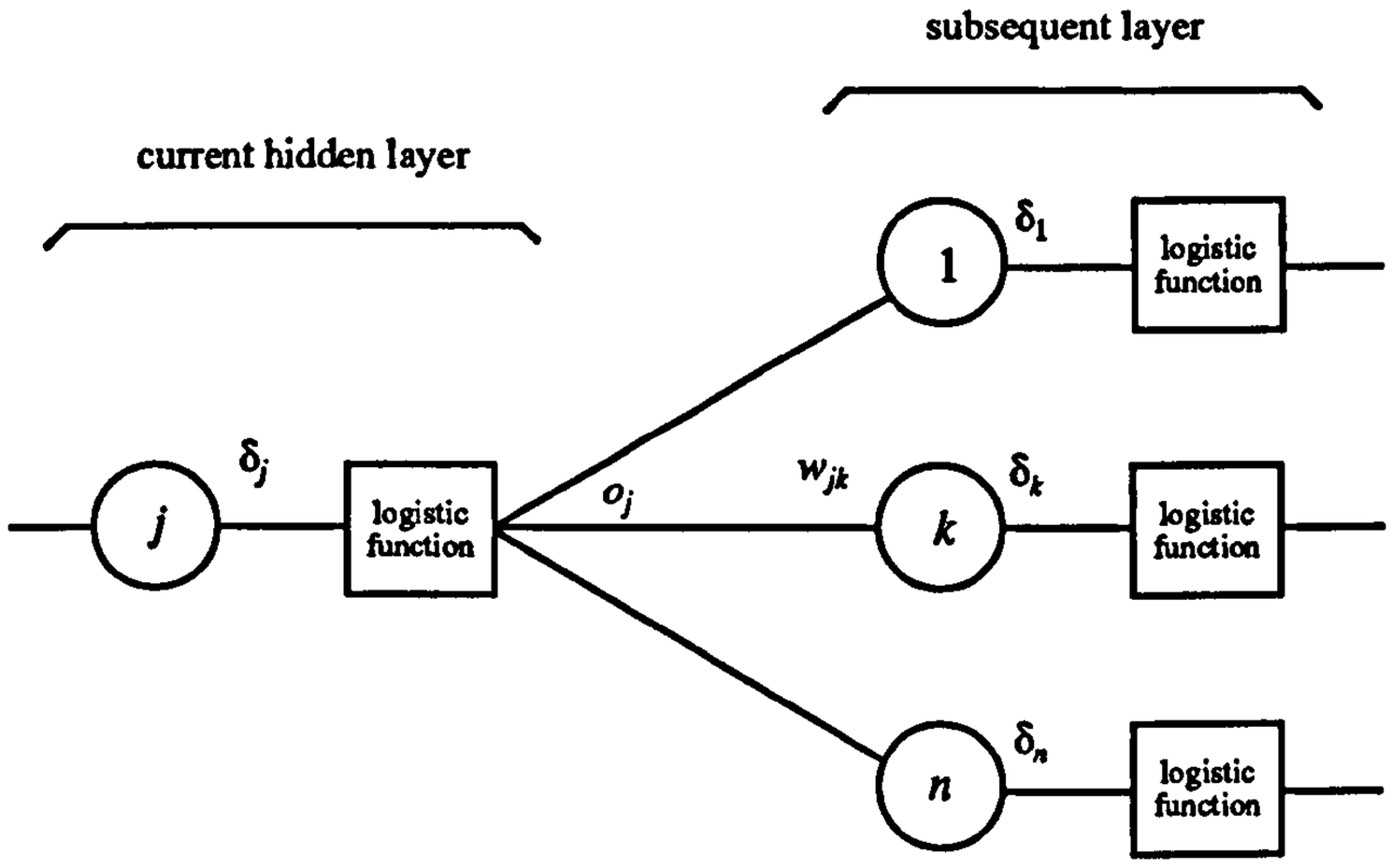


Figure 4.12 - Weight corrections for hidden layers

The value δ_k is already available as the subsequent layer was dealt with in the previous step. Eq. (4.7) is used to evaluate each δ_j in the current hidden layer, and these δ_j will be used again in the next step, when the layer previous to the current hidden layer will be considered.

This process is repeated, taking one layer at a time and moving backwards, until the input layer is reached. The evaluation error, calculated at the output layer, is transmitted through the network, thus allowing all weights to be adjusted accordingly.

Two of the main improvements to the basic backpropagation algorithm are the *momentum method* and the *exponential smoothing method.* The momentum method is also due to Rumelhart, Hinton and Williams [107] and it consists of adding an extra term to the weight corrections. This new term is proportional to the previous correction applied to the weights. Once a correction is made to a weight, it will be "remembered" in all subsequent corrections. The equations for the momentum method are as follows:

$$\Delta w_{ij}(n+1) = \eta \,.\, \delta_j \,.\, o_i + \alpha \,.\, \Delta w_{ij}(n) \qquad (4.8)$$
$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n+1)$$

where α is the momentum coefficient, usually set at values around 0.9. The momentum method normally improves the training process, but it may also have negative effects in some cases.

The exponential smoothing method was devised by Sejnowsky and Rosenberg [98] and it works in a similar manner to the momentum method. The equations are as follows:

$$\Delta w_{ij}(n+1) = (1-\alpha) \,.\, \eta \,.\, \delta_j \,.\, o_i + \alpha \,.\, \eta \,.\, \Delta w_{ij}(n) \tag{4.9}$$
$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n+1)$$

In this case, α is the exponential smoothing coefficient and it is set at values between 0 and 1, allowing the smoothing to be controlled. When $\alpha = 0$ the weight corrections are the new calculated values (minimum smoothing - exactly as the basic backpropagation procedure of Eq. (4.6)), and when $\alpha = 1$ the new adjustment is ignored and the previous one is repeated (maximum smoothing).

Although the backpropagation algorithm was a major milestone in the history of neural networks, leading to a renewed interest in the field and to many successful applications from the mid-1980s onwards, it also presents some important limitations that are discussed below.

The main difficulty is probably the local minima problem. Replacing the threshold function of perceptrons by the logistic function allowed the linear separability problem to be overcome, but, on the other hand, it makes the surface of the error function highly convoluted, full of hills and valleys. Very often the backpropagation procedure becomes trapped in a local minimum rather than moving towards the desired global minimum.

Network paralysis is also a problem associated with the backpropagation algorithm. If, during training, a weight becomes adjusted to a large value, the neuron may be operating at large values of output, in a region where the derivative of the logistic function is small. The error signal, proportional to this derivative, may also be too small, leading to very small weight corrections and ultimately to a standstill in the training process.

Temporal instability during training occurs when the network "unlearns" an already learned training vector because the weight adjustments for another training vector "erased" the knowledge the network had acquired.

Finally, a suitable step size for the weight corrections (which can be controlled through the training rate coefficient η) must be chosen. The convergence proof of Rumelhart, Hinton and Williams [107] assumes infinitesimally small weight adjustments, which implies infinite training time. In practice, a finite step size must be used. If the size step is too small, training can be extremely time consuming; if it is too large, paralysis or continuous instability can occur. Wasserman [110] described an adaptive step size algorithm which automatically adjusts the step size as the training proceeds.

4.4.6 - Statistical methods

Statistical methods can be used for both training a neural network and for evaluating the output of a previously trained network. Only the first case (training) will be discussed here. Using statistical methods for evaluating the output of a network is discussed by Hinton and Sejnowsky [111].

Training a neural network using a statistical method helps overcome one of the main problems of the backpropagation algorithm, namely the local minima problem. The general structure of statistical training is as follows:

Step 1 — Apply an input vector and calculate the evaluation error with respect to a previously known output vector;

Step2 — Select a weight at random and adjust it by a small random amount. If the adjustment decreases the evaluation error, keep this adjustment; otherwise discard it;

Step 3 — Repeat steps 1 and 2 until the network is trained.

The size of weight corrections is critical to the success of statistical training methods. If the size is too small and the network is already trapped in a local minimum, it may happen that the network stays at this minimum. If the size is too large, the training may not converge because many different points, optimal or not, will be visited during training. A common strategy for avoiding this problem is to start the training process with a large average step size and reduce it as training proceeds. This procedure is also referred to as *simulated annealing*. When a metal is heated above its melting point temperature, its atoms are in strong random motion. If the metal is allowed to cool down, it will eventually reach the minimum energy state. The distribution of energy states during this annealing process is given by:

$$P(x) \propto e^{\frac{-x}{kT}}$$

where $P(x)$ is the probability that the system is in a state with energy x;
k is the Boltzmann's constant;
T is the absolute temperature (degrees Kelvin).

At high temperatures $P(x)$ approaches 1 for all energy states (regardless of whether they are low energy or high energy states). As the temperature decreases, the probability of a high energy state is reduced in comparison with the probability of a low energy state.

Szu and Hartley [112] developed a statistical method for training neural networks as described above, using Cauchy distribution for evaluating the step size. A brief description of the method is presented below:

Step 1 Define a variable T that represents an artificial temperature. Initial values of T must be "large";

Step 2 Apply an input vector to the network and calculate the evaluation error;

Step 3 Evaluate a random weight change using Cauchy distribution and recalculate the evaluation error:

$$\Delta w = \eta \,.\, T \,.\, \tan[P(\Delta w)] \tag{4.10}$$

where Δw = weight change;
η = training rate coefficient;
T = current temperature;
$P(\Delta w)$ is the probability of a step size Δw.

$P(\Delta w)$ is evaluated through the Monte Carlo method: select a random number from a uniform distribution over the open interval $[-\frac{\pi}{2},\frac{\pi}{2}]$ and substitute it for $P(\Delta w)$;

Step 4 If the evaluation error is reduced, retain the weight change. Otherwise, calculate the probability of accepting this weight change (which increased the evaluation error) using Boltzmann distribution:

$$P(x) = e^{\frac{-x}{kT}} \tag{4.11}$$

where $P(x)$ is the probability of a change x in the evaluation error;
k is a constant analogous to Boltzmann's constant;

Select a random number r from a uniform distribution between 0 and 1. If $P(x)$ is greater than r, retain the weight change; otherwise return the weight to its previous value.

Steps 3 and 4 must be repeated for each weight in the network, gradually reducing the temperature T until an acceptably low value for the evaluation error is obtained. This process must then be extended to all input vectors in the training set, cycling over the entire set if required. It should be noted that Step 4 allows the system to take occasional steps in directions that increase the evaluation error, and hence helping the network to escape a local minimum.

The statistical method described above helps avoid local minima during the training of a network, but it can take many times the average training time of backpropagation to converge. Another problem associated with this method is the finite probability of applying extremely large weight corrections (due to the Cauchy distribution in Step 3), which can lead to network paralysis during training.

Wasserman [113] proposed a combined backpropagation/statistical method which seeks to combine the advantages of both methods. The structure of this combined method is the same as the pure statistical method described above. The only difference lies in the calculation of the weight corrections, which now consist of two components: a directed component calculated using the backpropagation algorithm and a random component calculated using Cauchy distribution. Both components are summed to produce the total weight change:

$$\Delta w_{ij}(n+1) = \beta . [(1 - \alpha) . \eta . \delta_j . o_i + \alpha . \eta . \Delta w_{ij}(n)] + (1 - \beta) . \Delta w \qquad (4.12)$$

where β is a coefficient controlling the contribution of both components (for $\beta = 0$ the method becomes purely statistical and for $\beta = 1$ it becomes purely backpropagation) and all other symbols have the same meaning as in Eqs. (4.9) and (4.10).

According to Wasserman [96], changing one weight at a time may be inefficient, so he suggests that all the weights in a layer be changed together before recalculating the output vector.

The possibility of applying very large corrections to the weights, leading to network paralysis, still exists in this combined method. Wasserman [96] suggests a corrective action to avoid this problem, by applying the logistic function to the weights themselves. According to this solution, whenever the output signal from a neuron exceeds some pre-defined minimum or maximum value (indicating neuron saturation), all the weights feeding that neuron are operated upon by the logistic function:

$$w'_{ij} = -5 + \frac{10}{1 + e^{-\frac{w_{ij}}{5}}}$$

where w_{ij} and w'_{ij} are respectively the old and the new values for the weight. The constants 5 and 10 were arbitrarily chosen, so other values may well be used. This procedure effectively reduces the magnitude of large weights, while leaving small weights nearly intact. Good results are reported with this procedure, although training times can still be long.

4.4.7 - Modified algorithm

The flow chart in Figure 4.13 depicts a modified algorithm, which was devised by the author during the implementation of the backpropagation procedure. It can be seen that there is an internal iteration loop so that the network can be trained with the same training vector for more than one single pass. It was found that this feature allowed the network to learn the current training vector more quickly, and it improved the overall training performance. As a rule of thumb, values for the maximum number of internal iterations may be set between 3 and 10.

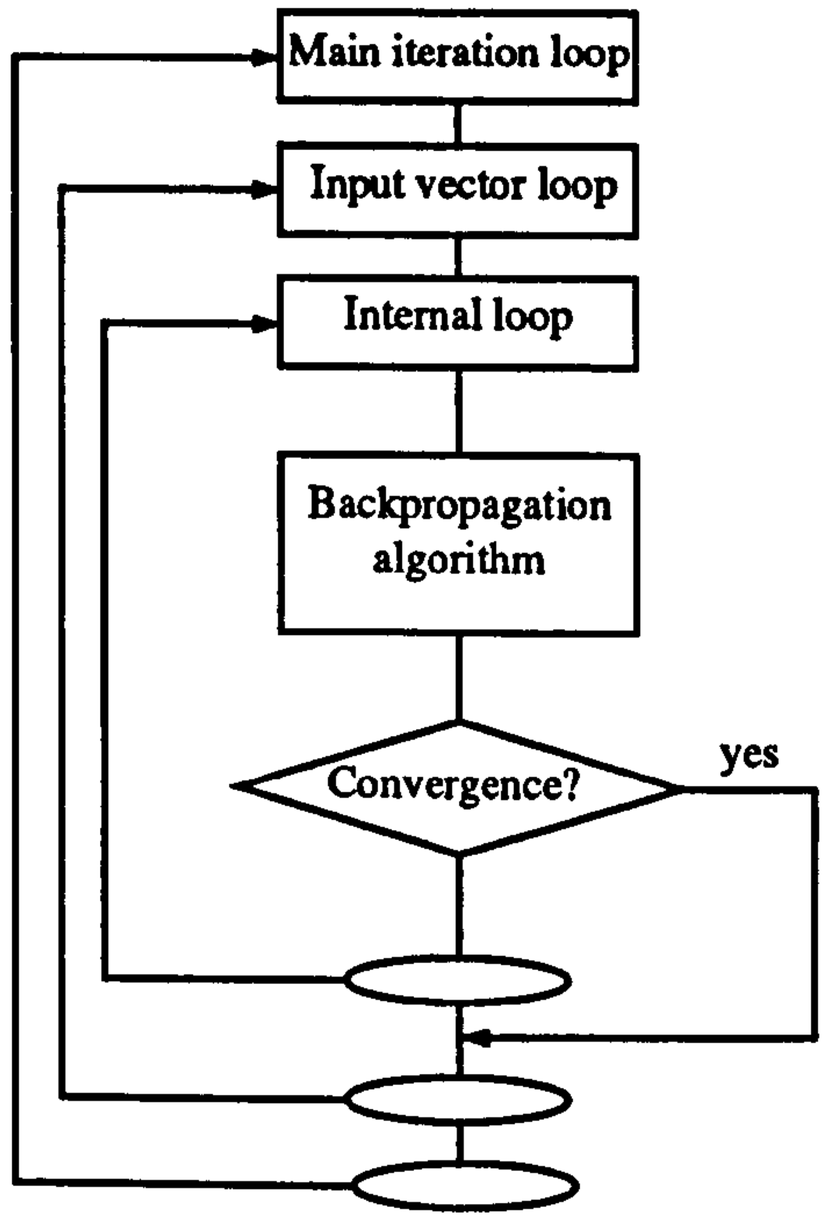


Figure 4.13 - Modified backpropagation procedure

Another special feature incorporated into the training process is the scaling of both input and output variables. As for the input variables, the purpose of the scaling is to prevent the saturation of the neurons due to high input values. All the input values are linearly mapped onto a new interval, usually ranging from -10 and +10. The scaling of the output variables has a twofold objective. First, it allows the output variables to be represented in any range rather than the [0, 1] interval which is the output mapping domain of the logistic function. Second, it helps avoid the "dead zone" near the output values 0 and 1. In these regions, the derivative of the logistic function approaches 0, thus making it difficult for the network to learn values near 0 or 1. The output values may be mapped, for instance, onto the interval [0.3, 0.7].

4.4.8 - Example

In this section, results obtained with a simple MLP network trained to simulate the exclusive-or function are presented. It should be noted that more complex cases are discussed in detail in chapters 5 and 6.

Tables 4.3, 4.4 and 4.5 show the main data, details of training, and results for this case, respectively.

Data item	Value		
Nº of layers	2		
Nº of neurons in each layer	Input	Hidden	Output
	2	5	1
Total number of weights	2x5 + 5x1 = 15		
Initial value for weights	random values between -5 and +5		
Generation of training set	manual		
Nº of training vectors	4		
Coefficient λ (see Eq. (4.5))	1.0		

Table 4.3 - Main data for example of MLP network

4.5 - Counterpropagation networks

4.5.1 - Introduction

The counterpropagation network (CP) is a hybrid model developed by R. Hecht-Nielsen [114-116]. It is a combination of the self-organising map (Kohonen [117]) and the outstar (Grossberg [118-120]). The CP network is simpler and more limited than the multi-layer perceptron, but its training is usually fast (it can be much faster than that of MLP) and it also forms a good statistical model of the input set.

Two versions of the CP network will be discussed in the following sections, namely the feedforward CP network and the full CP network. Finally, an example of the application of CP networks will be presented.

Training procedure	Pure backpropagation with exponential smoothing $\alpha = 0.95$; $\eta = 1$ (see Eq. (4.9))	
Number of internal iterations in each external iteration (see section 4.4.7)	External iteration	Nº of internal iterations
	1	370
	2	111
	3	121
	4	101
	5	71
	6	6
	7 (convergence)	0
Total training CPU time (Sun Sparc 2 workstation): 0.2 s		

Weights (final value)	Layer 1	-5.16000	-3.89813	2.41506	-2.59550	4.70345
		2.51737	3.35512	-2.77765	-3.18376	-3.44797
	Layer 2	4.23531	-3.13070	2.93618	-4.08047	-1.56317

Table 4.4 - Details of training for example of MLP network

Input		Output	
i_1	i_2	Desired	Evaluated
0	0	0	0.02430
0	1	1	1.00426
1	0	1	0.98816
1	1	0	-0.02397

Table 4.5 - Results for example of MLP network

4.5.2 - The feedforward counterpropagation network

Figure 4.14 presents the feedforward version of the CP network. As with the MLP, the purpose of layer 0 is only to distribute the inputs to all neurons in layer 1 (also called Kohonen layer) through weight matrix *W*. Layer 0 therefore does not perform any arithmetic operation. The weight matrix *V* connects the Kohonen layer to layer 2 (also called output layer or Grossberg layer). The outputs of the network are the outputs from Grossberg-layer neurons.

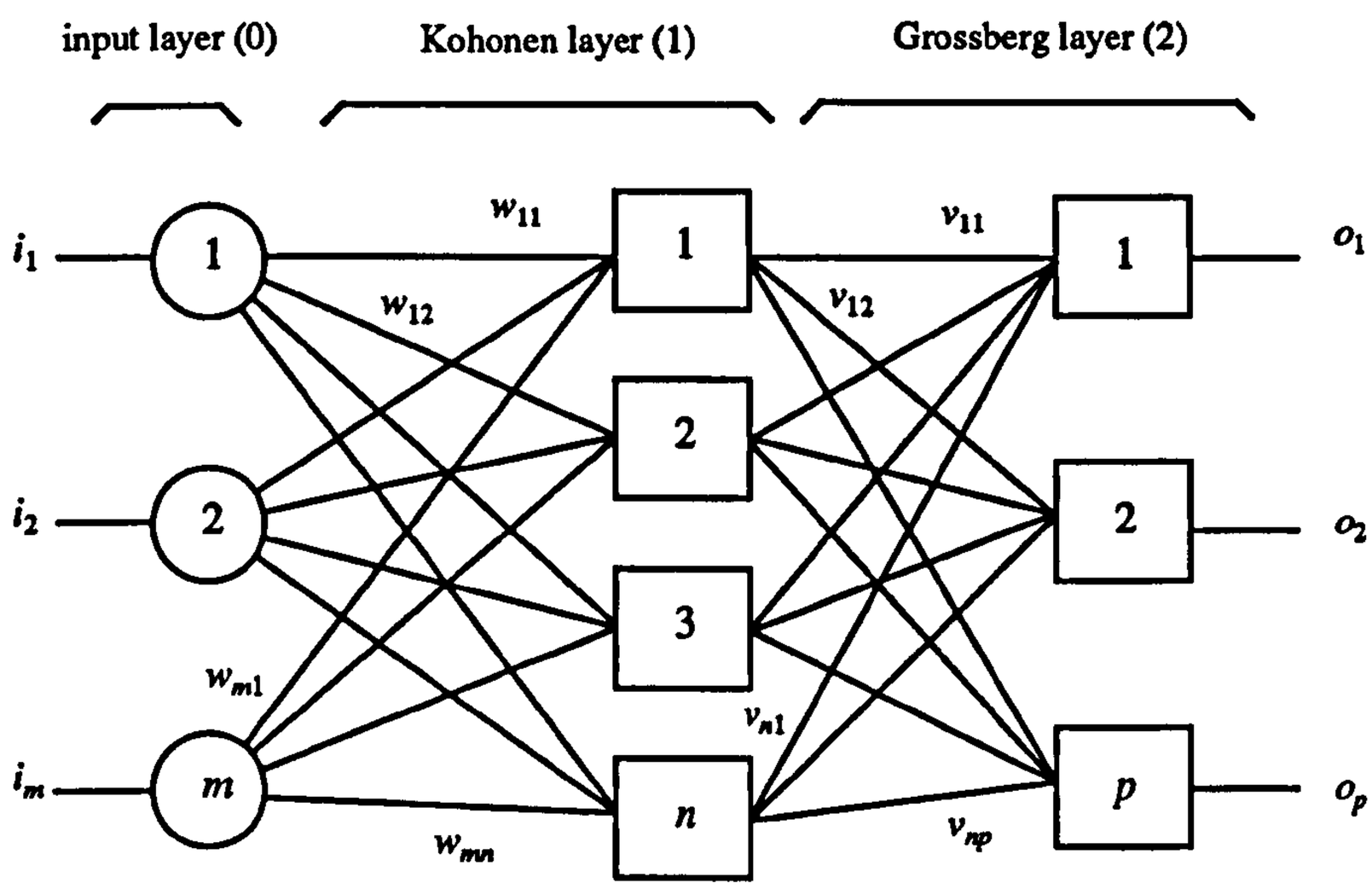


Figure 4.14 - Feedforward version of CP networks

Neurons in Kohonen layer evaluate the weighted sum of the inputs, using matrix *W*. But only the Kohonen neuron with the largest output is allowed to fire; its output is set to 1 and the output of all other neurons is set to 0. The Grossberg layer, receiving an input vector which is 1 at one location, simply reproduces the weight vector $[v_{j1}\ v_{j2}\ ...\ v_{jp}]$ associated with the winning Kohonen neuron *j*.

The main purpose of the Kohonen layer is to find out, from among the column vectors W_1, W_2, ..., W_n associated to the Kohonen-layer neurons, the most similar vector to the input vector being presented to the network (the dot product between these weight vectors and the input vector is an adequate measure of similarity). Once the best match is found, the Grossberg layer produces the output vector associated with this best match. Therefore, the Kohonen layer executes the recognition task and the Grossberg layer translates the input information into the output mapping domain.

When only one Kohonen neuron is allowed to fire, as explained above, the network is said to operate in the *accretive* mode. It is also possible to allow more than one Kohonen neuron to fire at the same time, when the network operates in the *interpolative* mode. Considering that *l* Kohonen neurons are allowed to fire at the same time ($1 \leq l \leq n$), the operation of the counterpropagation network is described by the following equations:

a) Kohonen layer: $N = I \,.\, W$ (4.13)

b) Grossberg layer: $O = N' \,.\, V$

where $I = \begin{bmatrix} i_1 & i_2 & \cdots & i_m \end{bmatrix}$ input vector;

$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$ Kohonen layer weight matrix;

$N = \begin{bmatrix} n_1 & n_2 & \cdots & n_n \end{bmatrix}$ output vector from Kohonen layer;

$N' = \begin{bmatrix} n_1 & 0 & \cdots & n_n \end{bmatrix}$ same as N, but with the largest l values only;

$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1p} \\ v_{21} & w_{22} & \cdots & v_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ v_{n1} & v_{n2} & \cdots & v_{np} \end{bmatrix}$ Grossberg layer weight matrix;

$O = \begin{bmatrix} o_1 & o_2 & \cdots & o_p \end{bmatrix}$ output vector.

The training of the counterpropagation network is carried out for Kohonen and Grossberg layers separately. Kohonen-layer training is of the unsupervised type. An input vector is applied and its dot product with each column of matrix W is evaluated. The weights associated with the winning neuron(s) (those with largest dot product) are modified according to:

$$w'_{ij} = w_{ij} + \alpha(i_i - w_{ij}) \quad (4.14)$$

where
- i = i^{th} input ($i = 1, 2, \ldots, m$);
- j = index indicating the winning Kohonen neuron(s);
- i_i = input i;
- w_{ij} = weight(s) connecting input i to winning Kohonen neuron(s);
- w'_{ij} = new value for w_{ij};
- α = training rate coefficient.

All remaining weights are kept unchanged. From Eq. (4.14) it can be seen that the training makes the column vector $[w_{1j}\ w_{2j}\ ...\ w_{mj}]'$ more similar to the input vector.

Grossberg-layer training is of the supervised type. Once the winning Kohonen neurons have been found (neurons *j*) and matrix *W* has been updated, the weight connecting the Kohonen neurons *j* to all the outputs are modified according to:

$$v'_{jk} = v_{jk} + \beta(y_k - v_{jk})n'_j \qquad (4.15)$$

where
- k = $1, 2, ..., p$
- j = index indicating the winning Kohonen neuron(s);
- n'_j = output of Kohonen layer neuron(s) j;
- v_{jk} = weight connecting Kohonen neuron j to Grossberg neuron k (old value);
- v'_{jk} = new value for v_{jk};
- y_k = desired output value for Grossberg neuron k;
- β = training rate coefficient, usually set to approximately 0.1 and reduced as training proceeds.

The training phase is carried out taking one training pair at a time and repeating the whole cycle over the training set if necessary. Initial values for weights in matrices *W* and *V* must be given prior to the start of network training. A common method for initialising weights is to set them at random values. Other methods of weight initialisation and also variations on the training procedure are discussed in [96].

It should be pointed out that normalisation of certain vectors to unity length is highly convenient because it can greatly improve both processing and training phases. The vectors that should be normalised are as follows: input vectors (*I*), output vector from Kohonen layer after modification for interpolative mode (*N'*), output vectors (*O*), column vectors from matrix *W* (initial value) and row vectors from matrix *V* (initial value).

4.5.3 - The full counterpropagation network

Figure 4.15 presents an example of a full counterpropagation network. During training, vectors *X* and *Y* are both presented as inputs and the network produces the corresponding vectors *X'* and *Y'* (both *X* and *Y* are considered to be normalised vectors). The training process modifies the weights so as to minimise the difference between *X* and *X'*, and *Y* and *Y'* (i.e., to produce an identity mapping of (*X*,*Y*) onto (*X'*,*Y'*)). In the processing mode, applying only the *X* inputs to the network (with all *Y* inputs set to zero) will still produce *X'* and *Y'* outputs. The same occurs when only the *Y* inputs are applied (with all *X* inputs set to zero). If a function *F* maps *X* onto *Y*, then the

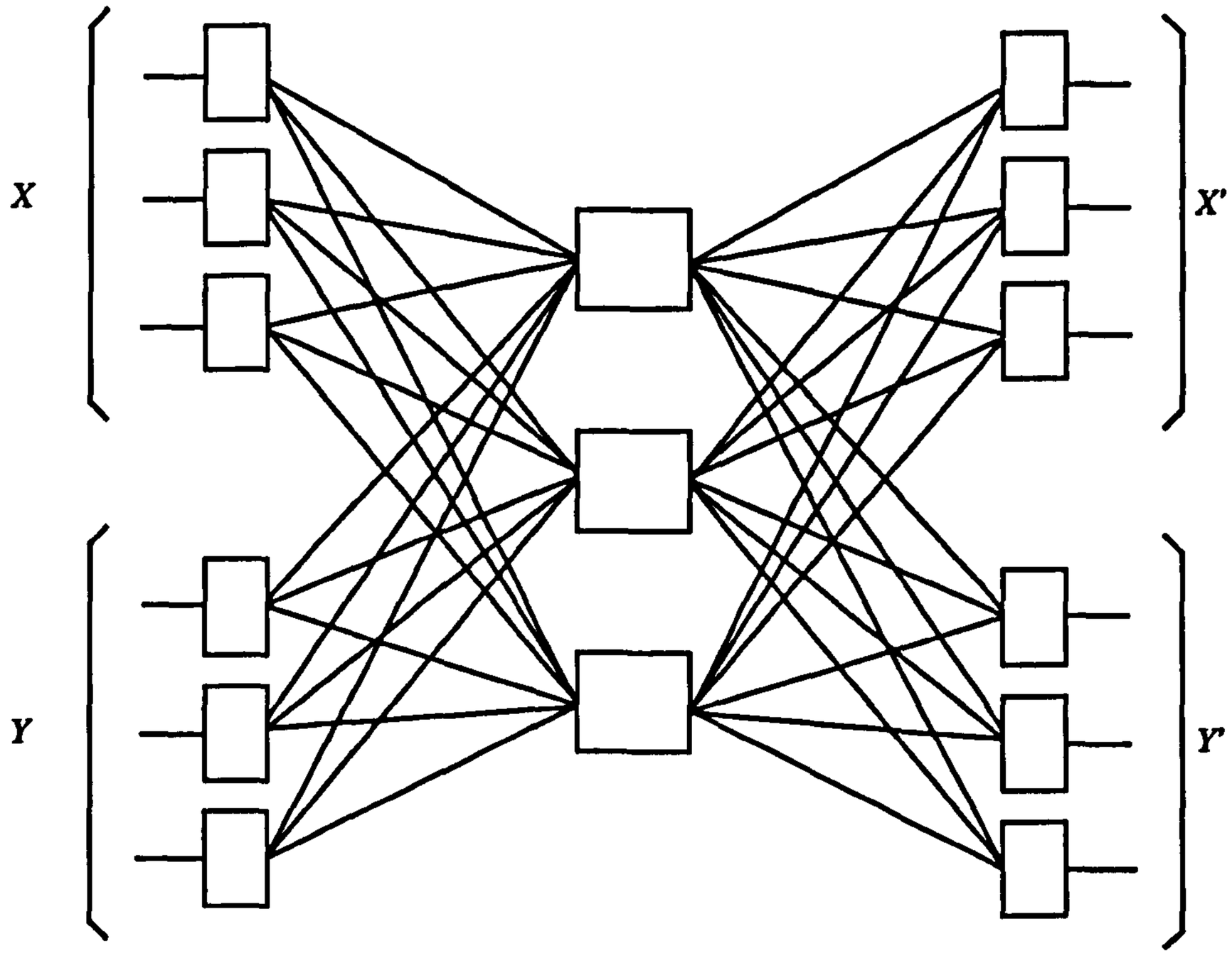


Figure 4.15 - Full counterpropagation network

full counterpropagation network is capable of simulating both *F* and its inverse. This ability to represent a function and its inverse can be useful in some applications.

4.5.4 - Example

As stated earlier, counterpropagation networks are capable of forming a good statistical model of the input environment. Therefore, a statistically significant training set is required for realising the full potential of the CP network. For illustration purposes, only a small example of the full counterpropagation network is presented here.

The training set under consideration consists of twelve 4-element vectors. The first two components of each vector represent a complex number in the complex plane (rectangular representation). The last two components represent a second complex number which is rotated 30 degrees anticlockwise with respect to the first complex number. Weight matrices were initialised with random values.

Tables 4.6 and 4.7 present the main data and training vectors respectively for this example.

Data item	Value
Dimension of vectors X and X'	2
Dimension of vectors Y and Y'	2
Nº of Kohonen-layer neurons	60
Nº of training vectors	12
Operating mode	accretive
α (see Eq. (4.14))	0.7
β (see Eq. (4.15))	0.1

Table 4.6 - Main data for example of CP network

Training vectors				Committed K-neuron (after training)
i_1	i_2	i_3	i_4	
1.	0.	0.96593	0.25882	43
0.86603	0.50000	0.70711	0.70711	49
0.50000	0.86603	0.25882	0.96593	36
0.	1.	-0.25882	0.96593	4
-0.50000	0.86603	-0.70711	0.70711	12
-0.86603	0.50000	-0.96593	0.25882	8
-1.	0.	-0.96593	-0.25882	28
-0.86603	-0.50000	-0.70711	-0.70711	59
-0.50000	-0.86603	-0.25882	-0.96593	29
0.	-1.	0.25882	-0.96593	57
0.50000	-0.86603	0.70711	-0.70711	23
0.86603	-0.50000	0.96593	-0.25882	7

Table 4.7 - Training vectors and corresponding K-neurons

Table 4.8 shows the response of the trained network to some incomplete vectors. It can be seen that the network is capable of correctly recalling full vectors when presented with incomplete

vectors (vectors 1 to 3). Also, it can be seen that the last input vector recalled an untrained Kohonen neuron, thus producing a meaningless output. This underlines the need to use an adequate training set and for ensuring a proper match between the training set and the number of Kohonen neurons.

Testing vectors				K-neurons recalled	Training vectors recalled			
i_1	i_2	i_3	i_4		i_1	i_2	i_3	i_4
1.	0.	0.	0.	43	1.	0.	0.96593	0.25882
0.86603	0.50000	0.	0.	49	0.86603	0.50000	0.70711	0.70711
0.	0.	-0.25882	0.96593	4	0.	1.	-0.25882	0.96593
0.79863	0.60182	0.	0.	46 (uncommitted)	0.30387	0.95271	0.97390	0.22699

Table 4.8 - Results for example of CP network

4.6 - Hopfield networks

4.6.1 - Introduction

Hopfield networks belong to a broader class of artificial neural networks, namely the recurrent networks. Recurrent networks are characterised by feedback connections between the outputs and the inputs of the network. They present some advantages and some disadvantages when compared to feedforward networks, which do not have feedback connections.

In the next sections, a brief discussion on recurrent networks will be presented, followed by the description of an example of the Hopfield network. Finally, the main characteristics of Hopfield networks will be presented through an example.

4.6.2 - Recurrent networks

Figure 4.16 presents an example of a recurrent network.

The behaviour of a recurrent network is dynamic because of the feedback connections. This

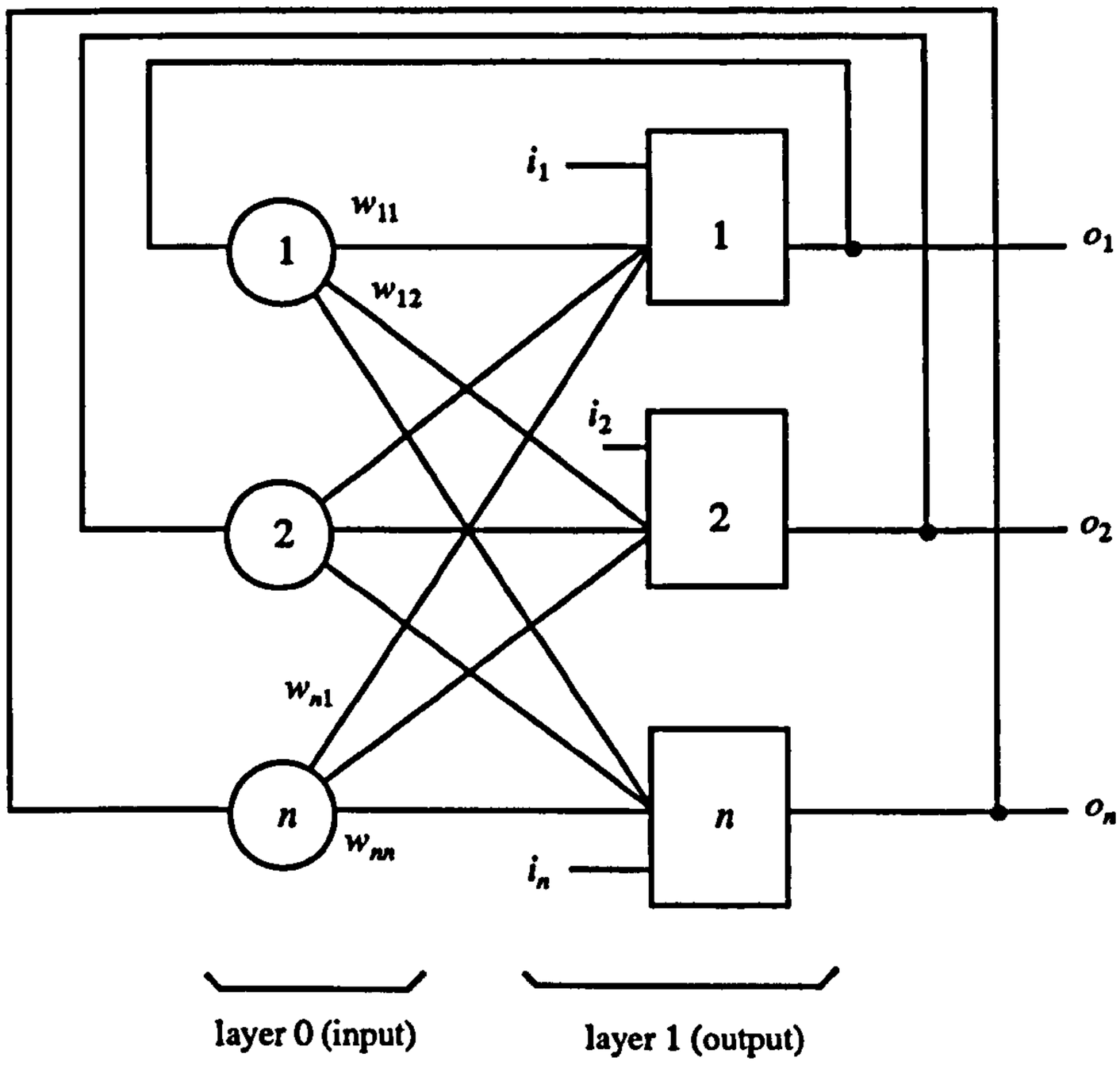


Figure 4.16 - Recurrent network

fact allows this type of network to exhibit a less limited behaviour than its feedforward counterparts. For instance, recurrent networks can simulate the operation of an associative memory, which means that the network can produce a correct output even when an incorrect or incomplete input vector is applied. Because human memory often operates in associative manner, recurrent networks provide a more accurate model of this phenomenon.

In normal operation, the outputs of the network are set equal to a given input vector. Then the input vector is removed and the outputs are fed back to the inputs of layer-1 neurons (again, layer-0 neurons perform no arithmetic operation). These new inputs produce a new output vector, which is sent back to the inputs as before. The process continues step by step until the output vector remains constant. When this happens, the network is said to have reached a stable state.

Recurrent networks do not always reach a stable final state. The stability of a recurrent network is guaranteed under certain circumstances by an important theorem due to Cohen and Grossberg [121], which will be explained in the next sub-section.

John Hopfield has been an active researcher in the field of recurrent networks, and for this reason the name *Hopfield networks* has been adopted to identify recurrent networks as the one

shown in Figure 4.16.

4.6.3 - Binary Hopfield network

The recurrent network of Figure 4.16 will be used to describe the binary version of the Hopfield network.

The output of each neuron is evaluated through the weighted sum of the inputs and then applying a threshold function to the sum. Symbolically:

$$n_j = \sum_i w_{ij} o_i + i_j \qquad (i \neq j) \tag{4.16}$$

$$\begin{aligned} o_j &= 1 && \text{if} \quad n_j > t_j \\ &= 0 && \text{if} \quad n_j < t_j \\ &\text{unchanged} && \text{if} \quad n_j = t_j \end{aligned}$$

where w_{ij} = weight connecting output i to input j;
o_i = output from layer-1 neuron i;
i_j = input to layer-1 neuron j;
o_j = output from layer-1 neuron j;
t_j = threshold value for layer-1 neuron j.

The set of all weights w_{ij} can be seen as a weight matrix W. In this case, W is a square matrix of size n x n, where n is the number of outputs (or layer-1 neurons). Hopfield [122] developed a training strategy for calculating the weight matrix W. This training procedure is straightforward and is given by Eq. (4.17):

$$W = \Sigma_i V_i^t \cdot V_i \qquad (i = 1, 2, ..., k) \tag{4.17}$$

where k = number of memories (i.e., training vectors);
V_i = row vector of size n (corresponding to the i^{th} memory to be encoded).

The stability theorem due to Cohen and Grossberg [121] states that the network is stable if the weight matrix is symmetrical with zeroes in its main diagonal. It should be noted that this criterion is sufficient but not necessary.

4.6.4 - Further considerations

The operation of a recurrent network can be compared to the minimisation of an artificial energy function. Hopfield networks tend to stabilise in a local minimum rather than the desired global minimum of the energy function. One way of dealing with this problem is to use statistical methods, which are very similar to that used in backpropagation training (see sub-section 4.4.6).

Hopfield [122] also investigated continuous recurrent networks. In this case, a continuous function (usually the logistic function) replaces the threshold function. The Cohen and Grossberg stability criterion also applies in this case.

Another important issue refers to the memory capacity of a recurrent network. This is still an open question and some conjectures and results have been established in recent years. To mention only two of them, Hopfield [123] showed experimentally that the maximum number of states that can be stored and retrieved without error is about $0.15n$, where n is the number of neurons. Abu-Mostafa and Saint Jacques [124] have shown that this limit cannot exceed n.

Despite the limitations of Hopfield networks, this model has one major advantage: its fast computation. This feature can be important in some applications; although the network may not produce the optimum solution, a suboptimal solution may be acceptable if obtained with low computing time.

4.6.5 - Example

A simple example was devised for illustrating the associative memory capabilities of the Hopfield network. A 6-neuron network was set up and trained with 3 vectors as shown in Table 4.9.

i_1	i_2	i_3	i_4	i_5	i_6
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0

Table 4.9 - Training vectors for example of Hopfield network

After training, the network was presented with incomplete versions of the training vectors. Table 4.10 shows the results obtained in this case. It can be seen that correct vectors were produced.

Testing vectors						Recalled vectors					
i_1	i_2	i_3	i_4	i_5	i_6	i_1	i_2	i_3	i_4	i_5	i_6
1	0	0	0	0	0	1	0	0	0	0	1
0	1	0	0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1	1	0	0

Table 4.10 - Results for example of Hopfield network

4.7 - Bidirectional Associative Memories (BAM)

4.7.1 - Introduction

Bidirectional Associative Memories (BAM) are recurrent networks like the Hopfield networks described in the previous section. Strictly speaking, Hopfield networks provide autoassociative memory capabilities because any input vector is associated with an output vector of the same type (i.e., the size of both input and output vectors is the same and the meaning of each component in both vectors is the same). This is due to the single-layered structure of Hopfield networks.

BAMs, on the other hand, are two-layered recurrent networks which allow two different types of vectors to be associated with each other (heteroassociative memory). As with Hopfield networks, there exist several versions of the BAM paradigm [96]. Only the binary version will be presented in detail here.

Closing this section, some considerations about the BAM model will be made and an example of the binary BAM will be presented.

4.7.2 - Binary BAM

Figure 4.17 shows a two-layer recurrent network which will be used to describe the binary BAM.

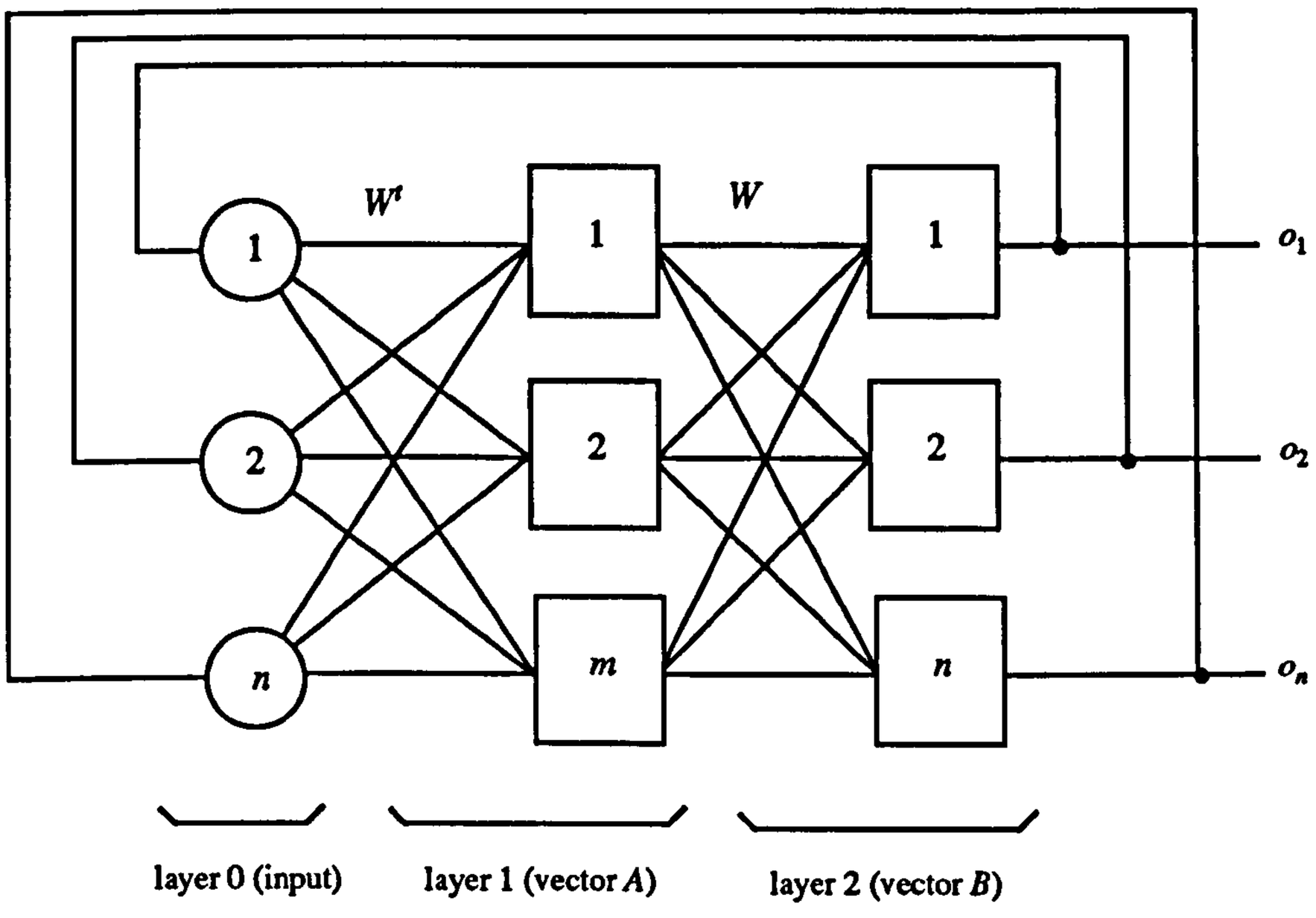


Figure 4.17 - Two-layer recurrent network

The output of each neuron is evaluated in the same way as for the neurons in Hopfield networks (Eq. (4.16)). Let W be the weight matrix associated with layer 2; its size is m x n, where m and n are the number of neurons in layers 1 and 2 respectively. Kosko [125] proved that the BAM network is unconditionally stable provided that the weight matrix associated with layer 1 is W^t. If $m = n$ and $W = W^t$ (symmetric matrices), and neurons in layers 1 and 2 are the same, then the BAM becomes an autoassociative Hopfield network. The evaluation of matrix W (and W^t) is also similar to that of Hopfield networks:

$$W = \Sigma_i A_i^t \cdot B_i \qquad (i = 1, 2, ..., k) \tag{4.18}$$

where k = number of memories (i.e., training vectors);
A_i = row vector of size m corresponding to the i^{th} memory (layer-1 vector);
B_i = row vector of size n corresponding to the i^{th} memory (layer-2 vector).

In normal operation (for retrieving a stored memory), an input vector A is applied to the outputs of layer-1 neurons. Note that this vector may be incomplete or incorrect with respect to the set

of vectors used during training. Then vector *A* is removed and the network is allowed to stabilise. When this happens, the network will produce a new vector *A'* (equal or not to the original vector *A*) and a vector *B'* (see Figure 4.17).

4.7.3 - Further considerations

The memory capacity of BAM networks is still an open issue, as it is for Hopfield networks [96]. Another serious problem is the spurious stable responses of the network, which may be only slightly related to the input vector. This is related to the local minimum problem of the associated energy function, and it is not yet a well-understood problem.

Despite these problems, advantages of BAMs are their simplicity and fast training. The theory behind them is developing quickly and it is possible that solutions for these problems be devised in the near future.

4.7.4 - Example

A network containing 16 neurons in both input and output layers was set up for this example. Table 4.11 shows the training set presented to the network.

Vector number		Vector components															
		i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}
1	*A*	-1	-1	-1	-1												
	B	-1	-1	-1	-1												
2	*A*					-1	-1	-1	-1								
	B					-1	-1	-1	-1								
3	*A*									-1	-1	-1	-1				
	B									-1	-1	-1	-1				
4	*A*													-1	-1	-1	-1
	B													-1	-1	-1	-1

Table 4.11 - Training vectors for example of BAM network (empty cells indicate value "1")

After evaluating the weight matrix, some corrupted versions of the training vectors were presented to the network. Table 4.12 shows the results obtained in this case, where it can be seen that testing vectors 2 and 3 correctly recalled the appropriate training vectors. The same did not happen with testing vectors 1 and 4, where spurious responses were produced. In all 4 cases, convergence was obtained within 2 iterations.

Input/ Output		Vector components															
		i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}
1	Input		-1														
	A																
	B																
2	Input		-1	-1				-1				-1				-1	
	A	-1	-1	-1	-1												
	B	-1	-1	-1	-1												
3	Input			-1				-1			-1	-1	-1			-1	
	A									-1	-1	-1	-1				
	B									-1	-1	-1	-1				
4	Input	-1	-1	-1	-1			-1	-1			-1	-1			-1	-1
	A	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	B	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Table 4.12 - Results for example of BAM network (empty cells indicate value "1")

4.8 - Adaptive Resonance Theory (ART)

4.8.1 - Introduction

The Adaptive Resonance Theory (ART) has been developed by Carpenter and Grossberg [126-130] with the main purpose of solving the *stability-plasticity dilemma*. This problem is the inability of some artificial neural networks to learn new inputs (plasticity) without disrupting the knowledge previously acquired (stability). For instance, when a backpropagation-trained MLP network has to learn a new training vector, the weight adjustments generated by this new vector may well destroy the knowledge the network had learned from the original training set, leading to

an unavoidable, complete re-training of the network. This can be a serious problem if the input environment is continually changing, as is often the case.

The ART architecture properly solves the stability-plasticity dilemma. It is a vector classifier which extracts salient features from the input vectors and classifies them into clusters. Its training is of the unsupervised type and every time the network reads a new input vector, a decision is made as to whether to create a new category for this vector or accommodate it within an existing category. The level of similarity upon which this decision is made is called *vigilance level* and is adjustable at any time, allowing coarser or finer classifications.

The main ART models are ART-1 and ART-2. ART-1 classifies binary input vectors, while ART-2 is a more general model capable of classifying both binary and real-valued vectors. Although the ART architecture was not used in the present work, a detailed description of the ART-1 version is included here because it is believed that useful applications in Power Systems could be developed based on the model. As an example, an ART-based classifier could be implemented for pre-processing training vectors for later use in Backpropagation training. If a large number of training vectors could be grouped by an ART network according to their statistical properties, then the MLP network could be trained with average examples of these groups. This would allow considerable reductions in the size of training sets and consequently in the computing time required by the Backpropagation algorithm.

4.8.2 - ART-1

Figure 4.18 shows a simplified ART-1 network, where the main functional blocks have been emphasised. Before the overall operation is described, each functional block will be presented in detail.

The structure of the comparison layer is presented in Figure 4.19. In this figure, each square box numbered 1, 2, ... , m represents a comparison-layer neuron, which is the arithmetical unit of this layer; m denotes the size of the input vector and n is the number of recognition-layer neurons (or memories).

Each comparison-layer neuron receives 3 inputs: (*i*) a component from the input vector $I = [i_1\ i_2\ \ldots\ i_m]$, (*ii*) a component from Gain 1 block (G_1, the same value for all neurons), and (*iii*) a component from the recognition layer. The output of the comparison layer is represented by the vector $C = [c_1\ c_2\ \ldots\ c_m]$.

Each comparison-layer neuron operates according to the *two thirds rule*, by which the output

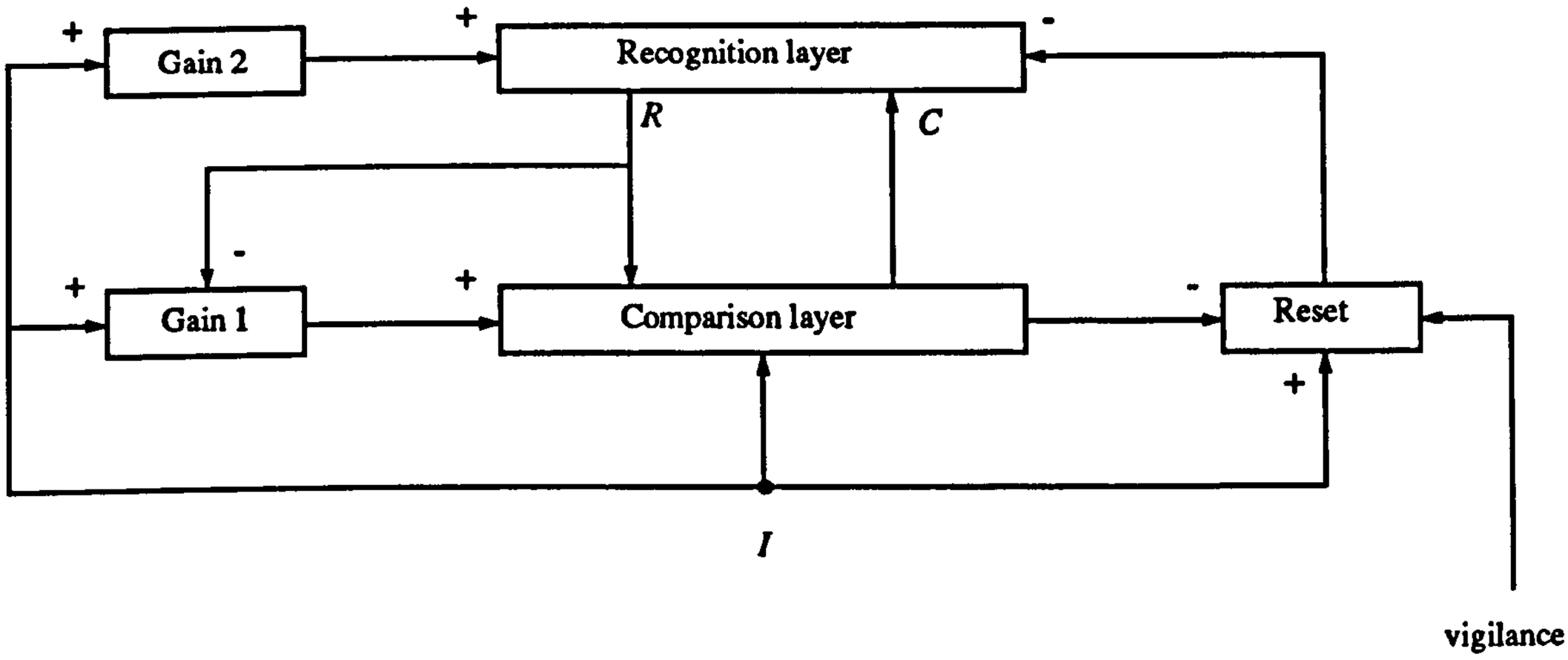


Figure 4.18 - ART-1 network

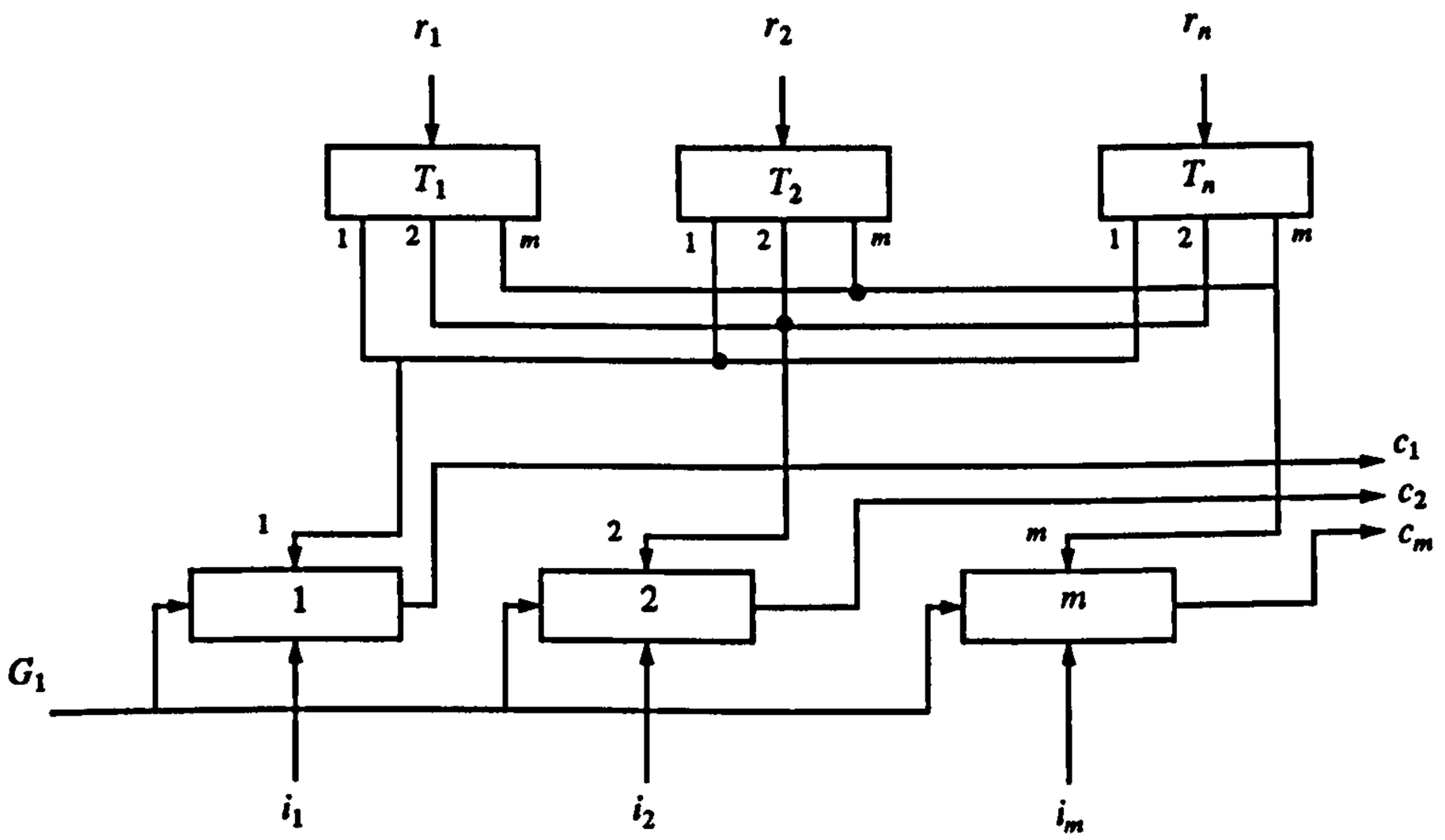


Figure 4.19 - Comparison layer structure

of the neuron is 1 only if at least two of its inputs are 1; otherwise its output is zero. Initially, the inputs from the recognition layer are all zero and Gain 1 is set to 1. Therefore the output vector *C* from the comparison layer is identical to the input vector *I*.

The structure of the recognition layer is presented in Figure 4.20. Each neuron in the recog-

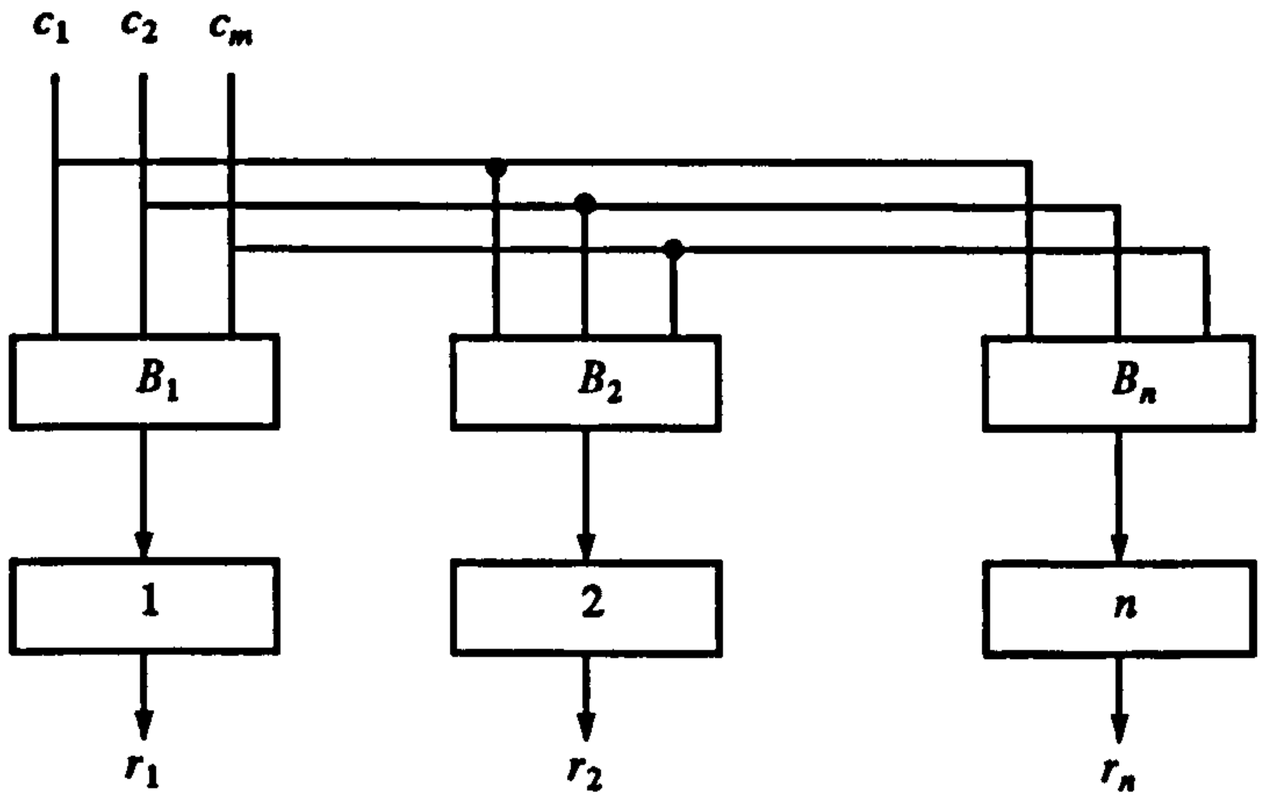


Figure 4.20 - Recognition layer structure

nition layer is represented by a square box numbered 1, 2, ... , n and has associated with it an m-dimensional real-valued weight vector B_1, B_2, ... , B_n. Each neuron evaluates the weighted sum of vector-C components (c_1, c_2, ... , c_m) using its corresponding B_i vector. Only the neuron with the largest sum fires; all others are inhibited. Therefore, the recognition layer serves to classify the vector C, firing the neuron which has the most similar weight vector B_i to vector C.

The outputs r_1, r_2, ... , r_n from the recognition layer constitute the vector R and are sent back to the comparison layer through the weight vectors T_1, T_2, ..., T_n, which are m-dimensional binary-valued vectors (see Figure 4.19).

The output from Gain 2 is simply the "or" of all the components of the input vector I. It is 1 if at least one of these components is 1; otherwise it is 0.

The output from Gain 1 is also the "or" of all the components of the input vector I, but only if all the components of vector R are 0. If vector R contains at least one component 1, then Gain 1 is set to 0. Table 4.13 presents the truth table for Gain 1 and Gain 2.

The reset module evaluates the similarity between vectors I and C. The similarity is defined as the ratio between the number of ones in vector C and the number of ones in vector I. If the similarity is below the vigilance level, a reset signal is sent to the recognition layer in order to disable the currently fired neuron.

"Or" of I	Gain 1		Gain 2
	"Or" of $R = 0$	"Or" of $R = 1$	
0	0	0	0
1	1	0	1

Table 4.13 - Truth table for Gain 1 and Gain 2

The ART classification process consists of three main tasks: recognition, comparison, and search, which will described in the following paragraphs.

Recognition phase

The initial state of the input vector I is set equal to the null vector (all components equal to zero). This sets Gain 2 to zero and hence all recognition-layer neurons are disabled (see Figure 4.18). Therefore, the output vector R from the recognition layer is a null vector.

An input vector I is then applied. If it is different from the null vector, Gain 1 and Gain 2 will be set equal to 1. The non-zero components of the input vector I will fire their corresponding neurons in the comparison layer, because these neurons will have the two non-zero inputs required to fire: one from Gain 1 and one from the input vector. So at this stage, the vector C will start as a copy of the input vector I.

Next, the recognition layer will compute the dot product between vector C and each of the recognition-layer weight vectors ($B_1, B_2, \ldots, B_n$). The winning neuron (the one with the largest dot product, or the most similar to vector C) will fire. Assuming that this neuron is the i^{th} one, the weight vector T_i associated with it will feed back the third input of the comparison-layer neurons. This vector T_i represents the i^{th} category of stored patterns due to the previous inputs seen by the network. Therefore, the recognition layer sends the most similar stored pattern back to the comparison layer.

Comparison phase

Once the recognition layer has found the best match for the input vector, the comparison layer will compare this stored pattern to the input vector. It should be noted that, once the output vector R from the recognition layer is no longer a null vector, Gain 1 is reset to 0. At this stage, the inputs to the comparison layer are the vector T_i and the input vector I. This will cause the vector C

to be updated; it will become the logical "and" between vectors T_i and I. This new vector C will then be compared to the input vector, with similarity being equal to:

$$s = \frac{n_C}{n_I} \leq 1$$

where n_C, n_I are the number of ones in vectors C and I respectively. If the similarity is greater than the vigilance level ρ, the network will assume that the input vector I is an exemplar of category i, and it will proceed to train (update) vectors T_i and B_i so as to accommodate the input vector as a new member of category i. After this, the classification process is terminated.

On the other hand, if the similarity is below the vigilance level, the network will search for another recognition-layer neuron which matches the input vector with a similarity above the vigilance level.

Search phase

The search phase occurs whenever a stored pattern returned by the recognition layer does not match the input vector with satisfactory similarity. This is possible because previous training processes have modified weight vectors B_i and T_i in such a way that the best match found by the recognition layer may not recall an acceptable stored pattern. In terms of the network, once a similarity test fails, the reset block sends a signal to the recognition layer so as to disable the currently activated neuron i; vector R becomes equal to the null vector, Gain 1 is set to 1 and vector C is made equal to the input vector again. A search process starts by which all the neurons except neuron i (now disabled) will be tested until a neuron is found such that its similarity is above the vigilance level. If that neuron is found, and assuming that it is the j^{th} one, the search process terminates and vectors T_j and B_j will be trained so as to accommodate the new exemplar. If no neuron is found, a new recognition-layer neuron k (fresh memory or new category k) is commissioned and vectors T_k and B_k are initialised so as to represent the input vector.

Training

Whenever a suitable category i is found so as to accommodate a new input vector, i being either an already existing category or a newly created one, weight vectors T_i and B_i are updated in order to include the information from the input vector. The equations for updating these vectors are as follows:

$$b_{ij} = \frac{L \cdot c_j}{L - 1 + \Sigma_k c_k} \qquad j = 1, 2, ..., m$$

$$t_{ij} = c_{ij}$$

where i = the category being updated or created;
j = the j^{th} component of vectors B_i, T_i, and C;
m = size of input vectors;
L = a constant value greater than 1 (typically 2).

Weight initialisation

Weight vectors B_i and T_i must be given initial values for the ART-1 to operate properly. According to Carpenter and Grossberg [126], initial values for these vectors should be:

$$b_{ij} < \frac{L}{L-1+m} \qquad j = 1, 2, ..., m$$

$$t_{ij} = 1$$

where parameter L has the same meaning as before.

4.8.3 - Example

The operation of the ART-1 model will be illustrated through the same example as shown in reference [96]. A binary square matrix of dimension 8 is used to encode characters from the alphabet. In terms of the network, these characters are treated as 64-element binary vectors. Some characters are then presented to the network in order to be classified. Figure 4.21 shows both the input data and the classification results produced by the network. In this case, the vigilance level was set to the value 0.95.

From Figure 4.21 it can be seen that the first three inputs, being different one to another (in the sense that their similarity is below the vigilance level), caused the network to commit three new categories (fresh memories). The fourth input, a corrupted "*E*", is classified as being member of category 3 due to a similarity of 0.962 (above the vigilance level). Note that the fourth input contains an extra asterisk as well as a missing asterisk in its lower stroke. After training, the pattern corresponding to the third category is a combination of both uncorrupted and corrupted "*E*"s (only the missing asterisk was transferred to the new category 3). The fifth input is a more corrupted "*E*" (again with both a missing and an extra asterisk) which could not be accommodated in the category 3 due to a similarity of 0.923 (below the vigilance level), therefore causing the net-

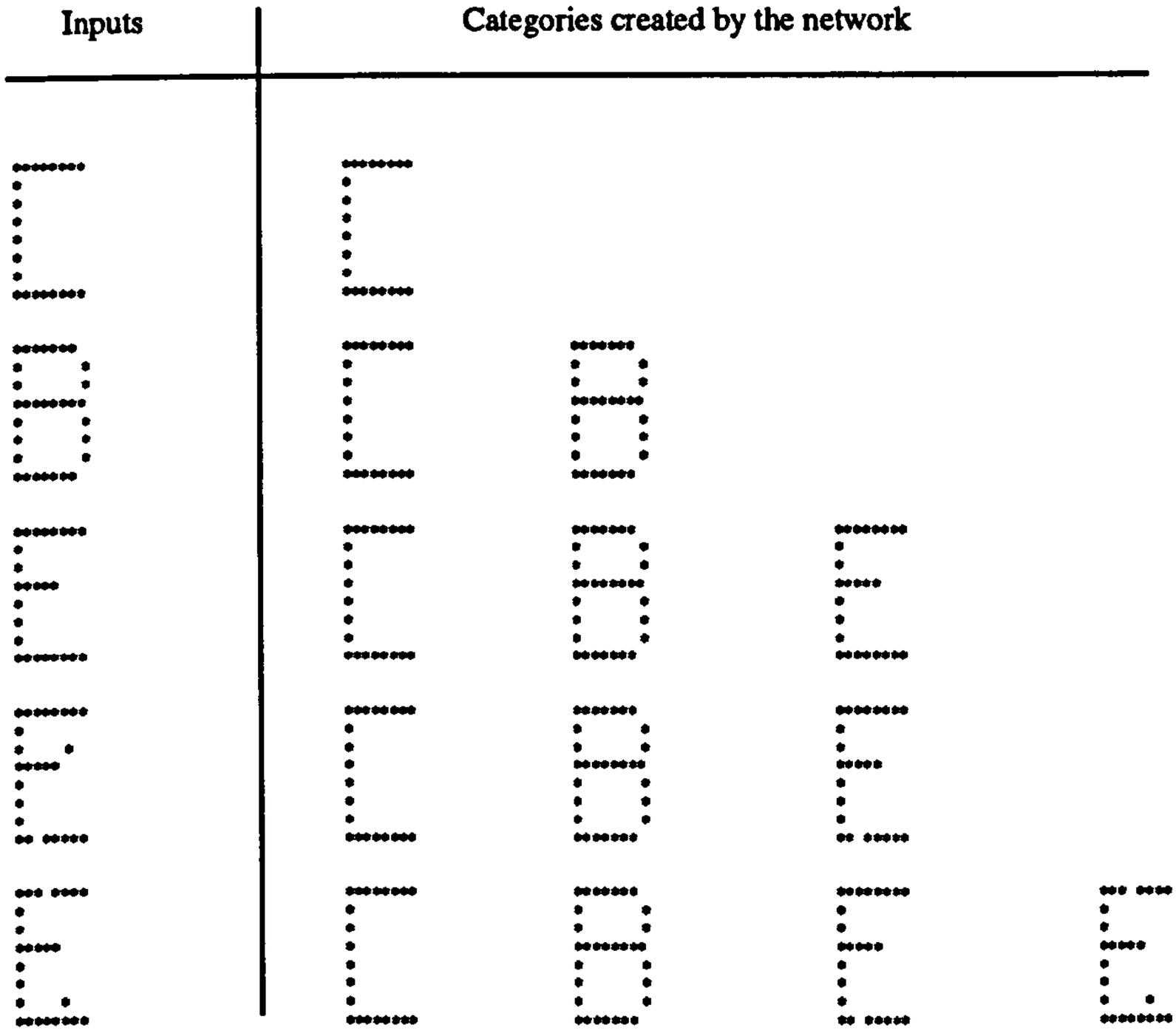


Figure 4.21 - ART-1 classification session

work to create a new category.

4.9 - Summary

This chapter presented some introductory concepts on biological neural networks and the relationship with their artificial counterparts. The main paradigms of artificial neural networks currently available were described and implemented, and their operation was illustrated through examples.

The conclusions of this chapter can be summarised as follows:

- the desire to understand and mimic the human brain is the driving force behind the development of artificial neural networks (ANNs), as well as other branches of Artificial Intelligence;

- in the 1940s, Perceptrons were the first models of ANNs to be proposed. In the late 1960s, Minsky and Papert showed that there are several restrictions on what single-layer perceptrons can represent and learn;

- Multi-Layer Perceptrons can overcome the limitations inherent to single-layer Perceptrons, but for many years there was no automatic procedure for training MLPs. The Backpropagation algorithm was the first systematic method for training MLPs, and its most recognised description was published in 1986. After this, the whole field of ANNs experienced renewed interest, with innumerable applications being proposed in many fields of science. Today, MLPs are the most popular paradigms of ANNs;

- more recently, other ANN models have been proposed. The self-organising map and the ART are examples of networks that are trained in the unsupervised mode, which seek to take into account some other fundamental aspects that can be observed in human behaviour;

- despite the very rapid development of the ANN field, there is still little equivalence between biological and artificial neural networks, and this is due to the fact that only very little is known about the functioning of the human brain;

- in this work, a simple modification to the original backpropagation algorithm was devised and tested. This modification improved the convergence properties of the training algorithm. Further work is needed in order to quantify the gains brought about by this modification (and possibly to give some indication as to an "optimum" number of internal iterations).

In the following two chapters, the voltage stability problem will be analysed from two different viewpoints. In both approaches, the MLP network, trained with the Backpropagation algorithm, will be used so as to produce useful information with respect to the voltage stability of power systems.

CHAPTER 5

STATIC ANALYSIS OF THE VOLTAGE STABILITY PROBLEM

5.1 - Introduction

This chapter presents an initial study of the voltage stability problem in power systems using an artificial neural network approach. In this case, knowledge about the electrical problem was extracted solely from the steady-state load-flow equations, hence the static nature of the proposed solution. In Chapter 6, the approach will be developed so as to take into account the dynamic aspects of the problem.

The material presented here is a much more detailed description of the work which the author reported in reference [131]. The outline of this novel method, its computational implementation, and the results obtained in three different cases will be presented in the following sections. At the end of the chapter, a general discussion of the proposed methodology will also be presented.

5.2 - Outline of the method

The aim of the present approach is to train an ANN to make it capable of computing a static voltage stability index as well as other electrical parameters of interest. It should be pointed out that the necessary theoretical background can be found in chapters 3 and 4 (current approaches to the voltage stability problem and artificial neural networks respectively).

The static voltage stability index that will be used in this chapter is the Minimum Singular Value (MSV) of matrix G_S defined by the following equations (see sub-section 3.2.2):

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = J \cdot \begin{bmatrix} \Delta \theta \\ \Delta V \end{bmatrix} = \begin{bmatrix} F_\theta & F_V \\ G_\theta & G_V \end{bmatrix} \cdot \begin{bmatrix} \Delta \theta \\ \Delta V \end{bmatrix} \qquad (5.1\ a)$$

$$\Delta Q = G_S \,.\, \Delta V \tag{5.1 b}$$

$$G_S = G_V - G_\theta \,.\, F_\theta^{-1} \,.\, F_V \tag{5.1 c}$$

Eq. (5.1 a) represents the linearisation of the power flow equations (cf. sub-section 5.3.2) around an operating point; sub-matrices F_θ, F_V, G_θ and G_V represent partial derivatives of bus-injected active and reactive power with respect to bus angles and voltages. Eq. (5.1 b) relates changes in bus reactive power injections to changes in voltages when there are no changes in bus reactive injections; i.e., Eqs. (5.1 b) and (5.1 c) can be obtained by imposing the condition $\Delta P = 0$ in Eq. (5.1 a).

Another interpretation for matrix G_S can be established by computing the determinant of the full Jacobian matrix J using Schur's formula [132]:

$$det\, J = det\, F_\theta \,.\, det\, G_S \tag{5.2}$$

Eq. (5.2) is valid only if the assumption that sub-matrix F_θ is non-singular is made ($det\, F_\theta \neq 0$). Matrix G_S is also called *Schur's complement* and is defined as in Eq. (5.1 c). The non-singularity of matrix F_θ can be seen as an absence of static angle stability problems. If this is the case, it follows from Eq. (5.2) that the full Jacobian matrix J will become singular only when matrix G_S becomes singular. Therefore, matrix G_S can be seen as a matrix associated with J and containing information relevant to the static voltage stability of a power system.

The choice of the MSV of matrix G_S as index for voltage stability was made because this technique can produce useful information about an operating point of a power system. As seen in Chapter 3, the Singular Value Decomposition not only gives some indication of the proximity to the steady-state stability limit, but it can also indicate, through the right and left singular vectors, critical areas of voltage stability and the most effective buses for the application of corrective actions.

Having assumed the MSV of G_S as the static voltage stability index, the next choice regards the neural network model to be used. Since the electrical problem includes the computation of a global, scalar index from the current operating state of the system, it seems appropriate to consider the MLP network, which can produce correct answers to interpolation problems.

It is important to remember here that the computation of a full Singular Value Decomposition increases as the cube of the size of the matrix under consideration. This constitutes another major reason for choosing the MLP architecture at this stage, given its extremely fast computation.

An essential step when setting up an MLP network is the definition of the training set. This basically means the specification of input and output variables and the number of vectors to be used during training and testing of the network. To achieve good training results, the MLP should be trained with a *training* set that properly represents the problem at hand. Once the training is completed, the MLP must be tested using a *testing* set different from the training set; only the independence of the training and testing sets can guarantee that it is possible for the MLP to learn useful knowledge about the physical problem.

Table 5.1 presents the types of input and output variables that will be considered throughout this chapter.

Input	Bus active load
	Bus reactive load
	Branch state (open or closed)
Output	Minimum Singular Value of J
	Minimum Singular Value of G_S
	Bus voltage
	Bus angle
	Bus active generation
	Bus reactive generation

Table 5.1 - Types of input and output variables

Once the input and output variables and the number of training and testing vectors have been defined, the next step is to create both training and testing sets. The general procedure to achieve this is as follows:

1. select an input vector;
2. execute a load-flow run, using the actual values from the input vector for the input variables;
3. extract the value of the output variables from the load-flow solution and complete the current training/testing vector with these values;
4. go to step (1), if necessary.

As will be seen later, the input vector can be read from an external source (given by the user) or generated automatically using the Monte Carlo method. In the latter case, values for the input variables are randomly generated using adequate minimum and maximum values and a pre-specified distribution model (e.g., Gaussian, uniform, etc.). The Monte Carlo method is particularly suited to problems where the space of possible states is too large to be generated manually (by explicit enumeration, for instance); it efficiently covers the state space with any desired density, which is in turn controlled by the number of training/testing vectors being created.

5.3 - Computational system

5.3.1 - Introduction

In this section, the computational system that was developed for assessing the static voltage stability of power systems will be described. This system was implemented using the *C* language and is currently operating on Unix-based workstations in the Department of Electronic Engineering, QMW.

Figure 5.1 shows the structure of the computational system. It can be seen that it consists of four separate modules (programs) and three different structures of permanent files. These files allow information to be exchanged among the programs.

Program **FLOW4** is the module responsible for load-flow calculations. It reads data from an external user file and, on completion of the calculations, it stores a complete case in a binary file that possesses a file extension "**.flo**". Binary files of type **.flo** can be read by program FLOW4 (for instance, when creating a brand new case from an older case with some minor modifications) and also by the interface INTF1.

Program **INTF1** is the interface module between the load-flow and neural network program (FLOW4 and STATISTIC1 respectively). It acquires a solved load-flow base case from a **.flo** file and constructs a complete training/testing set for the neural network. This training/testing set can then be stored in binary files with extension "**.trf**"; these files can be read by program INTF1 for creating new cases from existing cases, and also by program STATISTIC1.

Program **STATISTIC1** is responsible for both the training and the operation of MLP networks. Training/testing sets can be given either manually by the user or through a **.trf** file previously generated by the interface program INTF1. The program also has facilities for storing/

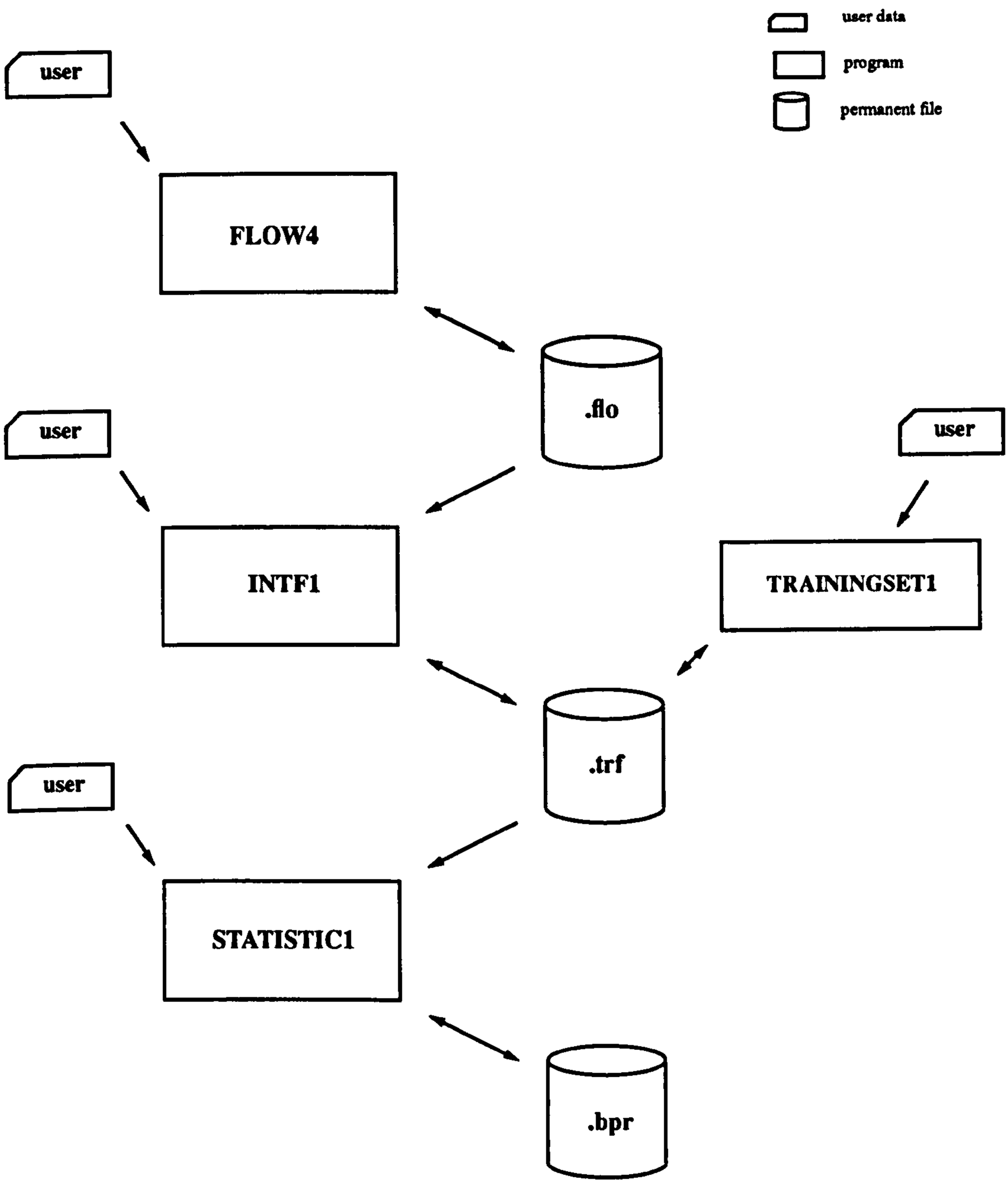


Figure 5.1 - Computational system for static voltage stability assessment
(output results from all programs not shown)

retrieving trained networks in/from binary files with extension "**.bpr**".

Finally, module **TRAININGSET1** can be used for generating training/testing sets in special user-oriented applications. This means that specific modelling features can be incorporated into the core program as *C* functions. This program is capable of storing the generated training sets in **.trf** files for later use by the program itself and by the neural network program (STATISTIC1).

Modules FLOW4, INTF1 and STATISTIC1 will be described in detail in the next sub-sections.

5.3.2 - Load-flow program (FLOW4)

In terms of the program FLOW4, buses in the power system must be of one of the types defined in Table 5.2.

Bus type	Purpose	Variables set by the user	Variables computed by the program
PQ	To represent loads	p, q	v, θ
PV	To represent generators	p, v, q_{min}, q_{max}	q, θ
PQV	To represent load buses with voltage controlled by transformers with automatic tap setting	p, q, v	θ and tap of associated transformer
Swing	Swing bus (1 bus only)	v, θ	p, q

Table 5.2 - Types of buses considered by the load-flow program

Assuming that a power system has l buses of type PQ, m buses of type PV and n buses of type PQV, the structure of the system of equations to be solved is as follows:

equations in P: $l + m + n$
equations in Q: $l + n$

where the equations in P and in Q respectively have the form:

$$p_i = v_i^2 \cdot g_{ii} + v_i \cdot \sum_{i \neq j} v_j (g_{ij}\cos\theta_{ji} - b_{ij}\sin\theta_{ji}) \qquad (5.3\ a)$$

$$q_i = -v_i^2 \cdot b_{ii} - v_i \cdot \sum_{i \neq j} v_j (g_{ij}\sin\theta_{ji} - b_{ij}\cos\theta_{ji}) \qquad (5.3\ b)$$

and where p_i, q_i = active and reactive power injected at bus i;
v_i, θ_i = voltage (pu) and angle (rad) at bus i;

θ_{ji} = θ_j - θ_i;
$g_{ij} + jb_{ij}$ = element (i,j) of the nodal admittance matrix of the power system (pu).

The load-flow problem can then be stated as finding values for v_i (at buses of type *PQ*) and for θ_i (at buses of types *PQ*, *PV* and *PQV*) such that the power injections given by Eq. (5.3) equal the specified values of p_i (for buses of types *PQ*, *PV* and *PQV*) and q_i (for buses of types *PQ* and *PQV*). In program FLOW4, the resulting system of equations is solved using the Newton-Raphson method with the full Jacobian matrix (matrix *J* in Eq. (5.1 a)) computed at each iteration. Once the iterative process has converged within a pre-specified tolerance, the reactive power injected at *PV* buses is computed, as well as the active and reactive power injected by the swing bus.

The last step in the solution process is the computation of the minimum singular values of matrices G_S and/or *J* (see Eq. (5.1)). This is accomplished using the algorithm described by Lawson and Hanson [11], whereby the matrix under consideration is first transformed into upper bidiagonal form through Householder transformations, and then the so-called *QR* algorithm is applied to the transformed matrix.

In program FLOW4, loads are represented as voltage-dependent injections, according to:

$$p_i(v_i) = p_{0i} \cdot \left[\frac{v_i}{v_{0i}}\right]^{e_i} \qquad (5.4\ a)$$

$$q_i(v_i) = q_{0i} \cdot \left[\frac{v_i}{v_{0i}}\right]^{e_i} \qquad (5.4\ b)$$

where p_i, q_i = actual active and reactive power (pu) absorbed at bus *i* when the bus voltage is v_i (pu);
p_{0i}, q_{0i} = active and reactive power (pu) absorbed at bus *i* when the bus voltage is v_{0i} (pu) (p_{0i}, q_{0i} and v_{0i} are reference values given by the user);
e_i = load exponent for bus *i*.

Eq. (5.4) is a convenient way to represent voltage-dependent loads. Particular cases, such as constant power, constant current, or constant impedance loads are easily modelled by setting the exponent e_i to the values 0, 1 or 2 respectively.

To allow for the automatic adjustment of transformer taps, the first step is to make the value of the tap explicit in Eq. (5.3), through the quantities g_{ii}, b_{ii}, g_{ij} and b_{ij}. This procedure allows the power injections at the transformer terminal buses to be expressed as functions of the transformer tap. The derivatives of these injections with respect to the tap are then easily included in the Jacobian matrix of the system. New values for the tap of automatically-adjusted transformers are computed at each iteration, whereas the voltage of the *PQV* buses controlled by these transformers is now kept constant.

Program FLOW4 also allows the reactive power generated at *PV* buses to be kept within pre-specified limits. At the end of each iteration, reactive power injections at buses *PV* are checked against their corresponding limits; in case there is a limit violation, the bus is converted into a *PQ* type. The reactive power injection of the bus is set to the violated limit and the corresponding *Q* equation is added to the system of equations. The bus voltage is released from its fixed value and new values will be computed in the next iterations. Once a *PV* bus is converted into type *PQ*, it cannot be changed back to type *PV* in the remaining iterations. This procedure helps to ensure the convergence of the iterative process. Also, checking against reactive power limit violations starts only after 2 iterations have been performed, in order to avoid improper corrective actions due to the solution path followed by the iterative process at early stages.

Finally, it should be pointed out that the coefficient matrix of the linearised system of equations (load-flow Jacobian matrix) is sparse due to the nature of practical power systems. Program FLOW4 fully exploits this characteristic through the use of linked lists for storing the Jacobian matrix. Greater computing speed and numerical precision are achieved thanks to this efficient computational technique.

5.3.3 - Interface program (INTF1)

Figure 5.2 shows a block diagram describing the main tasks in the generation of training vectors with program INTF1.

Referring to Figure 5.2, the definition of the state of the network means making an input vector available for use by the program, and this can be done manually or automatically (input and output variables for program INTF1 are those already defined in Table 5.1). In manual definition, the user simply specifies the value of all input variables in each input vector. In automatic definition, the Monte Carlo method is used to generate values for the input variables; these values are generated according to a chosen distribution model and are controlled by pre-specified minimum and maximum values. The program allows the user to choose between uniform and normal (Gaussian) distribution models. Automatic generation of input vectors is very convenient for generating large training/testing sets (with more than 100 vectors).

Once an input vector has been defined (either manually or automatically), the load-flow and SVD subroutines compute the value of the output variables. A complete training vector, containing input and output variables, is then stored and the program starts the generation of the next vector.

After the whole training/testing set has been generated the program executes a scaling process, which is a linear transformation of input and output variables in all the training vectors. Scal-

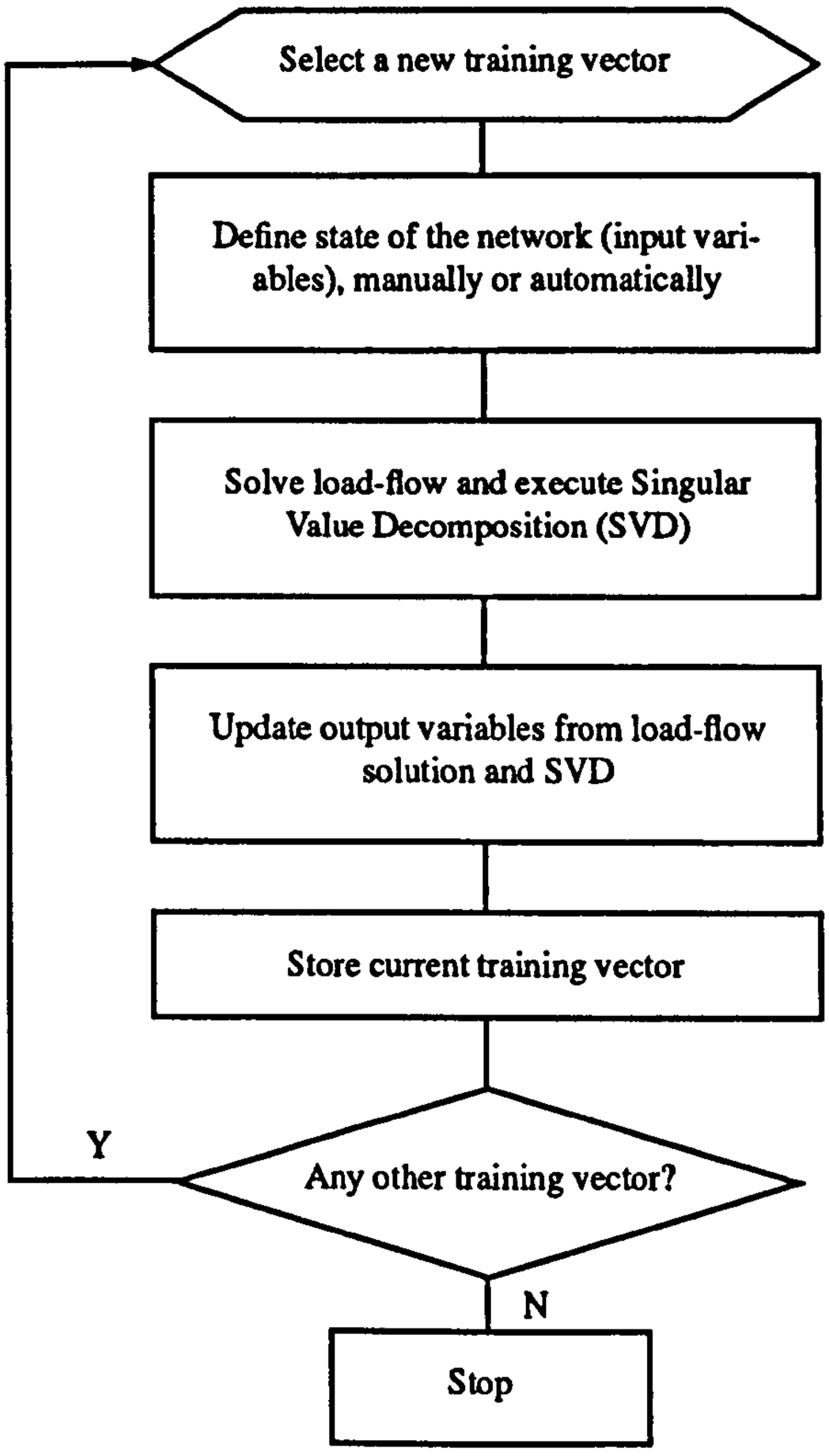


Figure 5.2 - Training vector generation in program INTF1

ing the input variables is done to map the input values onto a convenient range so as to avoid saturation of the first-layer neurons of the MLP. Scaling the output variables serves when output values outside the MLP range of [0, 1] are required in the application being constructed, and also to avoid the "dead zone" of the sigmoid function near the values 0 and 1 (see sub-section 4.4.7 for a more detailed explanation of the scaling process).

5.3.4 - Multi-Layer Perceptron program (STATISTIC1)

Program STATISTIC1 implements the Multi-Layer Perceptron paradigm. It allows the use of three different training algorithms: pure backpropagation, pure statistical, or combined back-

propagation/statistical (see sub-sections 4.4.5 and 4.4.6). When using backpropagation training, either the momentum method or the exponential smoothing method can be used. The modified algorithm described in section 4.4.7, with an internal iteration loop, can also be specified by the user. An arbitrary number of layers and neurons in hidden layers can be defined (the only limitation in this respect refers to the available memory for storing the weights in the computer).

5.4 - Results

5.4.1 - Introduction

The computational system described in the previous section was used in a series of study cases in order to illustrate the application of the proposed methodology. This series is labelled "Series A" and consists of three different cases. Cases A1 and A2 provide a qualitative description, while Case A3 presents a quantitative analysis. All three cases will be described in the following sub-sections.

5.4.2 - Case A1

The electrical network for this case is the 6-bus Ward and Hale system [133]. Table 5.3 shows the definition of input and output variables, as well as their corresponding rated, minimum, and maximum values.

In this case, the training set was set up manually by taking 26 different values for the only input variable Q_6, the reactive power absorbed by bus 6, and computing the corresponding output values. It should be noted in Table 5.3 that for values of Q_6 above the maximum of 79.6 MVAr, the load-flow does not converge, indicating proximity to the steady-state stability limit (the corresponding value for σ_G, the minimum singular value of matrix G_S, is 0.00405, which is very close to zero).

Tables 5.4 and 5.5 summarise the main data regarding the MLP structure and its training.

Figure 5.3 shows the performance of the MLP when the reactive load Q_6 is varied between its minimum and maximum values. The 3 curves generated by the MLP (σ_G, σ_J and V_3) are very close to the corresponding curves generated by the load-flow program (in fact, the corresponding curves are nearly the same), and this indicates good adherence of the MLP to the electrical problem. It can also be seen that the minimum singular values σ_J and σ_G are more sensitive to load changes than the bus voltages.

Variable		Rated value	Min. value	Max. value
Input	Q_6 (load) (MVAr)	5.00	0.	79.60
Output	σ_J	0.59565	0.00197	0.60585
	σ_G	1.40588	0.00405	1.43550
	V_3 (pu)	1.000	0.832	1.006
	V_6 (pu)	0.919	0.529	0.932
	P_1 (gen.) (MW)	95.23	94.90	126.09
	Q_1 (gen.) (MVAr)	43.57	39.45	168.50
	Q_2 (gen.) (MVAr)	18.56	16.47	84.18

Table 5.3 - Input and output variables for Case A1

Training set		Nº of neurons per layer		
Type of generation	Nº of training vectors	Layer 0 (input)	Layer 1 (hidden)	Layer 2 (output)
manual	26	1	3	7

Table 5.4 - MLP structure for Case A1

Parameter	Training session					
	1	2	3	4	5	6
Nº of iterations performed	1000	1000	1000	1000	1000	1000
Learning rate	1.	0.9	0.8	0.7	0.6	0.5
Expon. smoothing coeff.	0.3					

Table 5.5 - Training sessions for Case A1

Finally, it should be noted that the CPU time for the whole training in this case was 87.0 s on a Sun Sparc 2 workstation.

5.4.3 - Case A2

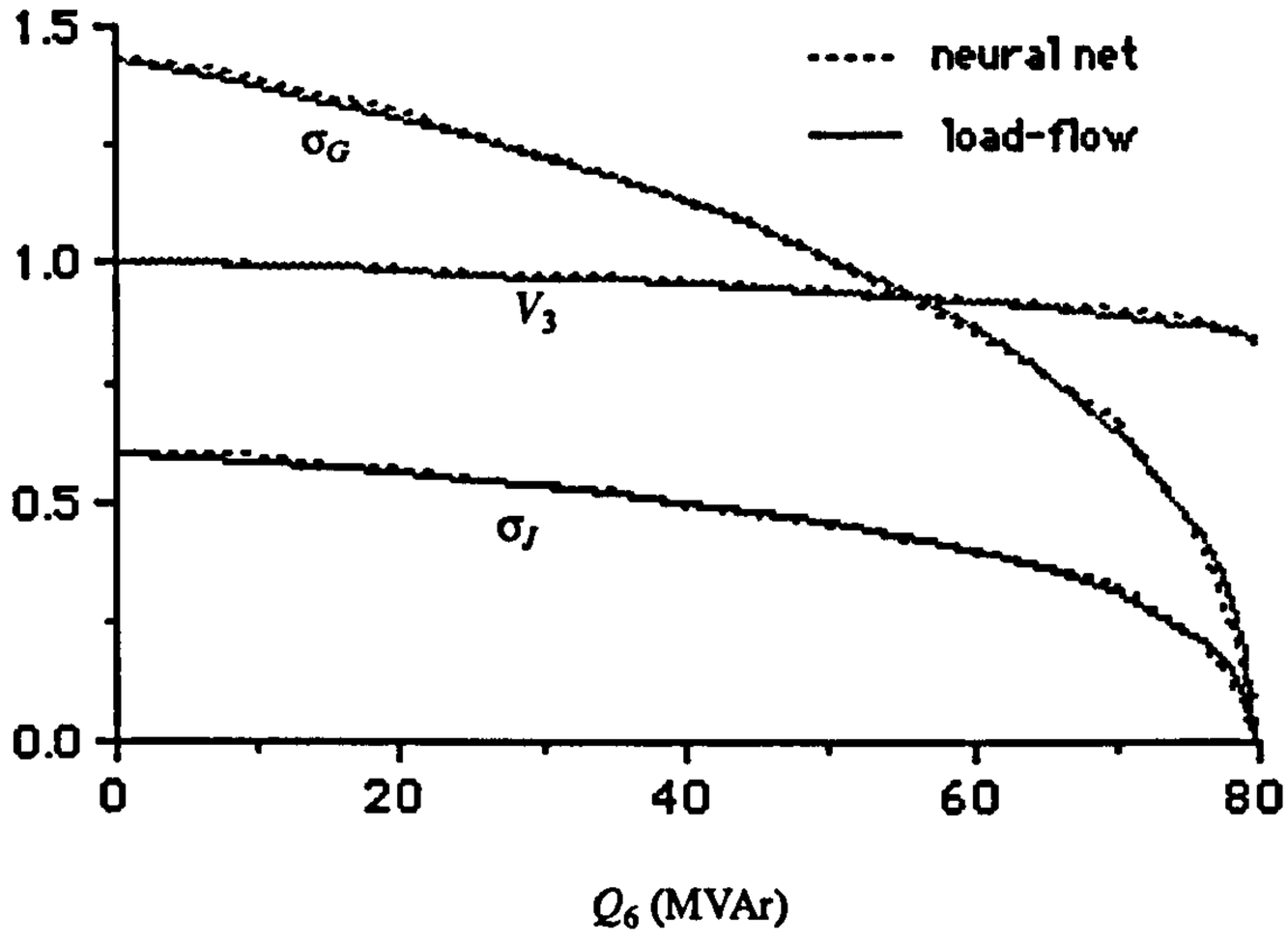


Figure 5.3 - MLP performance for Case A1 (6-bus Ward and Hale system)

In this case, the 30-bus AEP system [134] was used. The value of the active and reactive power at buses 17, 18, 19, and 20 was varied simultaneously while keeping their power factor constant at the value 0.89 [14]. Table 5.6 shows the definition of input and output variables, as well as their corresponding rated, minimum, and maximum values.

The training set was also set manually and it contained 20 different values for the input variables, distributed between the minimum and maximum values indicated in Table 5.6.

Tables 5.7 and 5.8 show the main data regarding the MLP structure and its training.

From Table 5.8 it can be seen that the third training session contained 465 iterations only. This is because the convergence criterion of the training algorithm was satisfied at the end of iteration 465 (see section 4.4 for a complete discussion of the training algorithm).

Figure 5.4 shows the performance of the MLP when the active load at buses 17, 18, 19 and 20 is varied between the values 0 and 22.64 MW. It can be seen that the curve generated by the MLP is very close to the reference curve produced by the load-flow program. Also, the error of the MLP prediction is higher in the end region of the curve (large values of active load). This is due to the fact that the MLP was tested with load values of up to 22.64 MW, while it was trained with val-

Variable		Rated value	Min. value	Max. value
Input	P_{17} (load) (MW)	9.00	1.70	21.84
	P_{18} (load) (MW)	3.20		
	P_{19} (load) (MW)	9.50		
	P_{20} (load) (MW)	2.20		
Output	σ_J	0.22266	0.04344	0.22783
	σ_G	0.49250	0.06437	0.49931
	V_{10} (pu)	1.045	0.841	1.055
	V_{19} (pu)	1.025	0.719	1.045
	P_1 (gen.) (MW)	261.04	241.65	348.32
	Q_1 (gen.) (MVAr)	-16.36	-15.99	112.60

Table 5.6 - Input and output variables for Case A2

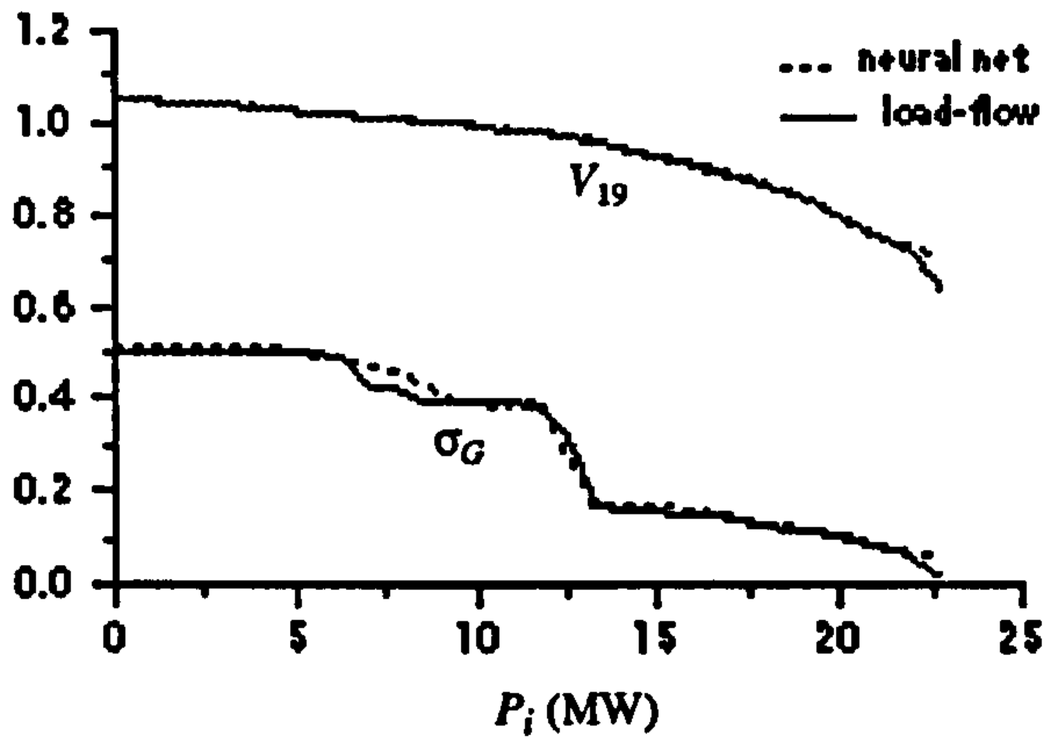
Training set		Nº of neurons per layer		
Type of generation	Nº of training vectors	Layer 0 (input)	Layer 1 (hidden)	Layer 2 (output)
manual	20	4	10	6

Table 5.7 - MLP structure for Case A2

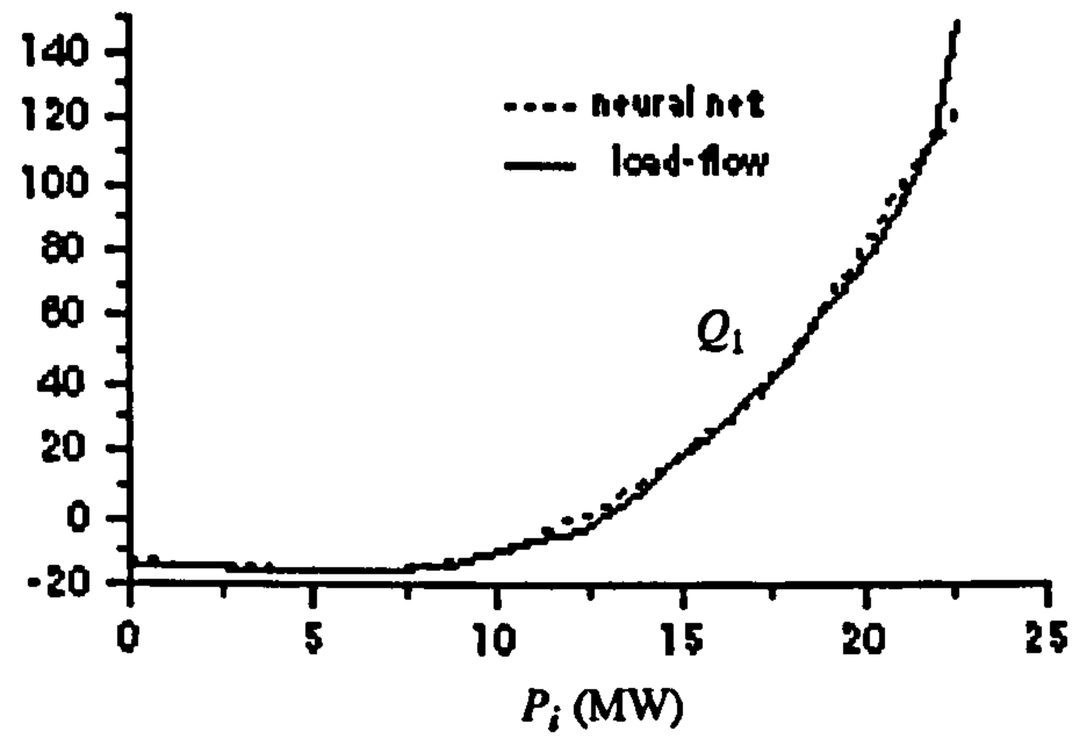
Parameter	Training session		
	1	2	3
Nº of iterations performed	3000	3000	465
Learning rate	1.	1.	1.
Expon. smoothing coeff.	0.3		

Table 5.8 - Training sessions for Case A2

ues of up to 21.84 MW only. This underlines the care that must be taken so that the MLP is not used with input data outside its training range (i.e., in extrapolative computations), even for small deviations.



(a) - σ_G and V_{19} (pu)



(b) - Q_1 (MVAr)

Figure 5.4 - MLP performance for Case A2 (30-bus AEP system)

It is interesting to note in Figure 5.4 (a) that the curve corresponding to the load-flow program contains some points where the minimum singular value of matrix G_s starts to decrease more rapidly as the demand increases. These points of sudden change correspond to generators reaching their maximum reactive power generation. In fact, when a generator reaches its reactive power generation limit, the bus is converted to type PQ. In terms of the Jacobian matrix, this means that an extra row/column is added to the matrix. It can be shown [13] that when a column is added to a matrix, its largest singular value increases and its minimum singular value decreases.

Finally, it should be noted that the CPU time for the whole training in this case was 186.9 s on a Sun Sparc 2 workstation.

5.4.4 - Case A3

The 30-bus AEP system [134] was used again in this case. Input variables are the load at buses 5, 7 and 8 as well as the state of lines 1-3 and 2-5 (open or closed). The three buses are the most heavily loaded buses in the system and the two lines are among the most critical for outage purposes (rated flow above 80 MVA for both lines). Output variables are the minimum singular values of matrices J and G_S, the voltage of buses 10 and 19, and active and reactive generation at bus 1 (the swing bus). Table 5.9 shows details of the definition of variables in this case.

The Monte Carlo method was used to generate 974 training vectors. Values for load at buses

Variable		Rated value	Min. value	Max. value
Input	P_5 (load) (MW)	94.20	56.54	131.82
	P_7 (load) (MW)	22.80	13.69	31.91
	P_8 (load) (MW)	30.00	18.02	42.00
	Line 1-3	0 (closed)	0	1 (open)
	Line 2-5	0 (closed)	0	1 (open)
Output	σ_J	0.22266	0.01165	0.22700
	σ_G	0.49250	0.02219	0.49558
	V_7 (pu)	1.002	0.777	1.007
	P_1 (gen.) (MW)	261.04	198.52	333.65
	Q_1 (gen.) (MVAr)	-16.36	-22.47	93.46

Table 5.9 - Input and output variables for Case A3

5, 7 and 8 varied between -40% and +40% of their rated values (with the power factor of the three buses kept constant at the corresponding rated value) and the probability of lines 1-3 and 2-5 being out of service was set to 0.1, deliberately high to present the MLP with more critical scenarios. Uniform distribution was used for both load levels and line state. Tables 5.10 and 5.11 show the

main data regarding the MLP structure and its training.

In order to evaluate the performance of the MLP for this case, a new testing set was gener-

Training set		Nº of neurons per layer			
Type of generation	Nº of training vectors	Layer 0 (input)	Layer 1 (hidden 1)	Layer 2 (hidden 2)	Layer 3 (output)
Monte Carlo (uniform distribution)	974	5	15	10	5

Table 5.10 - MLP structure for Case A3

Parameter	Training session				
	1	2	3	4	5
Nº of iterations performed	1000	1000	1000	1000	1000
Learning rate	1.	0.9	0.8	0.7	0.6
Expon. smoothing coeff.	0.2				

Table 5.11 - Training sessions for Case A3

ated through the Monte Carlo method with the same parameters as the original training set (i.e., load varying between -40% and +40% of the rated value and probability of a line being out of service set to 0.1). The testing set contained 971 vectors, none of them identical to any vector in the original training set. The testing set was then presented to the MLP. Table 5.12 presents the average and standard deviation of relative error obtained with the 971 vectors from the testing set.

It should be noted that the relative error of each testing vector was computed for each output variable as indicated by Eq. (5.5).

$$error\,(\%) = \frac{\left|v_M - v_r\right|}{v_{max} - v_{min}} \cdot 100 \tag{5.5}$$

where v_M is the value computed by the MLP, v_r is the reference value (correct output) and v_{min}, v_{max}

Parameter	Output variable				
	σ_J	σ_G	V_7	P_1	Q_1
Average error (%)	1.13	1.49	0.85	0.66	0.84
Standard deviation of error (%)	1.60	2.30	0.97	0.68	1.30

Table 5.12 - Average and standard deviation of relative error for Case A3 (using the testing set)

are the minimum and maximum values respectively for the output variable. Table 5.13 presents the corresponding distribution of relative error. It can be seen that for the first output variable (σ_J), 97.3% of the testing vectors were computed with a maximum relative error of 4%. The corresponding percentages for the remaining output variables are 93.1, 99.6, 99.5 and 98.8% respectively.

The CPU times in this case are presented in Table 5.14 (training) and Table 5.15 (MLP eval-

Class	Error limits (%)		Output variable				
	Min.	Max.	σ_J	σ_G	V_7	P_1	Q_1
1	0	2	90.64	84.37	95.16	96.70	93.62
2	2	4	6.69	8.75	4.43	2.78	5.15
3	4	6	1.13	3.60	0.10	0.21	0.72
4	6	8	0.41	1.13	0.00	0.31	0.10
5	8	10	0.21	0.51	0.00	0.00	0.00
6	10	12	0.10	0.51	0.00	0.00	0.21
7	12	14	0.51	0.10	0.21	0.00	0.00
8	14	16	0.10	0.72	0.10	0.00	0.00
9	16	18	0.00	0.10	0.00	0.00	0.00
10	18	20	0.00	0.00	0.00	0.00	0.10
11	20	22	0.00	0.00	0.00	0.00	0.00
12	22	24	0.00	0.00	0.00	0.00	0.00
13	24	26	0.00	0.00	0.00	0.00	0.10
14	26	28	0.00	0.21	0.00	0.00	0.00
15	28	30	0.00	0.00	0.00	0.00	0.00
Total	-	-	100.00	100.00	100.00	100.00	100.00

Table 5.13 - Histogram of relative error for Case A3 (% of total number of testing vectors, in each error class)

uation). All times were obtained on a Sun Sparc 2 workstation.

It should be noted that both load-flow and SVD times in Table 5.15 refer to program

Training session				
1	2	3	4	5
51	41	38	36	34

Table 5.14 - CPU time for MLP training in Case A3 (minutes)

1 load-flow	SVD (2 matrices)	971 testing vectors	1 testing vector
0.2	0.7	0.9	0.00093

Table 5.15 - CPU time for load-flow and MLP evaluation in Case A3 (seconds)

FLOW4, which was used to solve the 30-bus AEP base case with convergence to a maximum absolute power mismatch of 10^{-10} pu in 5 iterations. The CPU time for testing vectors refers to the processing phase of the MLP.

From Tables 5.13 and 5.15 it can be seen that the MLP delivers accurate results and also has a very low computing time. The main drawback of the backpropagation algorithm is its lengthy training phase, and therefore there is a trade-off between these conflicting factors.

5.5 - Summary

This chapter described a proposed methodology for assessing the static voltage stability of electric systems using artificial neural networks. This methodology was implemented through a self-contained computational system specifically developed for this purpose. The most important programs within the system were described in some detail.

The proposed methodology was validated through the use of the computational system in three different cases. Cases A1 and A2 showed good general performance of the methodology

from a qualitative point of view, whereas some quantitative considerations were discussed in Case A3, with particular reference to the evaluation error inherent to Multi-Layer Perceptron networks. Such error was less than or equal to 4% for at least 93% of the testing vectors. Also, the speed-up factor of the MLP with respect to the load-flow program was close to the value 1000 (computed as (0.2 + 0.7)/0.00093).

The conclusions of this chapter can be summarised as follows:

- the minimum singular value of matrix G_S is a more significant index for static voltage stability than the bus voltages. The latter remain fairly constant and only start to drop when the electric system is already undergoing potentially unstable processes. On the other hand, the minimum singular value of matrix G_S is more sensitive to such processes;

- the Multi-Layer Perceptron network is able to learn the physical relationships between the instability-inducing variables (i.e., bus load and branch outages) and the static voltage stability index;

- the advantages of using an MLP network for this electrical problem lie in its good accuracy and extremely fast computation, which make it a serious candidate for on-line applications. If the on-line use of sophisticated analytical tools is computationally very expensive, an MLP-based assessing tool can be seen as a convenient alternative for the problem at hand. Another major advantage of the neural network approach is the fact that the MLP learns only from examples that represent the physical problem, thus avoiding complex (or even unobtainable) analytical relationships between the electrical variables;

- the main disadvantage of the proposed approach is the lengthy training phase of the back-propagation algorithm. However, this does not pose a serious problem when the training can be carried out off-line, as is often the case.

The main improvement to be made to the present technique refers to the model of the electrical system. Dynamic factors of the VSP have not been considered here, and they represent an important aspect of the problem. In the next chapter, an attempt to include these dynamic factors in the model of the power system will be made.

CHAPTER 6

DYNAMIC ANALYSIS OF THE VOLTAGE STABILITY PROBLEM

6.1 - Introduction

This chapter constitutes an extension of the work developed in the previous chapter. The aim of this extension is to remove the main limitations of the static approach, through the use of dynamic simulations.

Initially, the outline of the method will be described, as well as its computational implementation. The core of this computational system is a dynamic simulation program, which will be described in its most relevant aspects. Two simple simulation cases will be presented in detail with the purpose of further illustrating some important mechanisms of voltage instability.

Mapping the results of a dynamic simulation study onto the ANN problem language is a complex task, and various alternatives can be envisaged. In this work, two different techniques were developed for implementing that mapping. These techniques were incorporated as two different interface programs acting between the dynamic simulation and the ANN programs.

The computational system was then used in some study cases. The results of these cases are presented separately, according to the interface program used in the mapping step.

At the end of the chapter, a general discussion of the proposed methodology and the results achieved will be presented.

6.2 - Outline of the method

The purpose of the method to be developed is to train a neural network to make it capable of

estimating some relevant parameters that can be extracted from a comprehensive dynamic simulation study.

The most important parameter considered here is the *simulation extra time* of a specific simulation scenario. This is the amount of time elapsed between the last time that a piece of information about the system was passed on to the neural network and the end of the simulation.

A dynamic simulation run can terminate either at its scheduled final time or before if a collapse occurs prematurely. If the neural network learns something from this process then, when presented with a real sequence of events and system states, it would be capable of predicting whether a collapse is likely to occur and also how much spare time there is before the collapse begins. This could be most valuable for system operators, who then would have time to enforce preventive actions in order to preserve the stability of the system.

The choice of the simulation technique was made because this approach allows a rich representation of the electrical system, as seen in Chapter 3. Another major reason for choosing this approach is the fact that in the present work the emphasis has been put on the issue of proximity to voltage collapse, as stated in Chapter 2. Thus, the simulation extra time appears to be an adequate solution for the problem at hand.

The most important drawback of the simulation technique is perhaps its high computing times. The choice of the MLP architecture was made with the aim of overcoming this limitation, as well as exploring the interpolation capabilities of this neural network model.

6.3 - Computational system

6.3.1 - Introduction

This section describes the computational system that was developed for analysing the voltage stability of power systems from a dynamic viewpoint. The system was implemented using the *C* language and is currently operating on Unix-based workstations in the Department of Electronic Engineering, QMW.

Figure 6.1 shows the structure of the computational system. It can be seen that it consists of six separate modules and four different types of permanent files. This files allow information to be exchanged among the programs. It should be noted that programs FLOW4, STATISTIC1 and TRAININGSET1 are exactly the same programs as described in Chapter 5, and therefore only the other programs will be described in detail in this chapter.

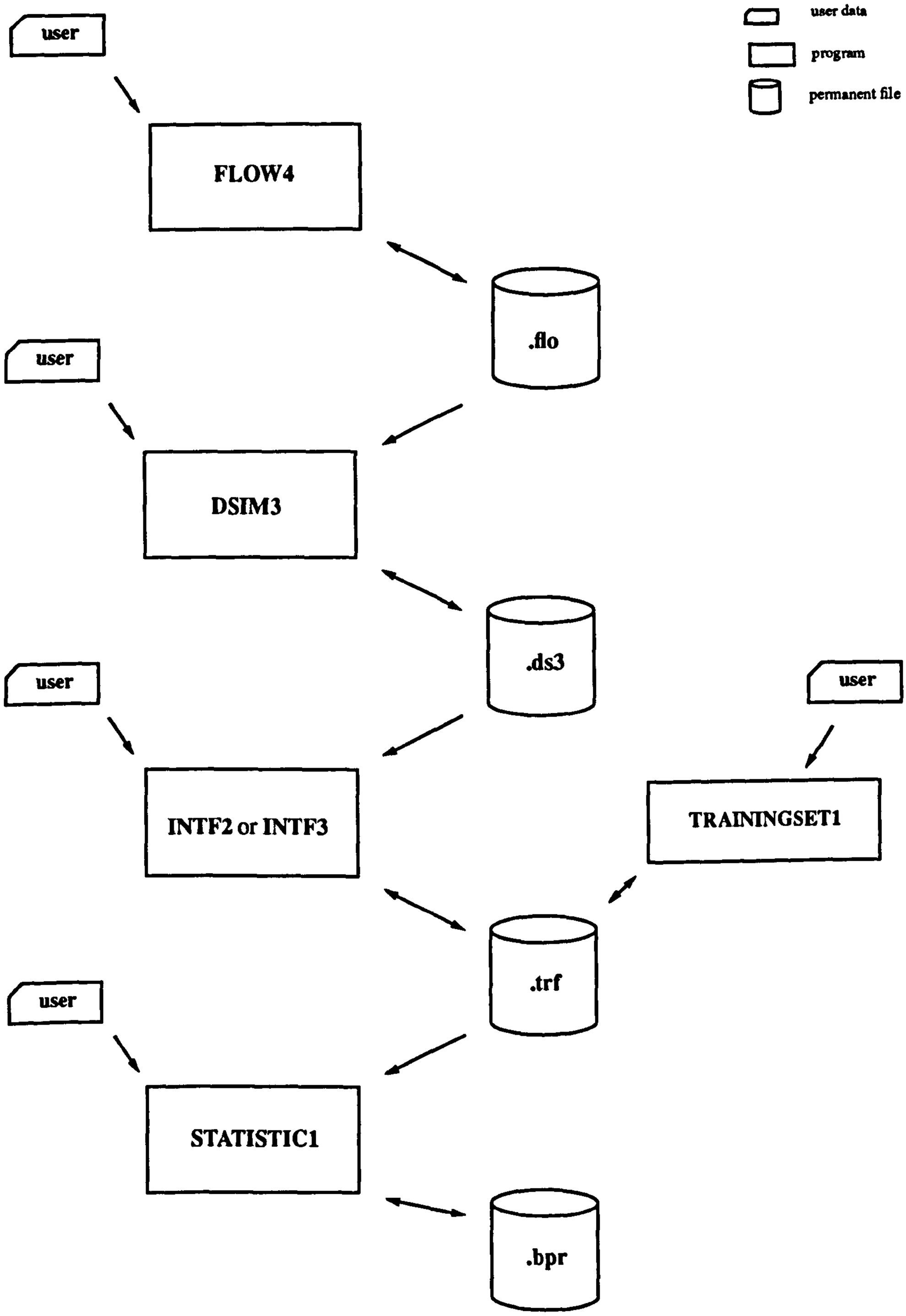


Figure 6.1 - Computational system for dynamic voltage stability assessment
(output results from all programs not shown)

Program **FLOW4** is the module responsible for load-flow calculations. It reads data from an external user file and, on completion of these calculations, it can store a complete case in a binary file with file extension "**.flo**". Binary files of type **.flo** can be read by program FLOW4 and also by the dynamic simulation program DSIM3.

Program **DSIM3** executes dynamic simulations of power systems. It reads the basic network structure from **.flo** files and reads additional data from an external user file. These additional data refer to those system components that exhibit dynamic behaviour, such as generators, induction motors, dynamic loads and on-load tap changers, which are not present in the load-flow file. Program DSIM3 allows storing/retrieving complete dynamic simulation cases in/from binary files with extension "**.ds3**"; these files can be read by program DSIM3 when creating brand new cases from existing cases, and also by the interface programs INTF2 and INTF3.

Programs **INTF2** and **INTF3** are the interface modules between the dynamic simulation and the neural networks programs (DSIM3 and STATISTIC1 respectively). They acquire a dynamic simulation case from a **.ds3** file and construct a complete training/testing set for the neural network. This training/testing set can then be stored in binary files with extension "**.trf**"; these files can be read by either program INTF2 or INTF3 for creating brand new cases from existing cases, and also by program STATISTIC1. It should be pointed out that program INTF2 implements the first methodology that was developed for mapping the results of the dynamic simulation onto the neural networks problem language, while program INTF3 implements the second methodology developed.

Program **STATISTIC1** is responsible for both training and operation of MLP networks. Training/testing sets can be given either manually by the user or through a **.trf** file previously generated by the interface programs INTF2 and INTF3. This program also has facilities for storing/retrieving trained networks in/from binary files with extension "**.bpr**".

Finally, module **TRAININGSET1** can be used for generating training/testing sets in special user-oriented applications. Specific modelling features can be incorporated into the core program as *C* functions. This program can store the generated sets in **.trf** files for later use by the program itself and by the neural networks program (STATISTIC1).

Program DSIM3 will be described in detail in the next sub-section. Two examples of dynamic simulations will be included and discussed. These examples illustrate some important aspects of the VSP.

At the end of this section, programs INTF2 and INTF3 will be introduced. The specific description of both programs, together with corresponding study cases, will be presented separately in the following sections.

6.3.2 - Dynamic simulation program (DSIM3)

6.3.2.1 - Introduction

In this sub-section, the main modelling features incorporated in program DSIM3 will be presented. These models were already available in the technical literature and they were adapted to suit the requirements of the present work. The models correspond to generators, induction motors, on-load tap changers, and loads. The definition of the transient operations list and the output set from the program will also be presented. At the end of the sub-section, the solution of the network equations, the numerical integration methods used in the program, and the overall procedure for taking into account all the individual models will be discussed.

6.3.2.2 - Generators

In terms of program DSIM3, buses in the power system must be of one of the types defined in Table 6.1.

Bus type	Purpose
3	To represent generator internal buses
2	To represent generator external (or "terminal") buses
1	To represent all other buses

Table 6.1 - Types of buses considered in the dynamic simulation program

Buses of types 2 and 3 are used in the "voltage-behind-reactance" generator model, Figure 6.2. This model is used in classical transient stability studies [135].

In order to incorporate the generator model in the load-flow network structure (read from the load-flow file), the following requirements apply to buses 2 and 3:

- bus 2 must exist in the load-flow network;
- bus 3 must not exist in the load-flow network (it is a new bus).

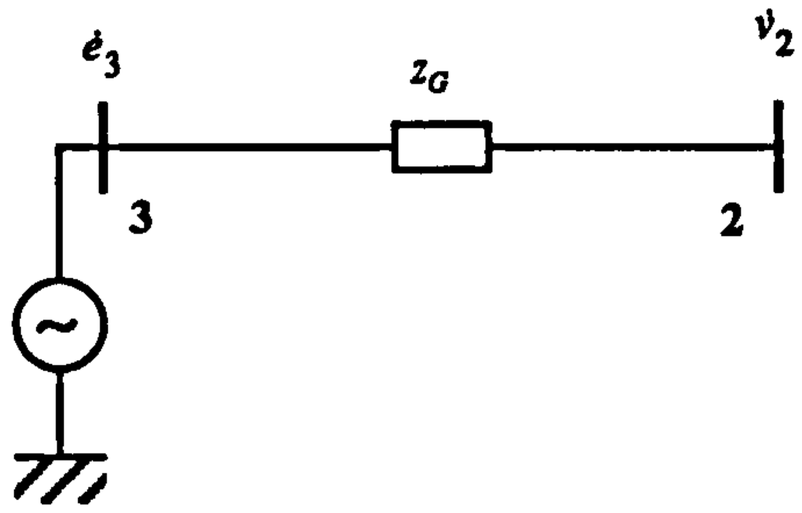
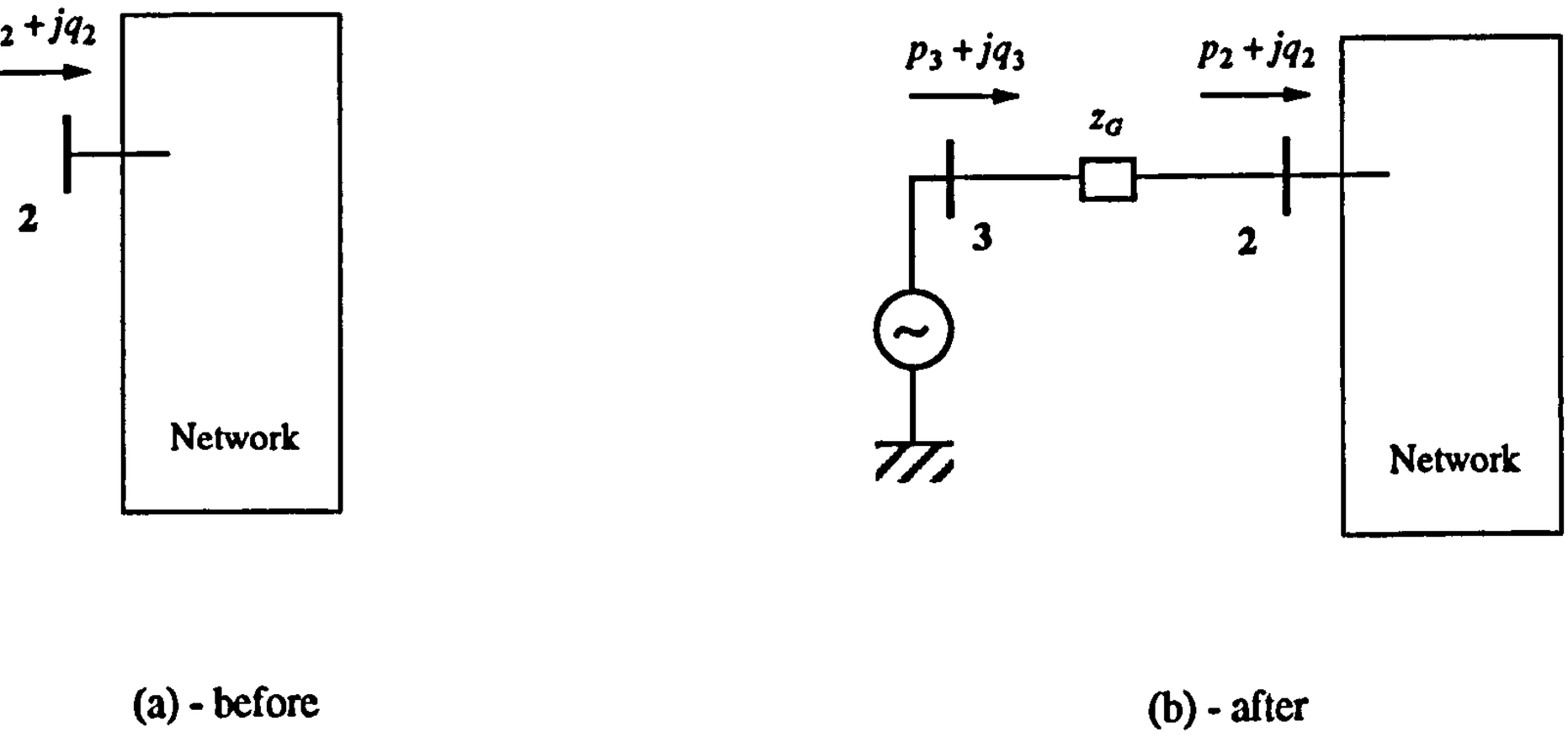


Figure 6.2 - Electrical model of a generator

The first step is to compute an equivalent injection at bus 3 ($p_3 + jq_3$) such that the original power flow from bus 2 to the network ($p_2 + jq_2$) remains unaltered. Figure 6.3 illustrates this point.



(a) - before

(b) - after

Figure 6.3 - Inclusion of internal node 3 in the electrical network

The injection $p_3 + jq_3$ and the voltage $\dot{e}_3$ (magnitude e_3 and angle δ_3) are easily computed from the knowledge of the impedance z_G, the previous voltage $\dot{v}_2 = v_2\angle\theta_2$ and the previous injection $p_2 + jq_2$. This yields the initial value of the generator state variable δ_3, which will be required in the integration of the dynamic equation.

The dynamic equation of a generator, also known as *swing equation*, describes the movement of the generator rotor with respect to a rotating reference [135], and is given by:

$$\frac{d}{dt}\omega(t) = \frac{\omega_R}{2H}(p_m(t) - p_e(t)) - \frac{D\omega_R}{2H}\omega(t) \qquad (6.1\text{ a})$$

$$\frac{d}{dt}\delta(t) = \omega(t) - \omega_R \qquad (6.1\text{ b})$$

with initial conditions:

$$\omega(0) = \omega_R$$
$$\delta(0) = \delta_3$$

where
- $\omega(t)$ = rotor angle speed (rad/s);
- $\delta(t)$ = rotor angle (rad);
- H = rotor inertia constant (s);
- ω_R = system angular reference (rad/s);
- $p_m(t)$ = input (mechanical) power to the machine (pu);
- $p_e(t)$ = output (electrical) power from the machine (pu);
- D = damping coefficient.

The difference $(p_m(t) - p_e(t))$ is called *accelerating power* and is the driving force of rotor oscillations, since a change on either the electrical power or the mechanical power alters the balance between input and output powers. A change in the electrical power may result as a consequence of a network disturbance such as three-phase short-circuit, whereas a change in the mechanical power may result from the action of the governor subsystem, which tries to keep the angular speed of the rotor at the value ω_R.

In this work the governor subsystem is modelled in a rather simple way. Table 6.2 presents the modelling alternatives for the governor.

Model	Description	Equation	Initial condition	Remarks
1	Constant mechanical power	$p_m(t) = p_e(0), \forall t$	-	$p_e(0)$ is the electrical power generated by the machine on initial conditions
2	Mechanical power equal to electrical power	$p_m(t) = p_e(t), \forall t$	-	Generator is an "infinite bus"

Table 6.2 - Governor models

Model	Description	Equation	Initial condition	Remarks
3	First-order governor	$\frac{d}{dt}p_m(t) = K_G(\omega_R - \omega(t))$	$p_m(0) = p_e(0)$	1. $p_e(0)$ as in model (1) 2. K_G = governor constant

Table 6.2 - Governor models

Finally, the excitation subsystem, which tries to keep the terminal voltage v_2 at a constant value through action on the field winding, is modelled as proposed in reference [46]. According to that work, the dynamics of the excitation subsystem are much faster than the voltage stability dynamics being studied. Therefore, the excitation subsystem renders the voltage v_2 constant for voltage stability purposes. Note that it is v_2 (the generator external voltage) that is assumed constant, and not e_3 (the generator internal voltage), as in classical transient stability studies. The consequences of this assumption will be discussed later in the sub-section corresponding to the solution of the network equations (sub-section 6.3.2.8).

6.3.2.3 - Induction motors

Figure 6.4 shows the equivalent circuit of induction motors used in program DSIM3. It can

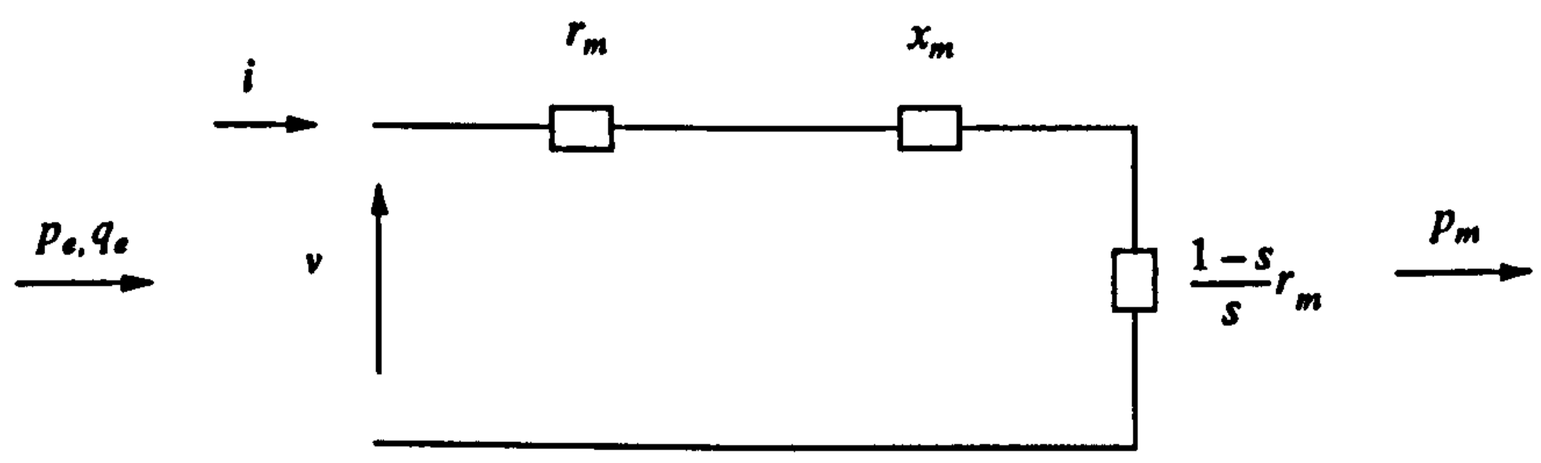


Figure 6.4 - Equivalent circuit of induction motors

be shown that the electric active and reactive powers absorbed by the motor are given by:

$$p_e = \frac{\frac{r_m}{s}}{\left[\frac{r_m}{s}\right]^2 + x_m^2} \cdot v^2 \qquad (6.2\ a)$$

$$q_e = \frac{x_m}{\left[\frac{r_m}{s}\right]^2 + x_m^2} \cdot v^2 \qquad (6.2\ b)$$

where p_e = active power absorbed by the motor (pu);
q_e = reactive power absorbed by the motor (pu);

$s = \frac{\omega - \omega_R}{\omega_R}$ = motor slip (pu);

ω = angular speed of rotor (rad/s);
ω_R = system angular reference (rad/s);

The dynamic equation for the motor is the same as adopted in reference [136], and is given by:

$$\frac{d}{dt}s(t) = \frac{\omega_R}{2H} \cdot \left[\frac{p_m}{1-s} - p_e\right] \qquad (6.3)$$

where H = inertia constant of the rotating parts (s);
p_m = mechanical power delivered by the motor (pu);

In the equivalent circuit of Figure 6.4, the mechanical power corresponds to the electrical power absorbed by the resistance $\frac{1-s}{s} r_m$. In this case, it can be shown that:

$$p_m = (1\text{-}s).p_e \qquad (6.4)$$

The diagram of Figure 6.5 shows the procedure for obtaining the mechanical power from the initial condition of the system. It should be noted that in program DSIM3 the value of the mechanical power remains constant throughout the dynamic simulation.

6.3.2.4 - On-load tap changers

In program DSIM3, automatic on-load tap changers of transformers (OLTCs) are modelled according to reference [52]. The controlling device of the transformer tap executes periodic checks on the voltage of a controlled bus with respect to a pre-defined set-point value. At the end of a

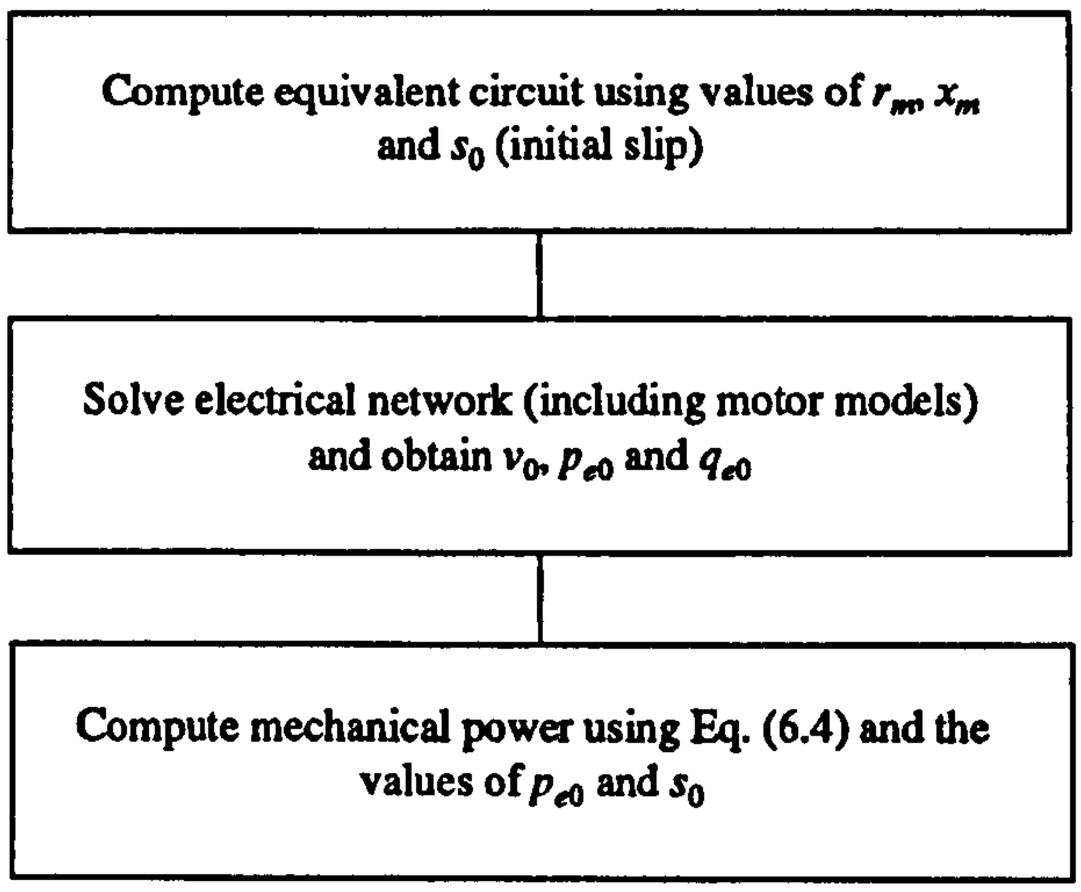


Figure 6.5 - Computation of the mechanical power of induction motors

checking period (or *cycle*), if the difference between the actual bus voltage and the set-point value is larger than a tolerance, and if there are still available tap positions, then the transformer tap is increased or decreased by one tap step, depending on the sign of this difference. Otherwise, the tap position of the OLTC is kept unchanged.

The behaviour of this discrete-time, discrete-tap device is summarised by the following equations:

$$tap(n+1) = tap(n) + d.f(v_b - v_s) \tag{6.5 a}$$

$$f(v_b - v_s) = \begin{cases} 1 & \text{if } v_b - v_s < -\varepsilon \\ 0 & \text{if } |v_b - v_s| \leq \varepsilon \\ -1 & \text{if } v_b - v_s > \varepsilon \end{cases} \tag{6.5 b}$$

where
$tap(n)$ = value of tap at the end of checking cycle n (pu);
d = step size of the tap changer (pu);
v_b = actual voltage of controlled bus (pu);
v_s = set-point value for voltage (pu);
$f(v_b - v_s)$ = control function;
ε = voltage tolerance (pu).

It is assumed that the transformer ratio is 1:$tap(n)$, and also that an increase on the tap value pro-

duces an increase on the voltage of the controlled bus. Another important parameter of this model is the cycle time T, which is the elapsed time between two consecutive voltage checks.

6.3.2.5 - Loads

Loads in program DSIM3 can be either static or dynamic. Static loads are modelled as voltage-dependent injections. The equations in this case are the same as in the load-flow program FLOW4 (see sub-section 5.3.2).

Dynamic loads are modelled by the following first-order differential equation:

$$\frac{d}{dt}g_l(t) = K_l(p_{ref} - p_{actual}(t)) \tag{6.6}$$

with initial condition $g_l(0) = \frac{p_{ref}}{v_0^2}$

where $g_l(t)$ = load conductance, adjusted by the load controller (pu);
K_l = constant that allows the load response speed to be controlled;
p_{ref} = reference active power (or set point) (pu);
$p_{actual}(t)$ = actual active power absorbed by the load (pu);
$g_l(0)$ = initial value for the load conductance (pu);
v_0 = initial value of load voltage (pu).

At any time t, the following relationship between conductance and active power holds:

$$p_{actual}(t) = v^2.g_l(t) \tag{6.7 a}$$

$$q_{actual}(t) = p_{actual}(t).tan\varphi \tag{6.7 b}$$

where v = voltage at the load bus;
$cos\varphi$ = power factor of dynamic load, assumed to be invariant.

From Eqs. (6.6) and (6.7), it can be seen that the dynamic load possesses a constant-impedance instantaneous response. The first-order controller tries to adjust the load conductance so as to maintain the active power absorbed at the value p_{ref}.

6.3.2.6 - Operations data

In a dynamic simulation program, the operations data list describes the events, and their associated times, that constitute the transient period under consideration. Table 6.3 shows the types of events recognised by program DSIM3.

Type of event	Description
1	Short-circuit at a bus
2	Opening of a branch
3	Closing of a branch
4	Multiplication of all loads by a global factor
5	Multiplication of all generation by a global factor
6	Modification of individual loads
7	Modification of individual generation
8	Modification of individual reactors

Table 6.3 - Operations recognised by program DSIM3

6.3.2.7 - Output data

The output data section specifies which variables are to be printed after the dynamic simulation is completed. In program DSIM3, output variables are printed as a function of time in both tabulated and graphics forms. Table 6.4 shows the options available for printing output variables with the program.

Output variable	Description
1	Minimum Singular Value of matrix G_s (σ_G)
2	Minimum Singular Value of matrix J (σ_J)
3	Angular speed of generators

Table 6.4 - Output variables printed by program DSIM3

Output variable	Description
4	Angle of generators
5	Angular difference between generators
6	Mechanical power of generators
7	Bus voltage
8	Bus angle
9	Bus active load
10	Bus reactive load
11	Bus active generation
12	Bus reactive generation
13	Bus reactor
14	Motor slip
15	Motor active power
16	Motor reactive power
17	Tap of automatic OLTC
18	Active power of dynamic loads
19	Reactive power of dynamic loads

Table 6.4 - Output variables printed by program DSIM3

6.3.2.8 - Solution of network equations

In program DSIM3, the equations corresponding to the electrical network are solved using the Newton-Raphson method, as in the load-flow program. However, due to the bus type definition adopted in program DSIM3 (see sub-section 6.3.2.2), the Jacobian matrix in this case is slightly different from the load-flow Jacobian. Table 6.5 shows the types of buses for program DSIM3, as well as the classification of variables in the solution of the network equations.

Bus type	Purpose	N° of buses	Known variables during solution of network eqs.	Unknown variables during solution of network eqs.
1	To represent load buses	n_1	p, q	v, θ

Table 6.5 - Types of buses and definition of variables for the solution of network equations

Bus type	Purpose	N° of buses	Known variables during solution of network eqs.	Unknown variables during solution of network eqs.
2	To represent generator terminal buses	n_2	p, q, v	θ
3	To represent generator internal buses	$n_3 = n_2$	θ	v, p, q

Table 6.5 - Types of buses and definition of variables for the solution of network equations

During the solution of the network equations, the only unknown variable for generator terminal buses is the angle θ, because the voltage v is assumed to be constant due to the action of the excitation subsystem. Also, the values of active and reactive generation are zero because these injections were transferred to the internal bus when the internal bus was created (see sub-section 6.3.2.2).

With respect to the generator internal buses, the only known variable during the solution of network equations is the angle θ. This angle is obtained from the solution of the generator dynamic equation, which is a prior step in the overall solution. Hence, when the solution of network equations starts, the value of θ for all buses of type 3 is already available, and it remains fixed throughout the solution of the network equations. This will be further explained later in this sub-section. Lastly, the value of active and reactive injections for buses of type 3 is obtained after convergence of the Newton-Raphson method.

From the explanation above, the linearised system of network equations to be solved is given by Eq. (6.8).

$$\begin{bmatrix} \Delta P_1 \\ \Delta P_2 \\ \Delta Q_1 \\ \Delta Q_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial P_1}{\partial \theta_1} & \frac{\partial P_1}{\partial \theta_2} & \frac{\partial P_1}{\partial V_1} & 0 \\ \frac{\partial P_2}{\partial \theta_1} & \frac{\partial P_2}{\partial \theta_2} & \frac{\partial P_2}{\partial V_1} & \frac{\partial P_2}{\partial V_3} \\ \frac{\partial Q_1}{\partial \theta_1} & \frac{\partial Q_1}{\partial \theta_2} & \frac{\partial Q_1}{\partial V_1} & 0 \\ \frac{\partial Q_2}{\partial \theta_1} & \frac{\partial Q_2}{\partial \theta_2} & \frac{\partial Q_2}{\partial V_1} & \frac{\partial Q_2}{\partial V_3} \end{bmatrix} \cdot \begin{bmatrix} \Delta \theta_1 \\ \Delta \theta_2 \\ \Delta V_1 \\ \Delta V_3 \end{bmatrix} \tag{6.8}$$

In this equation, submatrices $\frac{\partial P_1}{\partial V_3}$ and $\frac{\partial Q_1}{\partial V_3}$ become equal to the null matrix because it is assumed

that buses of type 1 do not connect to buses of type 3. Finally, it should be pointed out that the derivatives of the Jacobian matrix in Eq. (6.8) are obtained in the same way as in the load-flow program (see load-flow equations in sub-section 5.3.2).

During the solution of network equations, program DSIM3 also checks for reactive power violations at generator buses. This is done by computing the reactive power flow between buses 3 and 2, Figure 6.6.

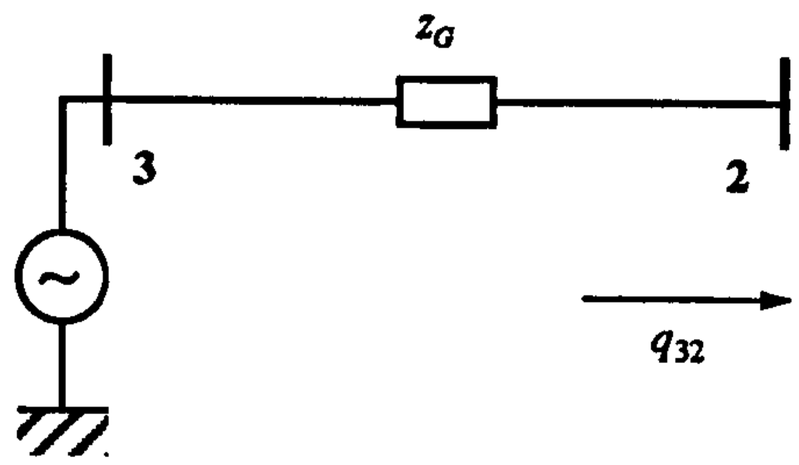


Figure 6.6 - Reactive power limit check for generators

It can be shown that the reactive power flow, q_{32}, is given by:

$$q_{32} = v_2 v_3 y \sin(\theta_2 - \delta_3 + \alpha) - v_2^2 y \sin\alpha \qquad (6.9)$$

where $\dot{y} = \frac{1}{\dot{z}} = \frac{1}{z\angle\alpha} = y\angle -\alpha$ is the admittance (pu) linking buses 2 and 3.

Whenever a reactive power limit violation is detected, bus 2 is transformed from type *PQV* into type *PQ* by adding the linearised version of Eq. (6.9) to the system of equations (6.8) and solving for v_2 in each subsequent iteration. The resulting system of equations is given by Eq. (6.10). In this case, voltage v_2 will be adjusted so as to eliminate the reactive power overload, and will stay at this new value during the remaining simulation time. Only if a new reactive power overload occurs will voltage v_2 be adjusted again.

6.3.2.9 - Numerical integration methods

$$\begin{bmatrix} \Delta P_1 \\ \Delta P_2 \\ \Delta Q_1 \\ \Delta Q_2 \\ \hdashline \Delta q_{32} \end{bmatrix} = \left[\begin{array}{cccc:c} \frac{\partial P_1}{\partial \theta_1} & \frac{\partial P_1}{\partial \theta_2} & \frac{\partial P_1}{\partial V_1} & 0 & \frac{\partial P_1}{\partial V_2} \\ \frac{\partial P_2}{\partial \theta_1} & \frac{\partial P_2}{\partial \theta_2} & \frac{\partial P_2}{\partial V_1} & \frac{\partial P_2}{\partial V_3} & \frac{\partial P_2}{\partial V_2} \\ \frac{\partial Q_1}{\partial \theta_1} & \frac{\partial Q_1}{\partial \theta_2} & \frac{\partial Q_1}{\partial V_1} & 0 & \frac{\partial Q_1}{\partial V_2} \\ \frac{\partial Q_2}{\partial \theta_1} & \frac{\partial Q_2}{\partial \theta_2} & \frac{\partial Q_2}{\partial V_1} & \frac{\partial Q_2}{\partial V_3} & \frac{\partial Q_2}{\partial V_2} \\ \hdashline 0 & \frac{\partial q_{32}}{\partial \theta_2} & 0 & \frac{\partial q_{32}}{\partial V_3} & \frac{\partial q_{32}}{\partial V_2} \end{array}\right] \cdot \begin{bmatrix} \Delta \theta_1 \\ \Delta \theta_2 \\ \Delta V_1 \\ \Delta V_3 \\ \hdashline \Delta V_2 \end{bmatrix} \tag{6.10}$$

The first method used for integrating the differential equations associated with generators, induction motors and dynamic loads is the *Modified Euler Method* (MEM) [135]. This method is very simple and at the same time it offers good numerical accuracy.

The MEM consists of two separate stages, namely a predictor and a corrector stage. The predictor stage is executed only once, and it gives the first estimate for the next value of the function being integrated. The corrector stage is iterative and repeatedly improves the estimate produced by the predictor. Supposing that the following first-order differential equation

$$\dot{x}(t) = f(t, x(t))$$

has been already integrated until time t inclusive, then the computation of the next value of the function is given by the procedure in the block diagram of Figure 6.7. Table 6.6 contains the key for the symbols in this figure.

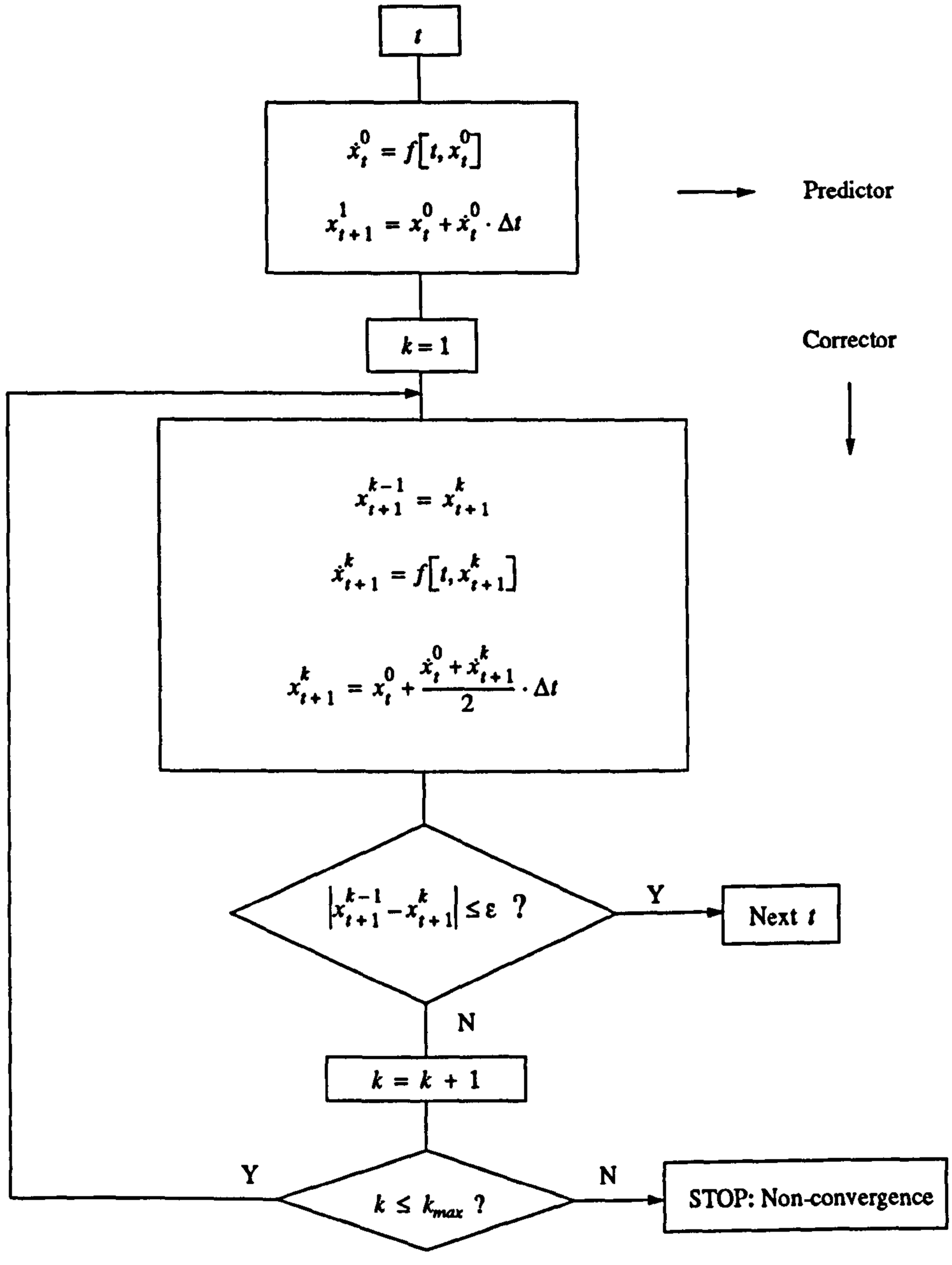


Figure 6.7 - One time-step computation using the MEM

From Figure 6.7, it can be seen that the MEM alternatively refines the value of the derivative at time t+1 ($\dot{x}_{t+1}^k$) and then refines the value of the function at time t+1 (x_{t+1}^k). When two subsequent values of the function are equal (within a tolerance), then the corrector stage is finished and the next time step will be selected for computation.

Symbol	Meaning
$\dot{x}_t^0$	Value of the derivative at time t (correct value)
x_t^0	Value of the function being integrated at time t (correct value)
$\dot{x}_{t+1}^k$	Value of the k^{th} estimate for the derivative at time $t+1$
x_{t+1}^k	Value of the k^{th} estimate for the function at time $t+1$
ε	Tolerance for the iterative process
k_{max}	Maximum number of iterations
Δt	Integration step

Table 6.6 - Key to symbols in Figure 6.7

Another numerical method was implemented and tested in program DSIM3. This second method is non-iterative and belongs to the family of the so-called *4th order Runge-Kutta* methods (RKM) [12]. Figure 6.8 shows the block diagram of this method.

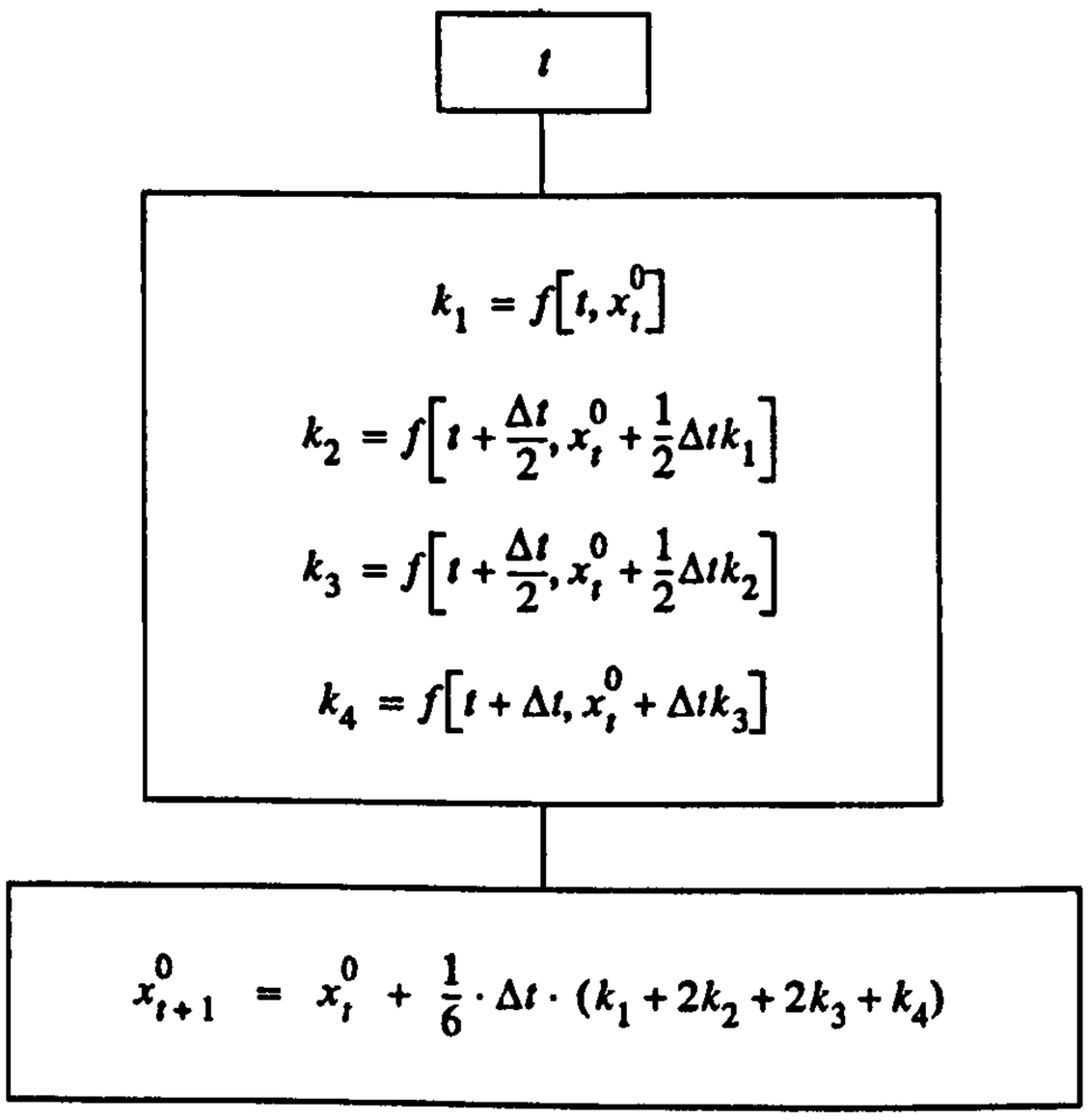


Figure 6.8 - 4^{th} order Runge-Kutta method (meaning of symbols given in Table 6.6)

Both MEM and RKM produced the same results when applied to many different test cases. However, differences occurred with respect to processing times. Two different machines were used, namely a Sun Sparc ELC workstation and a PC-type personal computer (PC 486 DX2 66 MHz), both running Unix operating systems. Table 6.7 shows a summary of processing times. It is interesting to note that the two methods compare differently depending on the machine being utilised.

Machine/method -->	Sun Sparc ELC		PC 486 DX2 66 MHz	
Case	MEM	RKM	MEM	RKM
A	177.30	188.50	192.65	186.81
B	642.60	686.90	701.88	679.84

Table 6.7 - CPU times (s) for different cases, integration methods, and machines

6.3.2.10 - Overall procedure

Figure 6.9 shows a block diagram of the dynamic simulation procedure in program DSIM3.

With reference to Figure 6.9, the initialisation procedure allows the consideration of either brand new cases or continuation of simulation cases previously executed.

The execution of specified operations means updating the state of the system according to the operations list (see sub-section 6.3.2.6). This means updating the status of branches and scheduled power of loads and reactors.

Checking OLTCs refers to verifying whether or not any OLTC has exhausted its cycle time at the current time *t*. If this is true, the tap value will be updated, if required by the voltage of the controlled bus and subject to the existence of available tap positions.

Updating matrix Y-bus occurs whenever a change in the network takes place (such as opening or closing of a branch), as well as when an OLTC has its tap position changed.

After the state variables of generators, induction motors, dynamic loads and OLTCs have been updated, the network equations are solved in order to compute new bus voltages and angles across the power system. With these new voltages and angles, the electrical power (active and

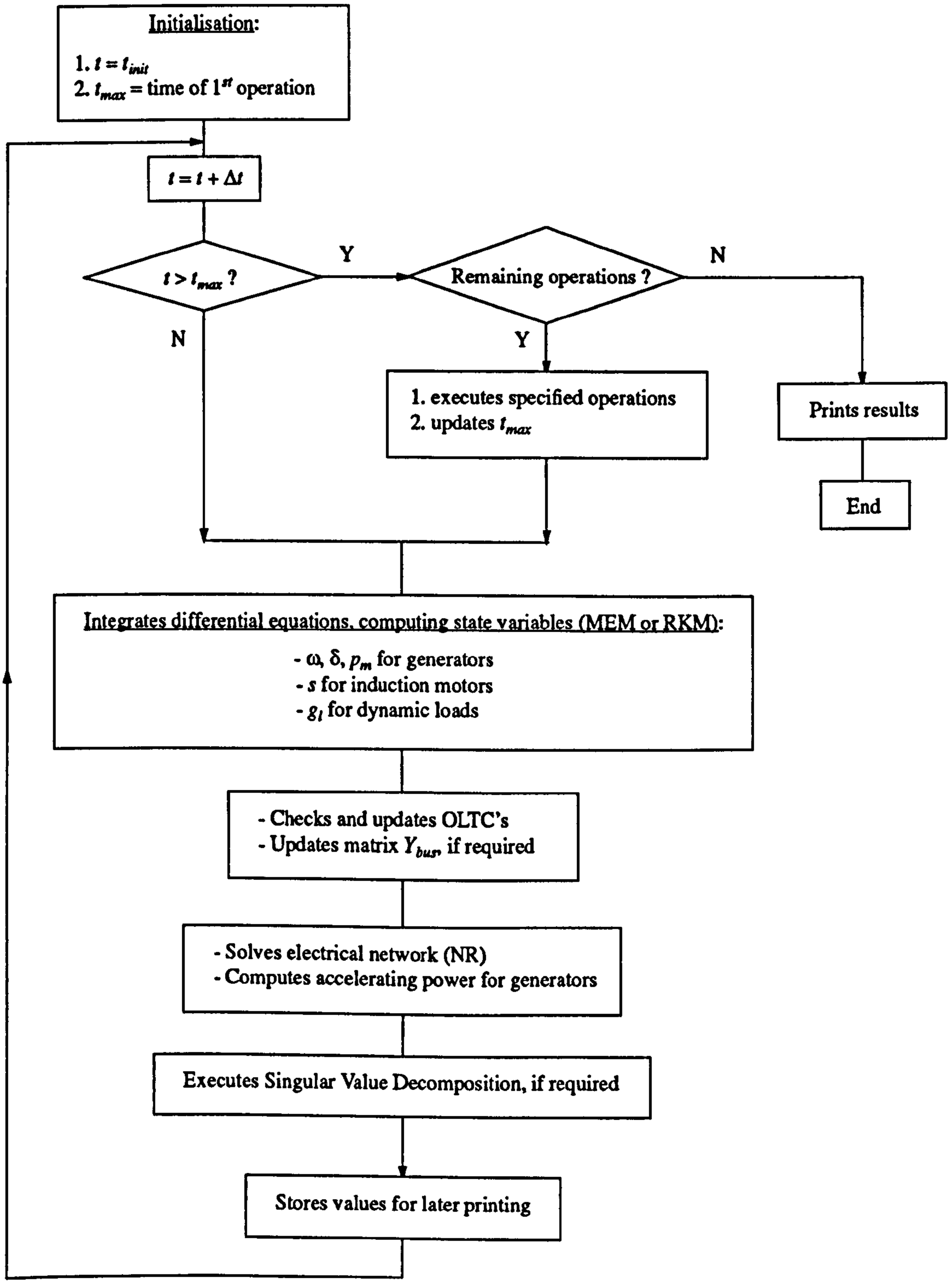


Figure 6.9 - Block diagram of dynamic simulation procedure

reactive) of generators is computed. The active power of generators is then used to evaluate the accelerating power, which will be used in the next integration step.

In the solution of network equations, the Jacobian matrix of the system (Eqs. (6.8) and (6.10)) must be assembled and factorised for each Newton-Raphson iteration for each time step in the dynamic simulation. Despite the use of linked lists for storing the Jacobian matrix, a large proportion of computing time is spent in these two tasks, particularly in the factorisation step. Assembling and factorising the Jacobian matrix only when structural changes occur in the network (such as branch opening or closing) proved to be extremely beneficial: savings in the order of 60 - 70% of the total CPU time were achieved with this simple technique (which is also known as *Very Dishonest Newton-Raphson Method*). Using a factorised Jacobian matrix that does not accurately reflect the actual state of the system may imply in a larger number of iterations for the Newton-Raphson method to converge, but this extra cost is largely offset by the savings in the factorisation subroutine.

6.3.3 - Examples of dynamic simulation

6.3.3.1 - Introduction

In this sub-section, the use of program DSIM3 in two different cases will be described. These cases are presented before the introduction of neural networks in the dynamic analysis of the VSP with the purpose of illustrating some important mechanisms associated with voltage instabilities.

The electrical network that will be used is shown in Figure 6.10. This network is derived from the 9-bus system found in chapter 2 of reference [135]. A new, radial subsystem, comprising buses 1000 through 1501, was connected to the original system at bus 400. This radial subsystem contains dynamic loads and OLTCs and will be used to study some specific transients.

Table 6.8 shows the main data regarding the buses in the network. Initial voltages and angles are also included. Tables 6.9, 6.10 and 6.11 present the main data for induction motors, dynamic

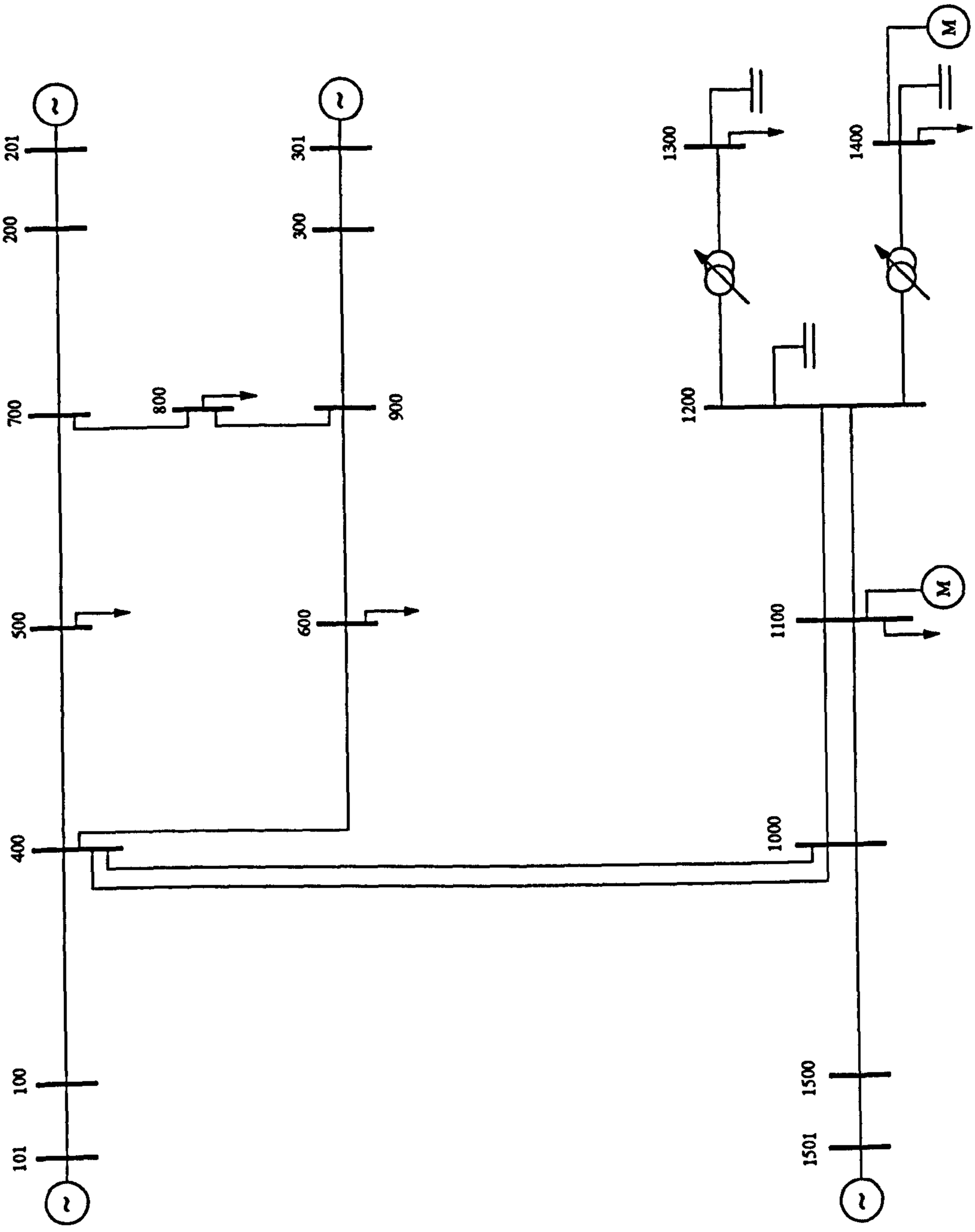


Figure 6.10 - Electrical network for study cases

Num.	Type	Rating (kV)	Volt. (pu)	Angle (deg.)	Load		Generation		Capac. (MVAr)	*Q*-limits (MVAr)	
					P (MW)	*Q* (MVAr)	*P* (MW)	*Q* (MVAr)		Min.	Max.
100	G.term.	16.5	1.040	0.159	0.	0.	94.472	27.686	0.	-40.000	40.000
101	G. int.	16.5	1.058	3.153	0.	0.	0.	0.	0.	-	-
200	G.term.	18.0	1.025	8.481	0.	0.	161.486	6.337	0.	-30.000	30.000
201	G. int.	18.0	1.050	18.841	0.	0.	0.	0.	0.	-	-
300	G.term.	13.8	1.025	3.888	0.	0.	83.772	-10.984	0.	-30.000	30.000
301	G. int.	13.8	1.016	12.272	0.	0.	0.	0.	0.	-	-
400	Load	230.0	1.026	-2.764	0.	0.	0.	0.	0.	-	-
500	Load	230.0	0.996	-4.602	125.000	50.000	0.	0.	0.	-	-
600	Load	230.0	1.013	-4.300	90.000	30.000	0.	0.	0.	-	-
700	Load	230.0	1.026	2.973	0.	0.	0.	0.	0.	-	-
800	Load	230.0	1.016	-0.015	100.000	35.000	0.	0.	0.	-	-
900	Load	230.0	1.032	1.229	0.	0.	0.	0.	0.	-	-
1000	Load	230.0	1.025	-2.948	0.	0.	0.	0.	0.	-	-
1100	Load	230.0	0.987	-5.768	57.255	36.128	0.	0.	0.	-	-
1200	Load	230.0	0.979	-6.835	0.	0.	0.	0.	4.788	-	-
1300	Load	13.8	0.971	-8.042	20.000	15.000	0.	0.	7.537	-	-
1400	Load	18.0	0.966	-9.284	33.649	14.325	0.	0.	4.662	-	-
1500	G.term.	13.8	1.040	-0.445	0.	0.	93.087	34.221	0.	-60.000	60.000
1501	G. int.	13.8	1.061	2.456	0.	0.	0.	0.	0.	-	-
Total					425.905	180.452	432.817	57.260	16.987	-	-

Table 6.8 - Bus data (load-flow solved for initial conditions)

loads and OLTCs, respectively. It should be pointed out that the complete set of data for this system can be found in the Appendix.

Bus	Rating (MVA)	*H* (s)	Initial slip (pu)	P_{elec} (MW)	Q_{elec} (MVAr)	P_{mech} (MW)
1100	35	10.000	0.030	27.255	13.628	26.438
1400	25	10.000	0.030	18.649	9.325	18.090

Table 6.9 - Induction motor data

Bus	Constant K_l	P (MW)	Q (MVAr)
1100	0.400	30.000	22.500
1300	0.400	20.000	15.000

Table 6.10 - Dynamic load data

Branch	Taps above tap 1.0	Taps below tap 1.0	Tap step (pu)	Initial tap	Cycle (s)	Set-point voltage (pu)	Voltage tol-erance (pu)	Controlled bus
1200 - 1300	5	5	0.010	0	4.50	0.971	0.0050	1300
1200 - 1400	5	5	0.010	0	4.50	0.966	0.0050	1400

Table 6.11 - OLTC data

6.3.3.2 - Case 1

The first study case of dynamic simulation corresponds to the transient analysis specified in Table 6.12. The first disturbance was scheduled 1.5s after the beginning of the simulation in order

N° of operation	Time (s)	Operation specification
1	-1.0	Start of analysis
2	0.5	Capacitors at buses 1200, 1300 and 1400 discon-nected
3	7.0	Opening branch 1100-1200 (circuit #1)
4	20.0	End of analysis

Table 6.12 - Transient operations list (Case 1)

to check the behaviour of the program in steady-state conditions (between t = -1.0 and t = 0.5s).

Figures 6.11, 6.12 and 6.13 show the temporal evolution of σ_G, v_{1200} and v_{1400} respectively. It can be seen that the voltage behaviour at bus 1400 is substantially different from bus 1200. This is

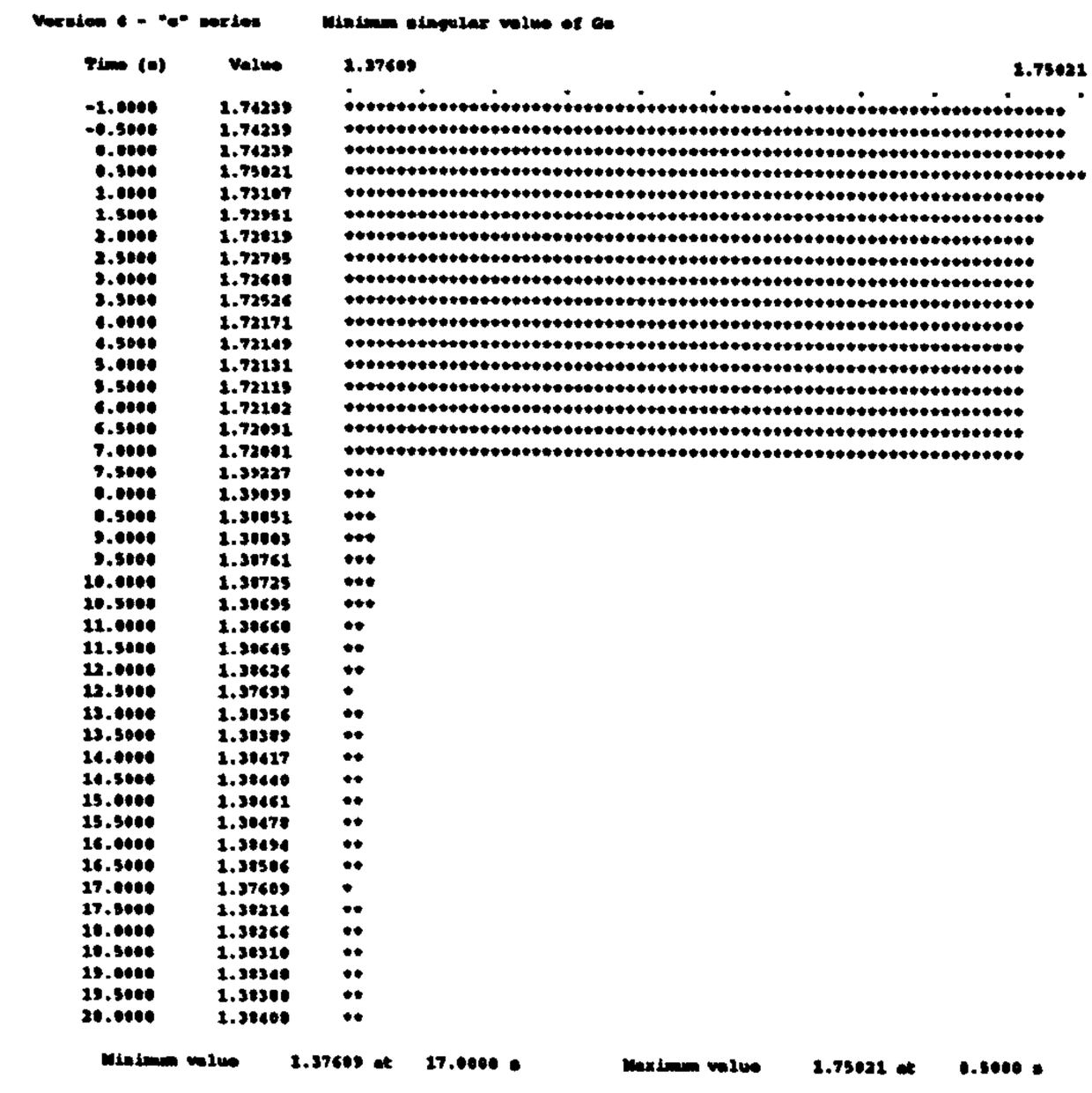


Figure 6.11 - Evolution of σ_G

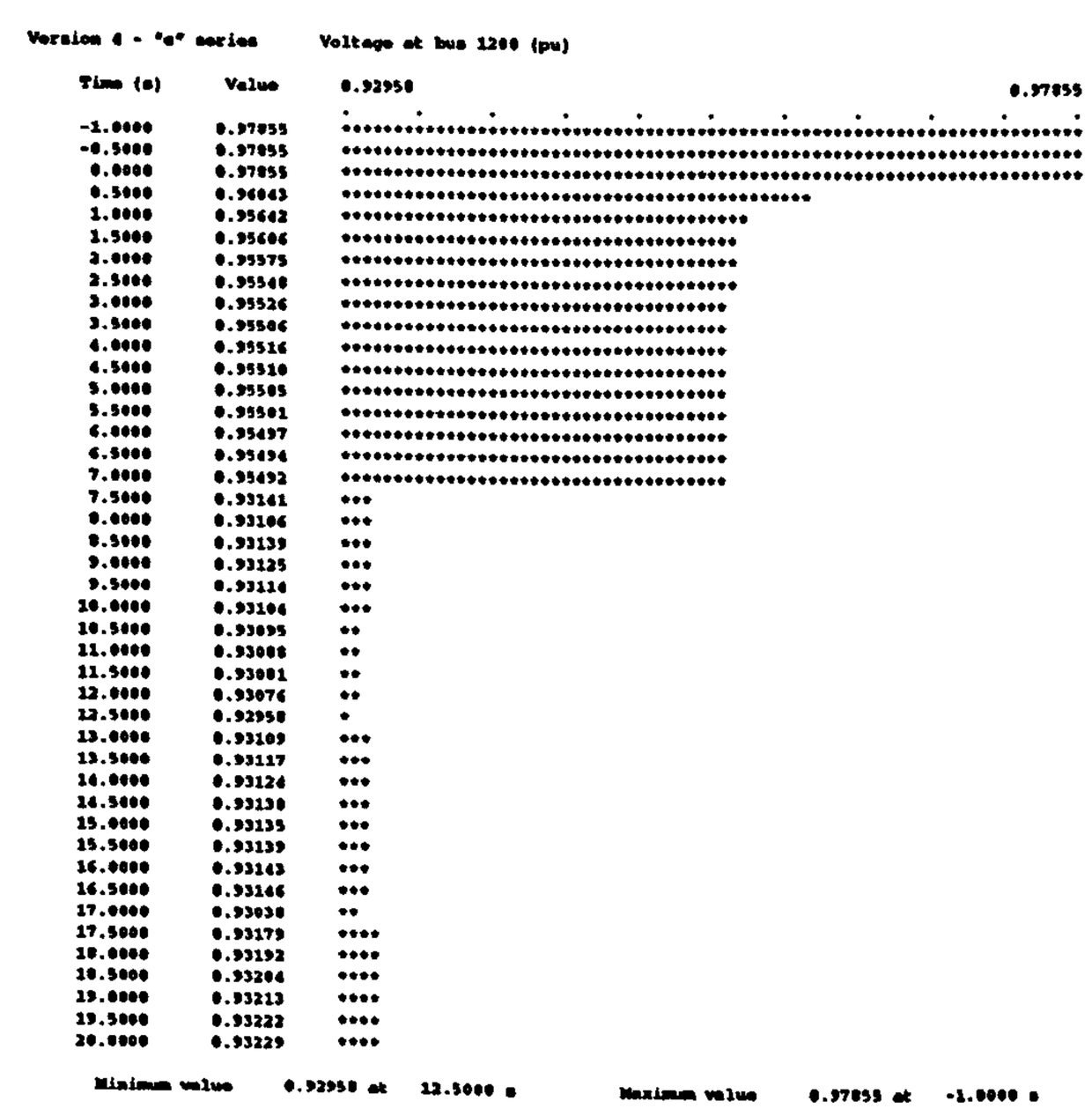


Figure 6.12 - Evolution of v_{1200}

due to the action of $OLTC_{1200\text{-}1400}$, which has its tap position changed at times 3.5, 8.0, 12.5 and 17.0s.

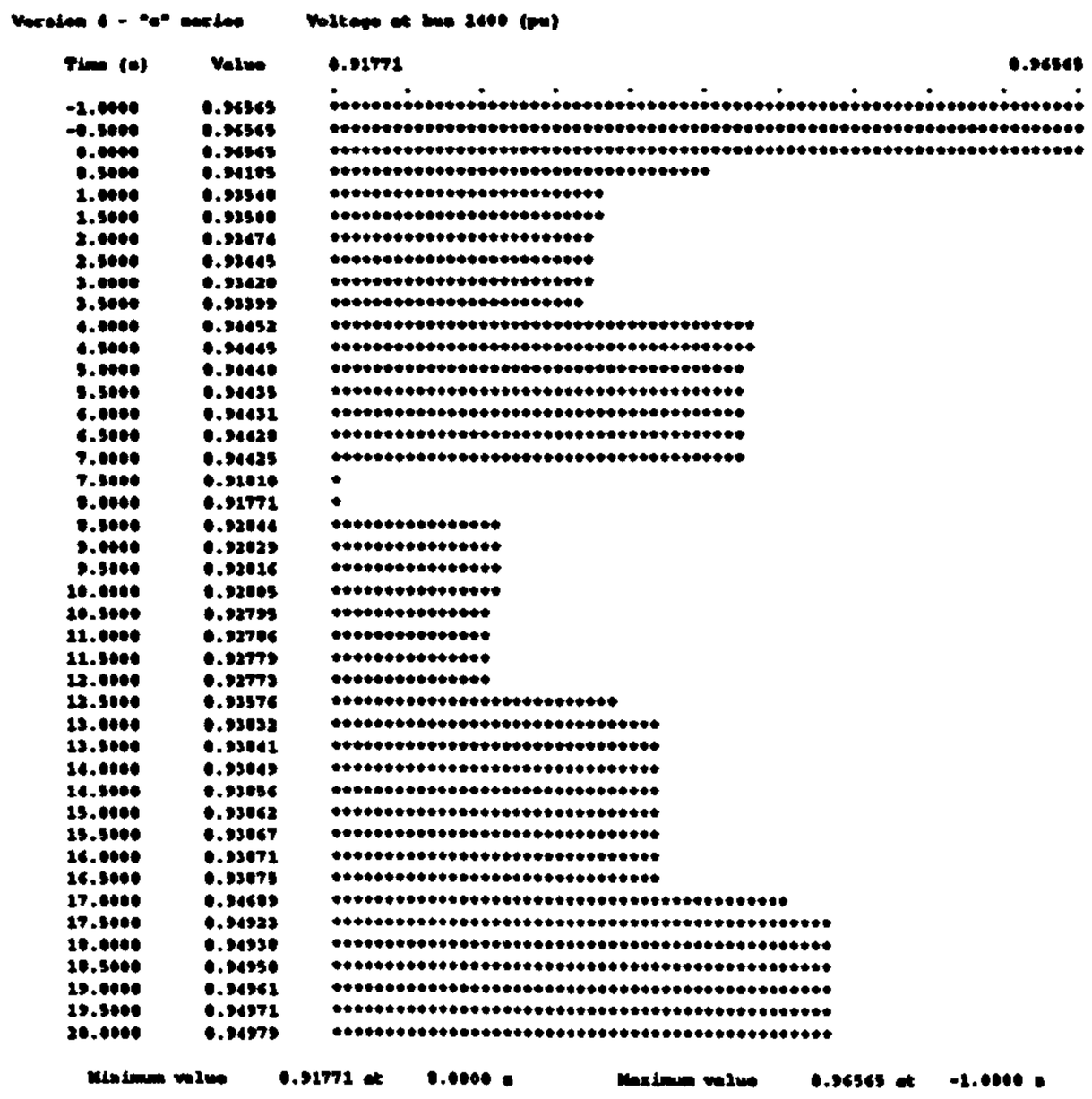


Figure 6.13 - Evolution of v_{1400}

Figures 6.14, 6.15 and 6.16 show the behaviour of slip, active power and reactive power respectively of the induction motor at bus 1400. From these figures, it can be seen that the slip and reactive power of the motor are much more sensitive to the voltage variation than the active power. The latter remains fairly constant due to the fact that the mechanical power is constant in the model of induction motors.

Figures 6.17, 6.18 and 6.19 show the behaviour of $OLTC_{1200\text{-}1400}$, the active power of dynamic load at bus 1100, and the active power of dynamic load at bus 1300 respectively. It should be noted that the curve of $OLTC_{1200\text{-}1400}$ is identical to the curve of $OLTC_{1200\text{-}1300}$; both OLTCs tried to restore the reference voltage at their corresponding controlled buses. The evolution of p_{1100} (dynamic load) clearly shows the exponential recovering behaviour of dynamic loads. The same can be observed in the case of the dynamic load at bus 1300, although more "transient subperiods" can be observed at bus 1300 due to the more irregular behaviour of voltage at this bus (because of $OLTC_{1200\text{-}1300}$). The evolution of voltage at buses 1100 and 1300, although not reproduced in this example, is similar to that of buses 1200 and 1400 respectively.

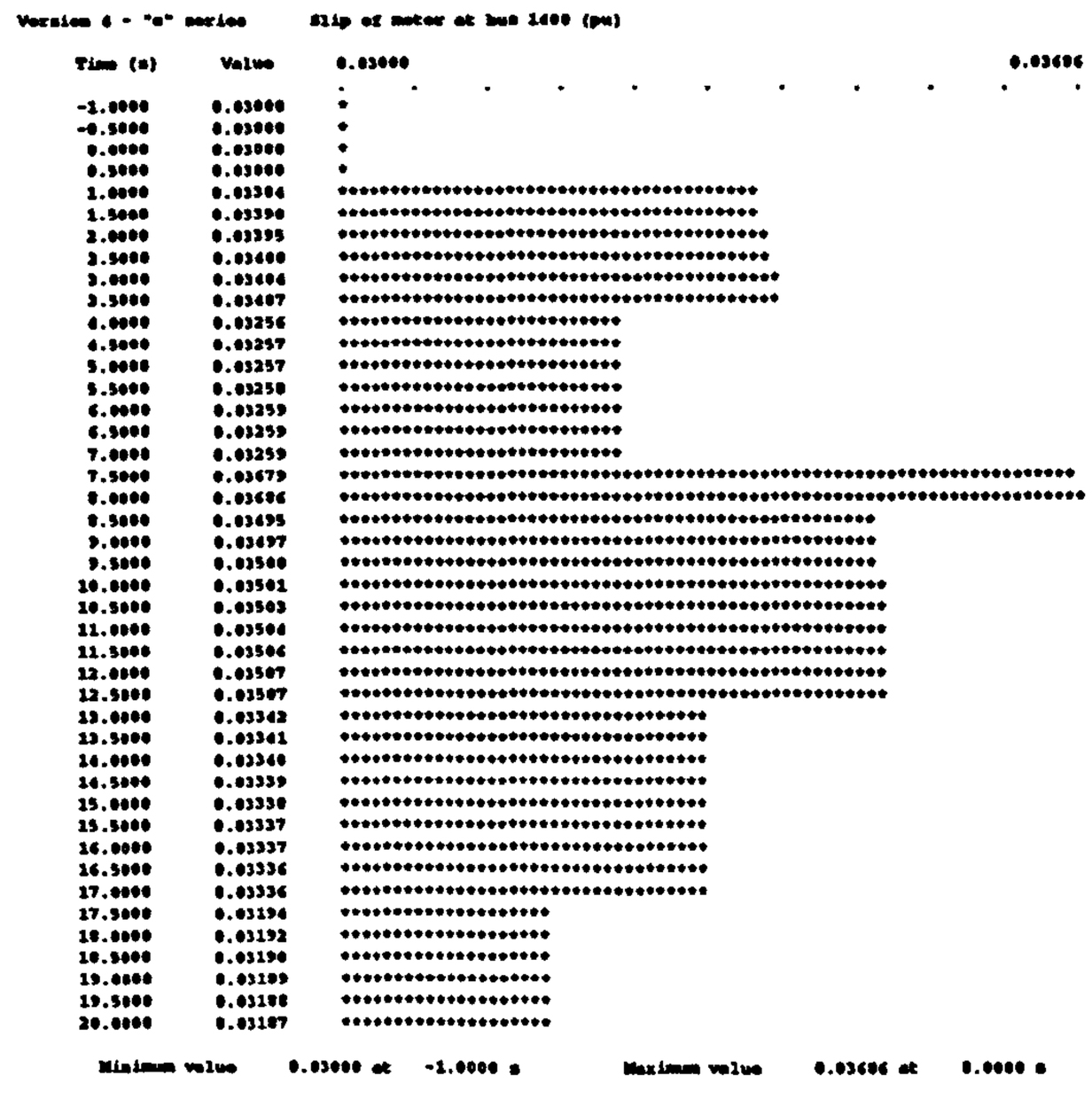


Figure 6.14 - Evolution of s_{1400}

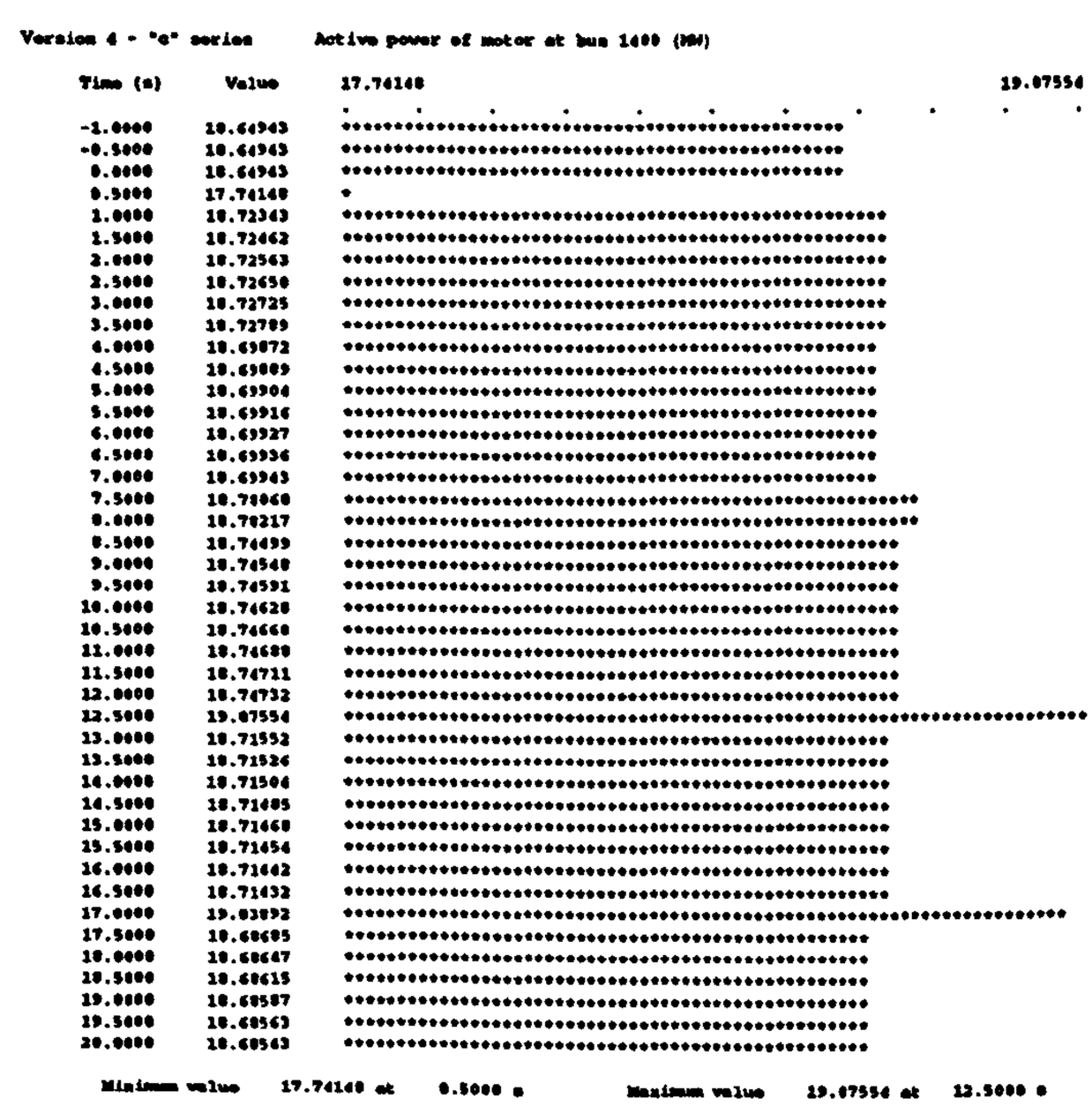


Figure 6.15 - Evolution of p_{1400}

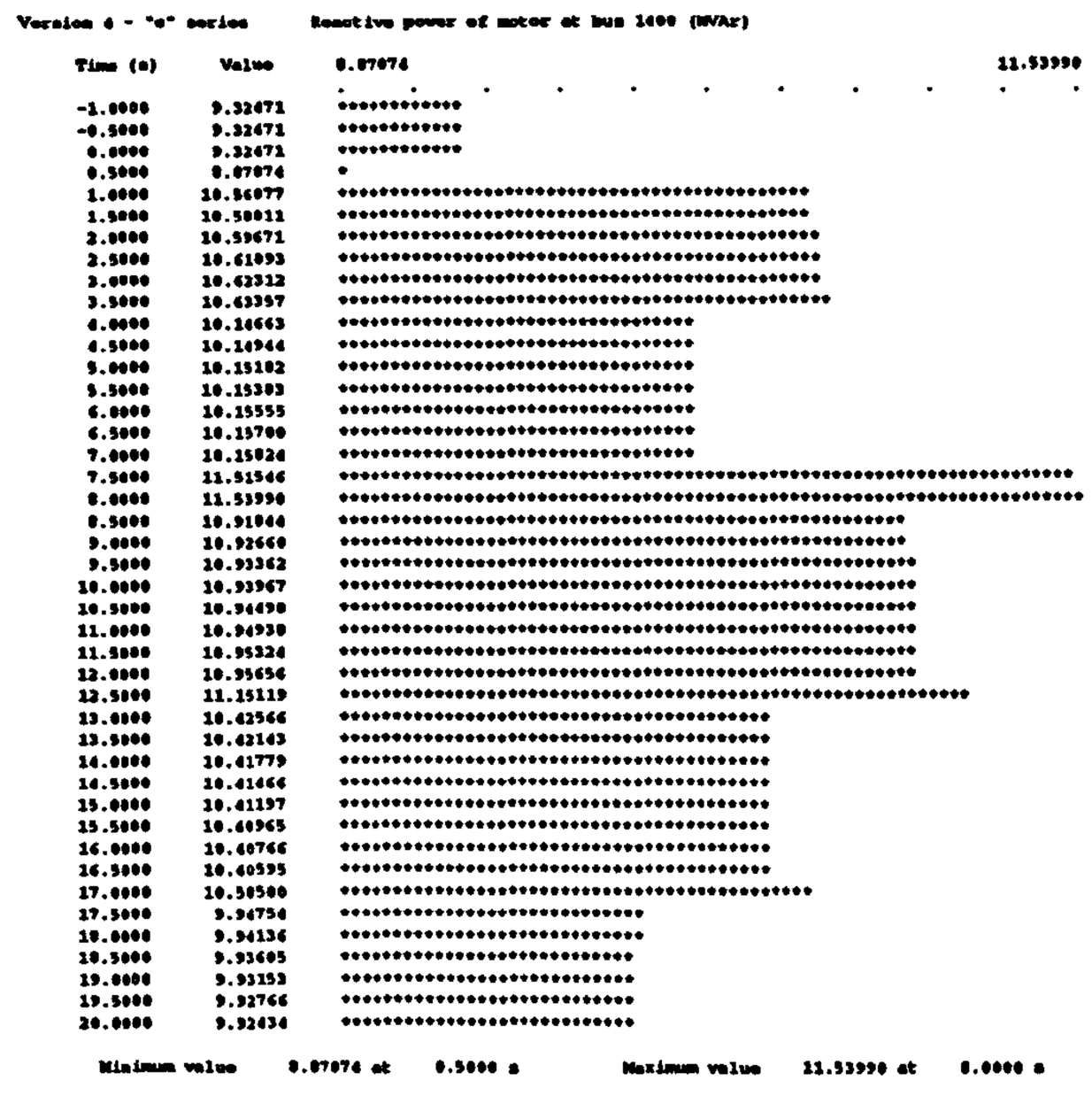


Figure 6.16 - Evolution of q_{1400}

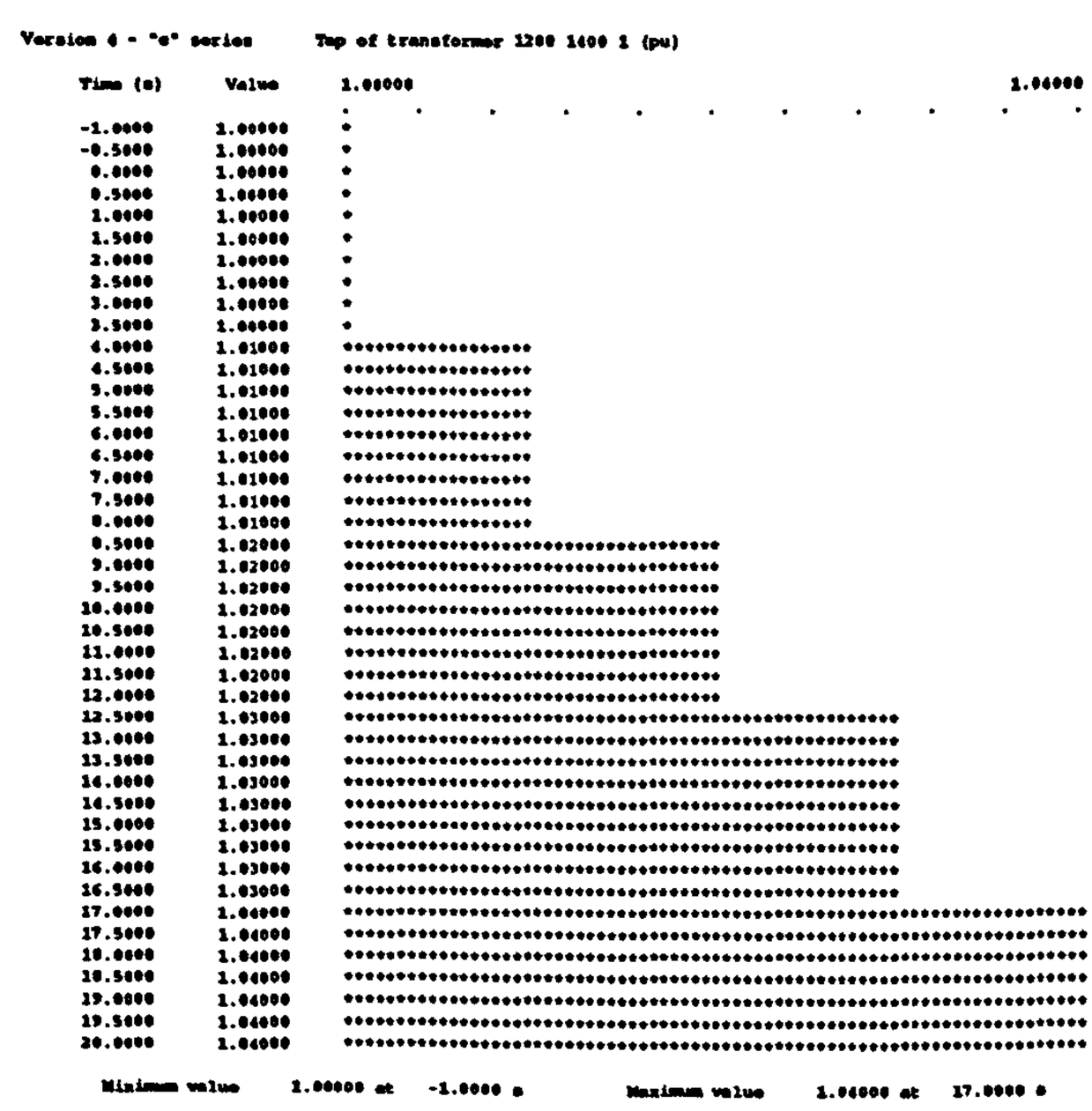


Figure 6.17 - Evolution of $OLTC_{1200\text{-}1400}$

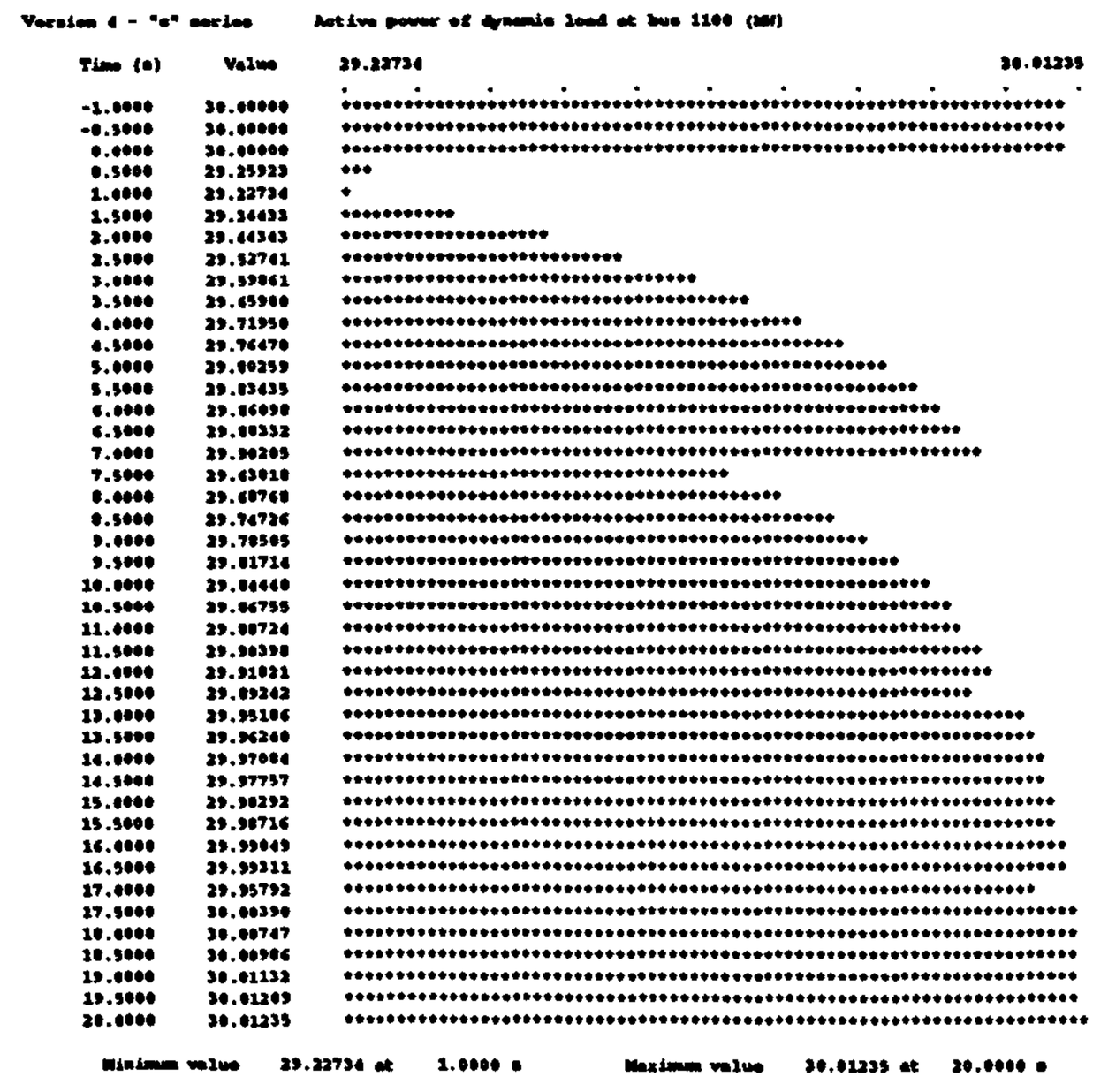


Figure 6.18 - Evolution of p_{1100} (dynamic load)

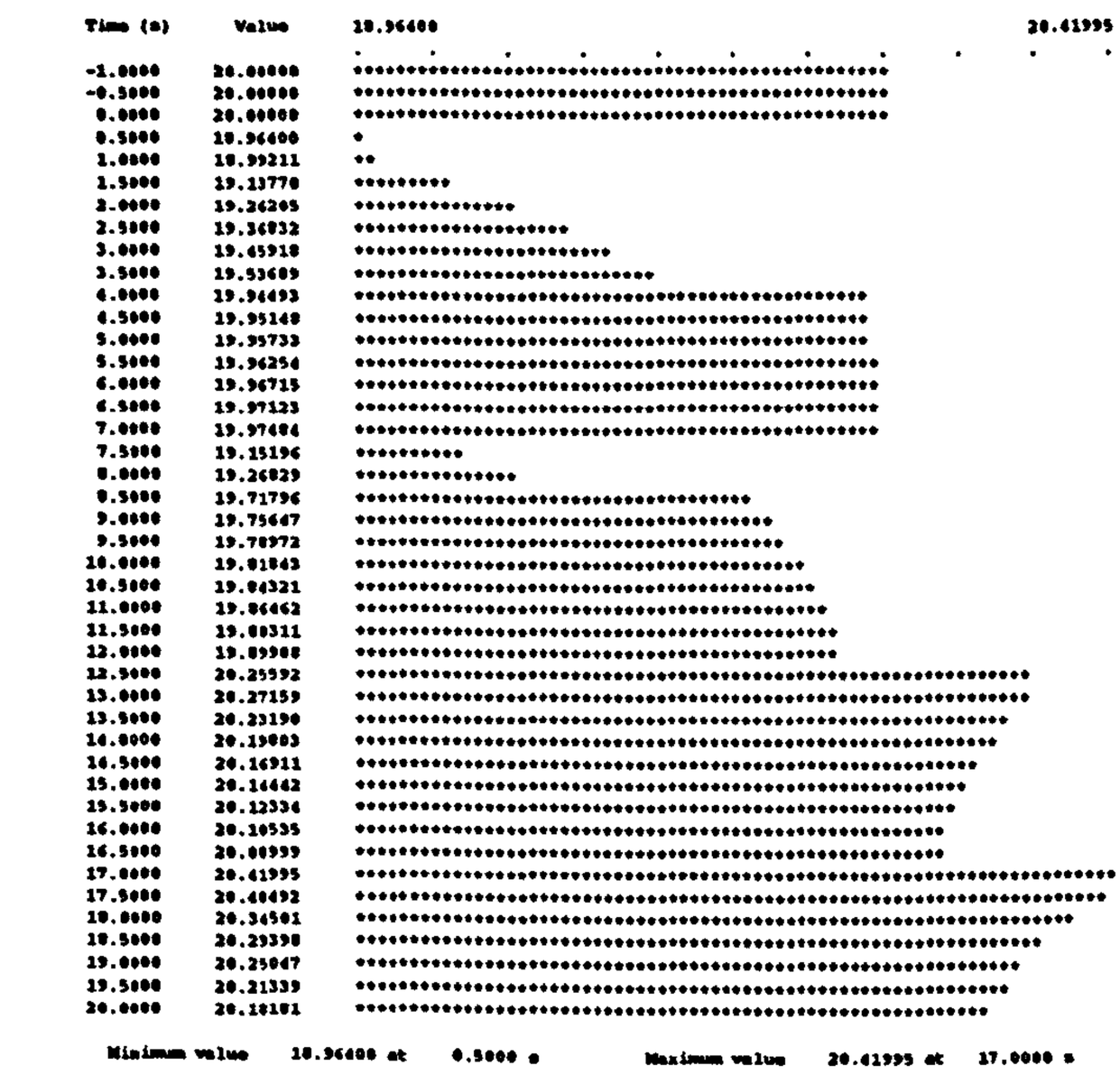


Figure 6.19 - Evolution of p_{1300} (dynamic load)

6.3.3.3 - Case 2

The second study case of dynamic simulation corresponds to the transient analysis specified in Table 6.13. This case is similar to the previous one; the only difference lies in an extra operation at

N° of operation	Time (s)	Operation specification
1	-1.0	Start of analysis
2	0.5	Capacitors at buses 1200, 1300 and 1400 disconnected
3	7.0	- Opening branch 1100-1200 (circuit #1) - New static load at bus 1400: (21.0 + *j*9.2)MVA
4	20.0	End of analysis

Table 6.13 - Transient operations list (Case 2)

time $t = 7.0$s, when the static component of the load at bus 1400 was increased from its rated value of (15.0 + *j*5.0)MVA to the value (21.0 + *j*9.2)MVA.

In this case, the power system did not withstand the disturbances, but the collapse occurred at $t = 12.5$, or 5.5s after the application of the second disturbance at $t = 7.0$s. Figures 6.20, 6.21 and 6.22 show the temporal evolution of σ_G, v_{100} and q_{100} respectively.

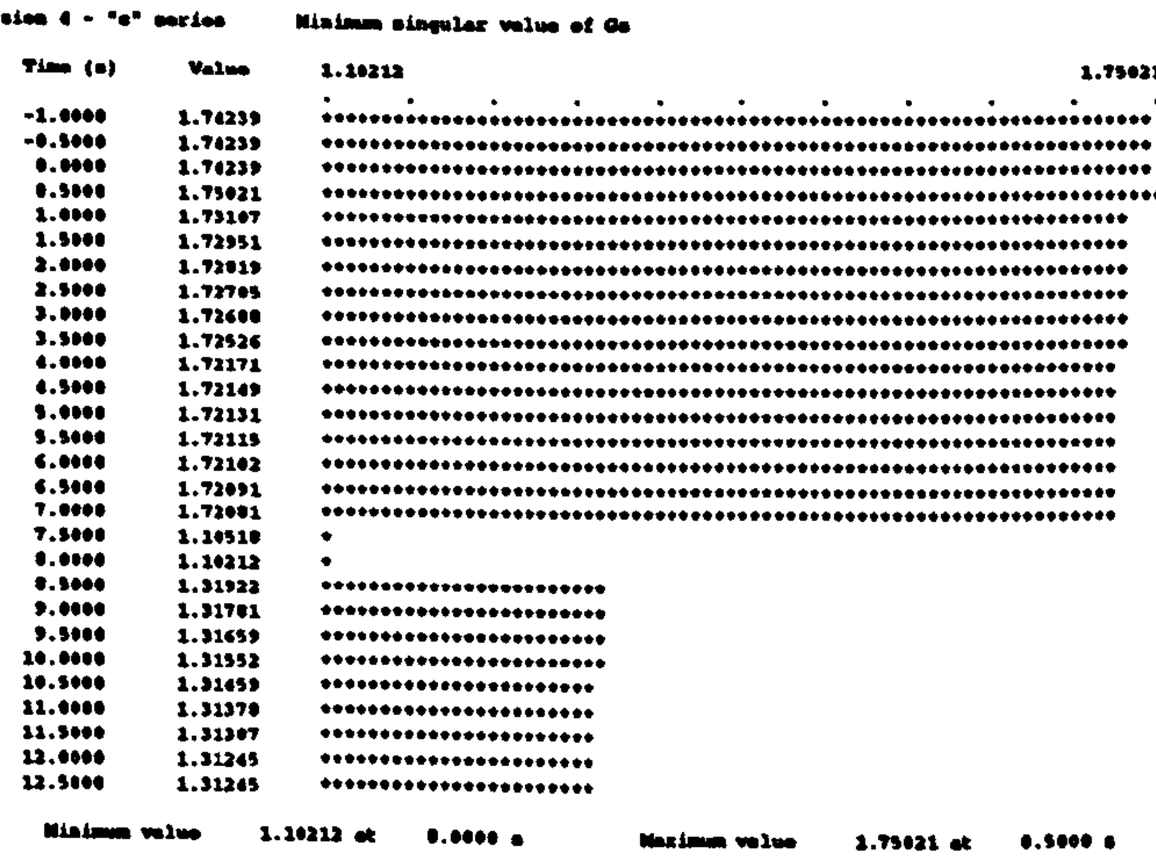


Figure 6.20 - Evolution of σ_G

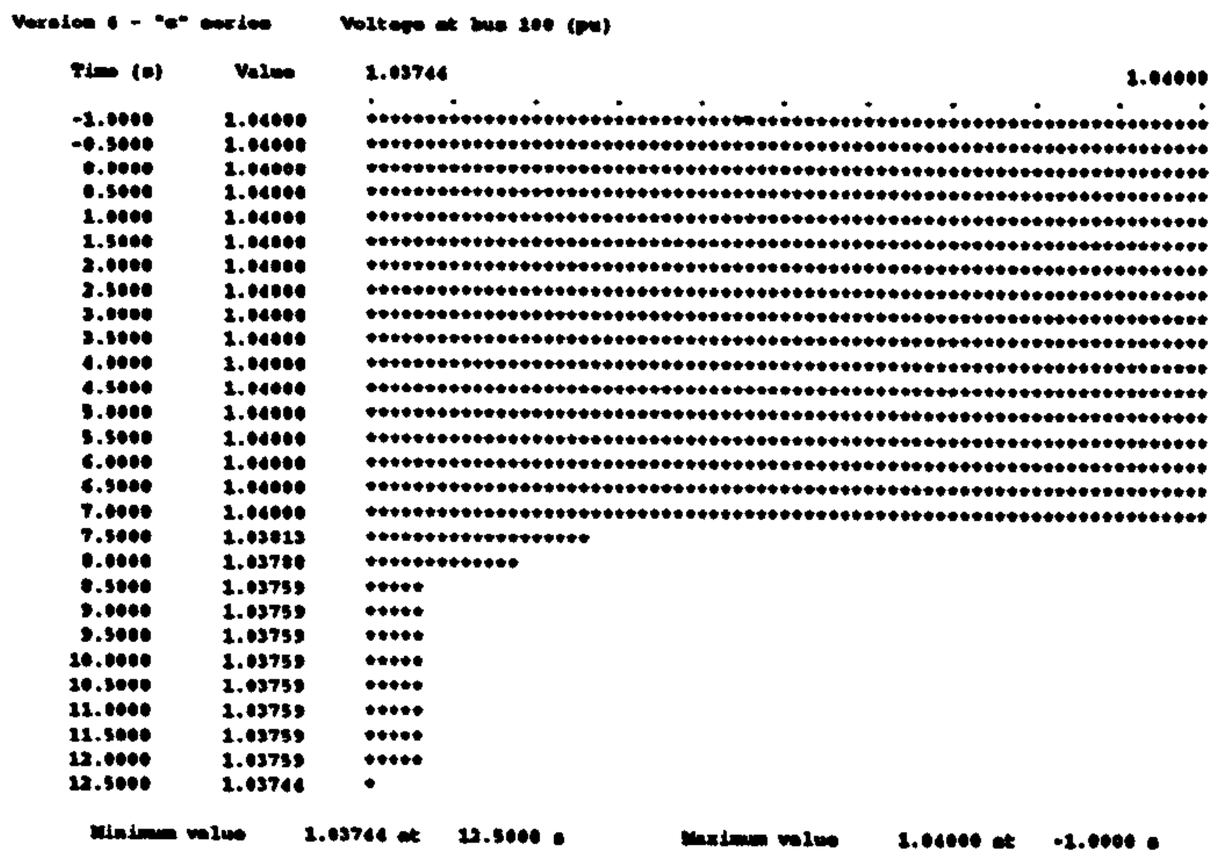


Figure 6.21 - Evolution of v_{100}

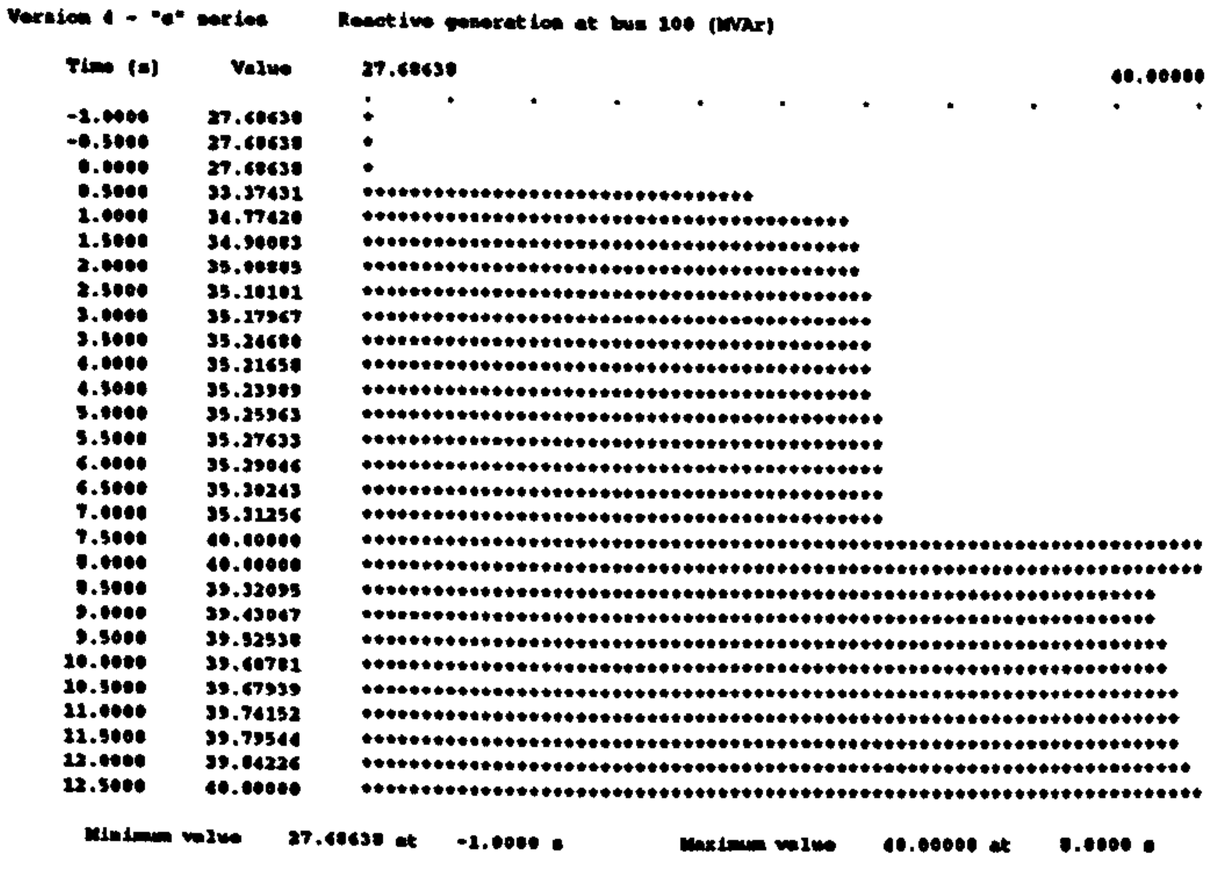


Figure 6.22 - Evolution of q_{100}

It is important to remember here that bus 100 corresponds to the terminal bus of a generator. From Figure 6.22, it can be seen that the upper reactive limit of 40MVAr was reached at $t = 7.5$s as a consequence of the major disturbance at $t = 7.0$s. The generator remained on its maximum reactive generation capability until some time between 8.0 and 8.5s. During this time, the voltage at bus 100 had to be reduced from its previous value of 1.04000pu in order to alleviate the reactive power burden on the generator. After $t = 8.5$s, the generator returned to a reactive power value below 40MVAr (although it was still close to this limit) and therefore the voltage at bus 100 could be maintained constant at its new value of 1.03759pu. This situation lasted until the last simulation time ($t = 12.5$s), when the generator reached its upper reactive power limit again (due to the action of the OLTCs, as will be seen later on) and the voltage at bus 100 had to be further decreased to the

value 1.03744pu.

It is interesting to notice that, during the time in which the generator operated at 40MVAr, the system's σ_G became a minimum. This is due to the fact that the size of the Jacobian matrices J and G_s increases by one row/column whenever a generator reaches its Q-limit (because of the new extra equation). As already seen in Chapter 5 (sub-section 5.4.3), when a matrix increases in size, its minimum singular value decreases.

Figures 6.23 and 6.24 show the voltage at bus 1400 and the slip of the induction motor at the same bus respectively. In Figure 6.23 the impact of the disturbances at t = 0.5s and t = 7.0s can

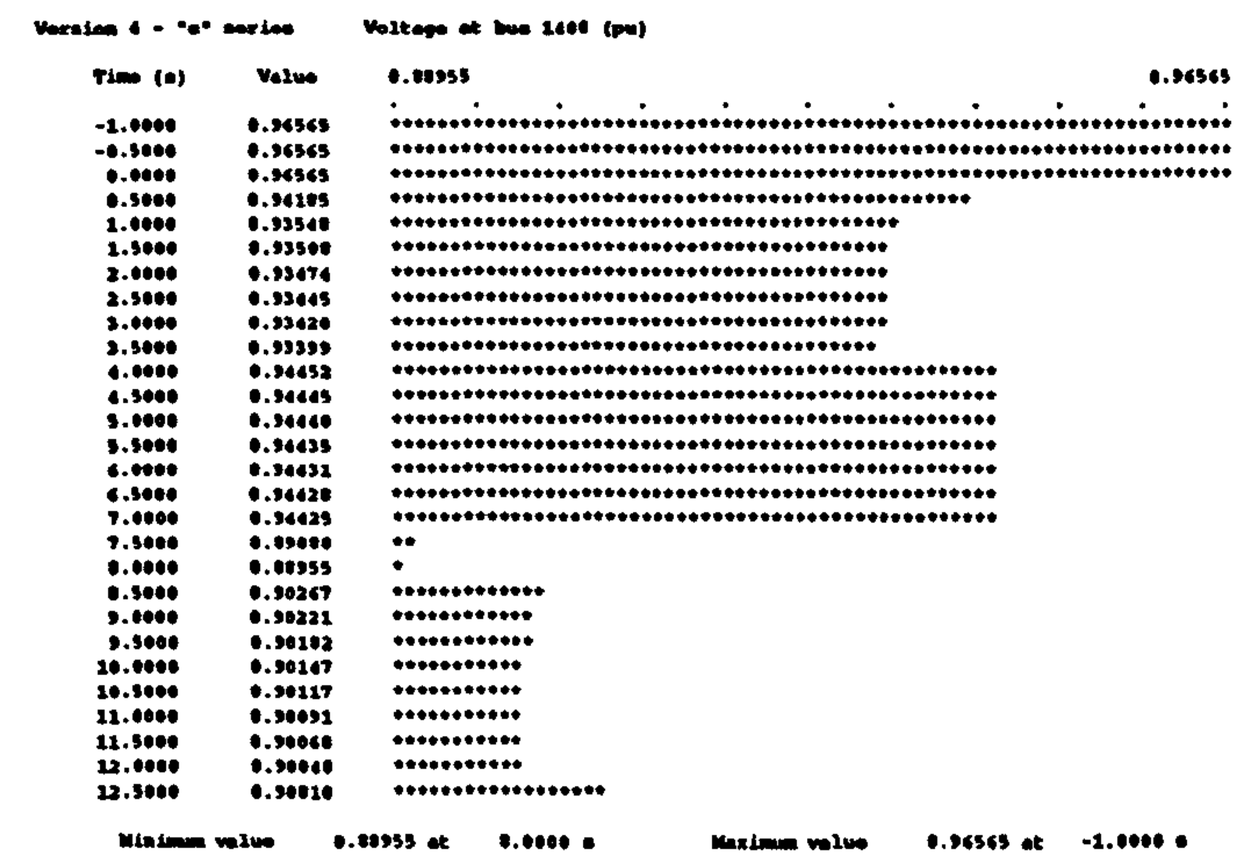


Figure 6.23 - Evolution of v_{1400}

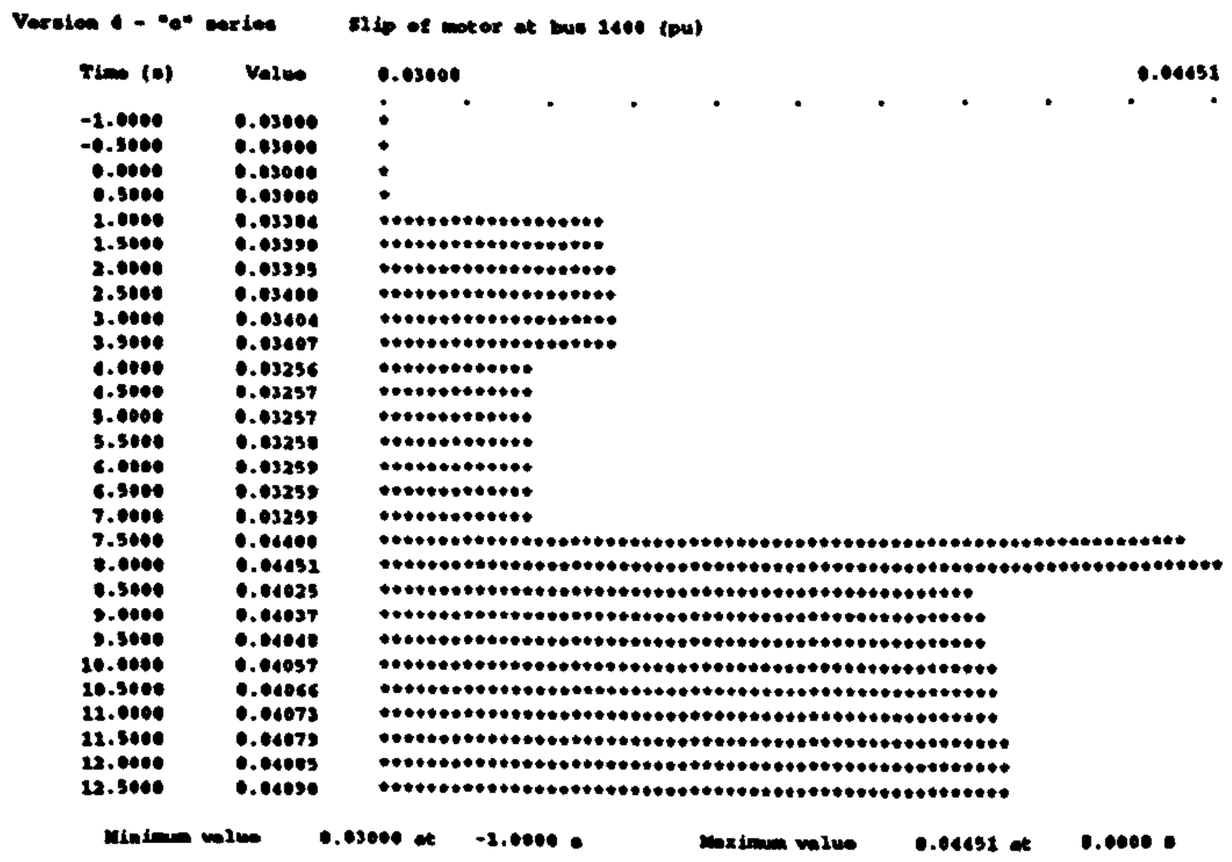


Figure 6.24 - Evolution of s_{1400}

clearly be seen. Also, it is possible to notice the voltage recovery that started at $t = 4.0$s, $t = 8.5$s and $t = 12.5$s due to the action of $OLTC_{1200\text{-}1400}$. Figure 6.24 shows that even with this voltage recovery, the slip of the induction motor at the same bus never came back to its initial value of 0.03pu.

Figure 6.25 shows the evolution of $OLTC_{1200\text{-}1400}$. This OLTC, as well as $OLTC_{1200\text{-}1300}$, completed one cycle at $t = 12.5$, and they tried to increase their tap positions in order to restore the set-point voltage at their controlled buses. The electrical system did not withstand this corrective action and the system eventually collapsed.

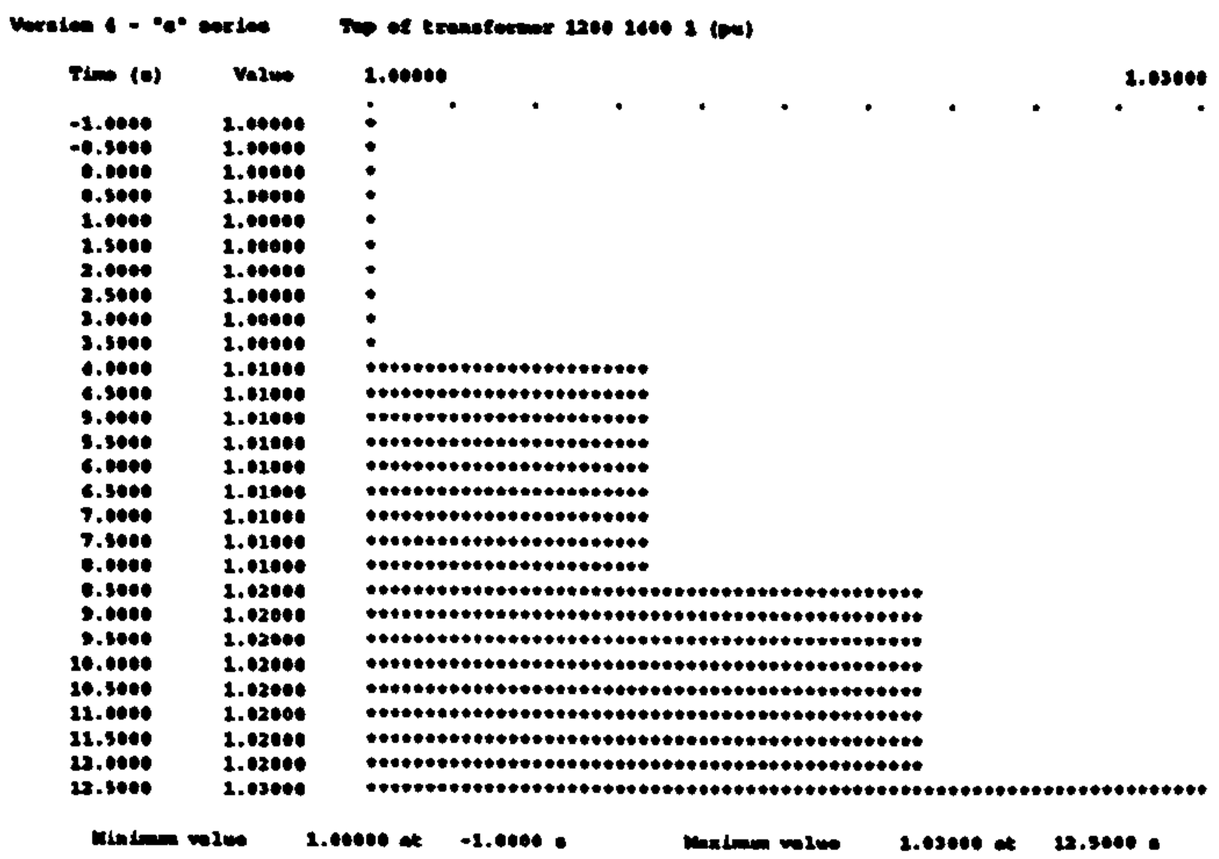


Figure 6.25 - Evolution of $OLTC_{1200\text{-}1400}$

Finally, Figure 6.26 shows how the dynamic load at bus 1300 contributed to the collapse of the system. This figure and the previous one hold the key to the collapse that occurred at $t = 12.5$s. After the major disturbances at $t = 0.5$s and $t = 7.0$s the load decreased drastically. The load controller tried to restore the load level to its set-point value, and did so in a smooth way. $OLTC_{1200\text{-}1300}$, which acted exactly in the same way as $OLTC_{1200\text{-}1400}$, also tried to restore the voltage at bus 1300 to its pre-disturbance level. But the dynamic load exhibits instantaneous constant-impedance behaviour, so when the tap was increased the power absorbed by the dynamic load increased in an abrupt way. The system could withstand these abrupt load increases at $t = 4.0$s and $t = 8.5$s, but could not do the same at $t = 12.5$s (recall the reactive power limit reached again by the generator 100 at $t = 12.5$s, Figure 6.22).

From this simple dynamic simulation, it can be seen that the collapse of the system occurred due to a reactive generation deficiency combined with automatic OLTC action and the instantaneous response of the load.

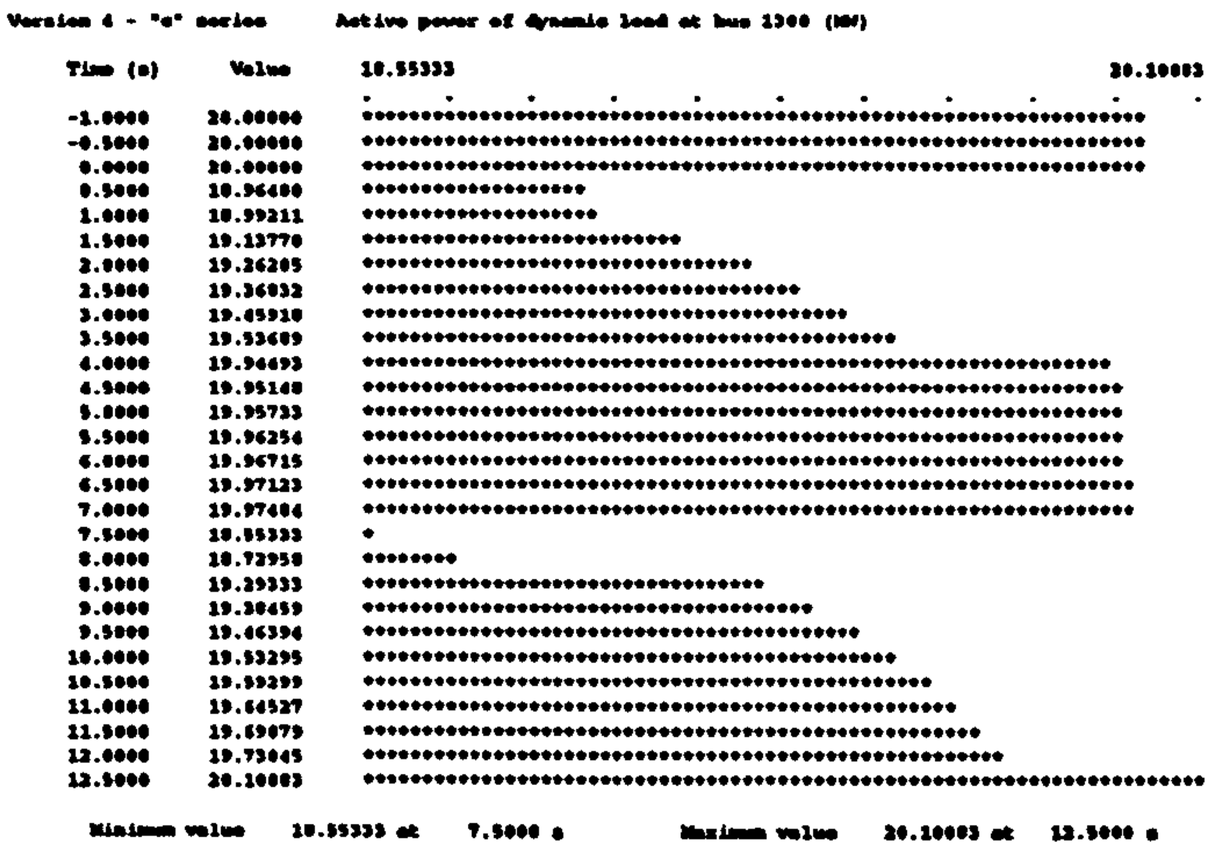


Figure 6.26 - Evolution of p_{1300} (dynamic load)

It should be pointed out that the total CPU time for both cases was 20.59 and 21.05s respectively, using the same personal computer as described in Table 6.7. Although the simulated time in the second case is nearly half of the time in the first case, the greater number of load-flow iterations required in the second case (due to the more stringent conditions) almost doubled the simulation cost.

6.3.4 - Interface programs (INTF2 and INTF3)

As mentioned earlier in this chapter, programs INTF2 and INTF3 constitute the interface between the dynamic simulation and the neural network programs. Their main purpose is to generate complete sets of training vectors which contain information about the dynamic behaviour of the electric system. Figure 6.27 shows a block diagram describing the main tasks in the generation of training vectors with either programs INTF2 or INTF3.

Referring to Figure 6.27, the definition of the transient analysis means making an input vector available for use by the program, and this can be done manually or automatically. In manual generation of training vectors, the user must provide all of the data in the input vectors. On the other hand, in automatic generation the program creates all input vectors using the Monte Carlo method. The occurrence of short-circuits and branch operations (opening or closing) during the transient analysis is controlled by associated probabilities given by the user. The actual values of load, generation and reactors are randomly selected using a distribution model and adequate limiting parameters. Both the distribution model (Gaussian or uniform) and the limiting parameters are also given by the user.

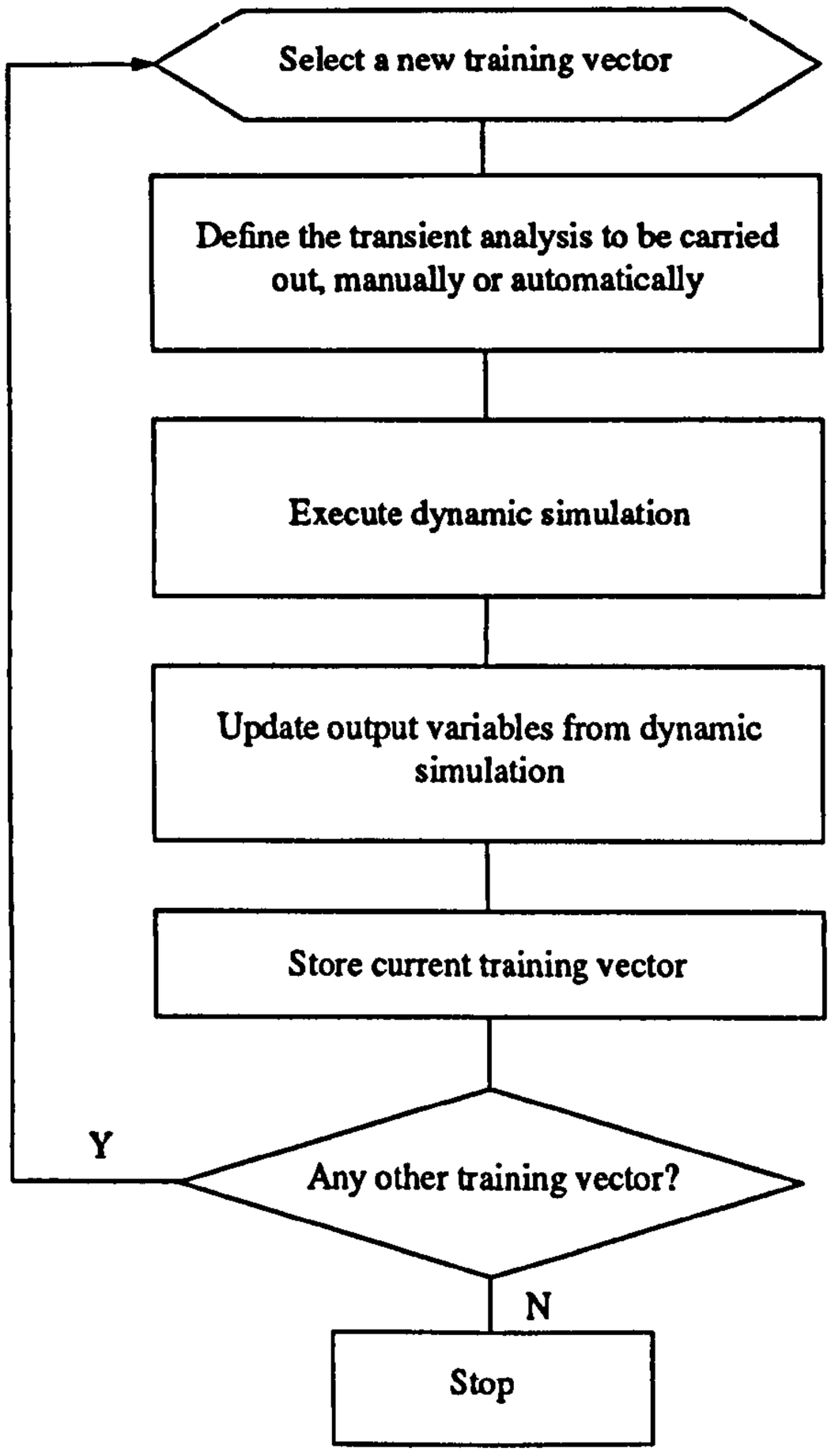


Figure 6.27 - Training vector generation in programs INTF2 and INTF3

Once an input vector has been defined (either manually or automatically), the dynamic simulation subroutines compute the required output variables. The outcome of the dynamic simulation will then say whether or not the system was able to withstand the imposed disturbances. A complete training vector, containing input and output variables, is then stored and the programs start the generation of the next vector.

After the whole training set has been generated, both programs execute a scaling process on the training data, in order to map the input data onto a convenient range for the MLP and also to avoid the "dead zone" of the sigmoid function near to the values 0 and 1. This scaling process is similar to that executed by the interface program INTF1 (see Chapter 5).

In the next sections, programs INTF2 and INTF3 will be described in greater detail. Both programs will be used to generate training and testing sets containing hundreds of vectors, each vector corresponding to a particular disturbance (which in turn generates one dynamic simulation scenario). These sets will be used for training and testing MLP networks, which will be used in the prediction of the future behaviour of the power system. The results obtained will then be presented and discussed.

6.4 - Program INTF2

6.4.1 - Introduction

In this section, program INTF2 will be presented. It implements the first methodology for mapping the results from dynamic simulations onto the MLP domain. The most relevant aspects of this methodology will be described in some detail. The application of the methodology in the prediction of the future behaviour of the system will be described through a series of study cases. The results obtained in these cases will then be discussed.

6.4.2 - Methodology

In program INTF2, the dynamic simulations to be executed are specified from *elementary transitions*. It is assumed that the power system was operating in steady-state until a *transition time*, when changes in system parameters occur (the transition time is assumed to be $t_{trans} = 0$ always). After the transition time, a dynamic simulation is run in order to assess the stability of the system.

In this case, input variables for the neural network reflect the value of system parameters before and after the transition. Therefore, a complete input vector will have as many entries as twice the number of input variables, in order to contain the "before-and-after" information about the transition. Table 6.14 shows the input variables considered in program INTF2, and Figure 6.28 presents the structure of the input vector.

Code	Description
0	Short-circuit at a bus

Table 6.14 - Input variables in program INTF2

Code	Description
1	State of a branch
2	Global load
3	Global generation
4	Bus load
5	Bus generation
6	Bus reactor

Table 6.14 - Input variables in program INTF2

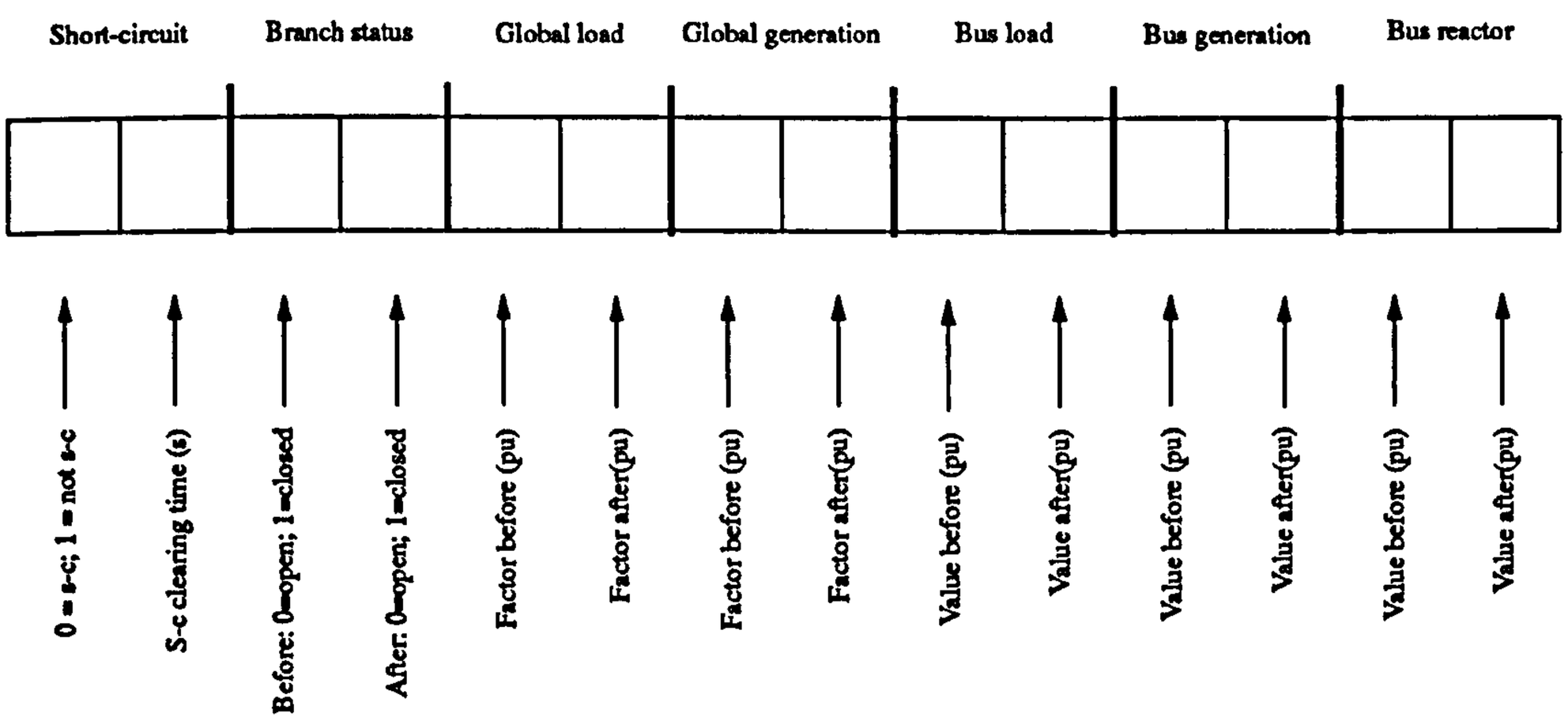


Figure 6.28 - Structure of the input vector in program INTF2

Table 6.15 presents the output variables defined in program INTF2. In this case, a single entry in the output vector is associated with each output variable. It can be seen from this table that in most cases an element of the output vector will contain the final value of the corresponding output variable (at the end of the simulation). Output variable 0 corresponds to the various simulation outcomes that can occur. Table 6.16 shows the codes for these outcomes.

In this methodology, the neural network is taught with a comprehensive training set which contains associations between elementary transitions and simulation outcomes. When in processing mode, the neural network will be presented with a sequence of transitions which can be generated, for instance, from the periodic sampling of relevant variables in the power system. It is

Code	Description
0	Simulation outcome (SO)
1	Final simulation time (FST)
2	Minimum singular value of G_s (fv)
3	Minimum singular value of J (fv)
4	Angular speed of generator (fv)
5	Angle of generator (fv)
6	Angular difference between generators (fv)
7	Mechanical power of generator (fv)
8	Bus voltage (fv)
9	Bus angle (fv)
10	Bus active load (fv)
11	Bus reactive load (fv)
12	Bus active generation (fv)
13	Bus reactive generation (fv)
14	Bus reactor (fv)
15	Motor slip (fv)
16	Motor active power (fv)
17	Motor reactive power (fv)
18	OLTC (fv)
19	Dynamic load (active power, fv)
20	Dynamic load (reactive power, fv)

Table 6.15 - Output variables in programs INTF2 and INTF3 ("fv" means "final value")

Code	Reason
0	Normal termination
1	Non-convergent load-flow (initial conditions)
2	Singular Jacobian matrix (initial conditions)
3	Non-convergent load-flow (during simulation)
4	Singular Jacobian matrix (during simulation)
5	Non-convergent MEM (generators)
6	Non-convergent MEM (motors)

Table 6.16 - Codes for simulation outcomes in programs INTF2 and INTF3

Code	Reason
7	Non-convergent MEM (generators and motors)

Table 6.16 - Codes for simulation outcomes in programs INTF2 and INTF3

expected that the neural network then be capable of recognising in advance both stable and potentially dangerous situations.

6.4.3 - Results

In this sub-section, results obtained with the methodology implemented in program INTF2 (i.e., elementary transitions) will be presented. The main output variables considered here are the simulation outcome (SO, see Table 6.16) and the final simulation time (FST, Table 6.15). The results are organised in a series of three different study cases (series B). In all cases, labelled B1, B2 and B3, the electrical system of Figure 6.10 will be used (see sub-section 6.3.3). Only the most relevant data regarding training/testing sets and MLP architecture will be presented here; the complete set of data is contained in the Appendix.

Table 6.17 defines the codes that identify training/testing sets and MLP architecture in series B. These codes are explained in Tables 6.18 and 6.19.

Case	Code for training/ testing set files	Code for MLP architecture
B1	TTB1	NNB1
B2	TTB1	NNB2
B3	TTB3	NNB3

Table 6.17 - Specification of training/testing set files and MLP architecture in series B

Table 6.20 presents the first set of results for Case B1. These results correspond to the average value, standard deviation and maximum value for the MLP evaluation error obtained in this case. It should be noted that the evaluation error is defined as in Chapter 5 (Eq. (5.5)); this definition will also be used throughout this chapter.

Code	N° of training/ testing vectors	Input variables	Output variables	Max. duration of simulation (s)
TTB1	500/500	- State of branch 1000 1100 #1 - State of branch 1100 1200 #1 - Global load - Load at bus 1100 - Load at bus 1300 - Load at bus 1400 - Reactor at bus 1200 - Reactor at bus 1300 - Reactor at bus 1400	- SO - FST	300.
TTB3			- SO - FST - Voltage at bus 1100 - Voltage at bus 1200 - Voltage at bus 1300 - Voltage at bus 1400 - Slip of motor 1100 - Slip of motor 1400	

Table 6.18 - Main data for training/testing set files in series B

Code	MLP structure (N° of neurons in each layer)	Training sessions (N° of iterations/learning rate)				
		1	2	3	4	5
NNB1	18/20/10/2	1000/1.0	1000/0.9	1000/0.8	1000/0.7	1000/0.6
NNB2	18/10/5/2					
NNB3	18/30/20/8					

Table 6.19 - Structure and training data for MLPs in series B

Parameter	Training set		Testing set	
	Output variable		Output variable	
	SO	FST	SO	FST
Average error (%)	0.75	1.03	14.26	30.79
Std. dev. of error (%)	0.93	0.68	23.00	40.70
Maximum error (%)	6.34	5.90	160.99	142.37

Table 6.20 - First set of results - case B1

It can be seen that the neural network correctly learned the information contained in the training set, but this was not useful at all when the testing set was presented to the MLP. This situation arises fairly often when setting up MLPs, and it suggests that the training/testing sets and the MLP architecture were not properly matched, either because the training/testing set did not repre-

sent the problem at hand or because there were too many hidden-layer neurons, so the neural network just memorised the training set without extracting the relevant features from it.

In an attempt to overcome this problem, in case B2 the structure of the MLP was modified so as to contain less hidden-layer neurons with respect to case B1 (20 and 10 neurons in layers 1 and 2 for case B1 compared to 10 and 5 neurons for case B2). The training/testing sets in this case are the same as in case B1. Table 6.21 shows the first set of results for case B2.

Parameter	Training set		Testing set	
	Output variable		Output variable	
	SO	FST	SO	FST
Average error (%)	7.31	25.71	14.93	38.71
Std. dev. of error (%)	8.92	14.03	18.97	29.66
Maximum error (%)	69.70	95.04	124.54	172.70

Table 6.21 - First set of results - case B2

From Table 6.21, it can be seen that the evaluation error for the training set increased substantially with respect to case B1, unlike the evaluation error for the testing set, which did not change appreciably.

Case B3 represents an attempt to solve the learning problems that emerged in cases B1 and B2. The training/testing sets now include other output variables (bus voltages) which normally exhibit a less abrupt behaviour than the SO and the FST. Also, the size of the MLP was increased so as to facilitate the acquisition of this extra knowledge by the neural network. Table 6.22 shows the first set of results for case B3.

From Table 6.22, it can be seen that once more the neural network correctly learned the information from the training set (where the evaluation error is satisfactorily low), but the same did not occur with the testing set, for which the evaluation error reached maximum values in the order of 100%. Table 6.23 shows the distribution of error in cases B1, B2 and B3. It should be noted that this distribution was computed using the corresponding testing sets only.

6.4.4 - Discussion of Series B cases

Parameter -->			Average error (%)	Std. dev. of error (%)	Maximum error (%)
Training set	Output variable	SO	0.80	0.87	4.76
		FST	0.73	0.86	5.47
		V_{1100}	1.85	1.61	16.25
		V_{1200}	1.96	1.95	17.20
		V_{1300}	1.22	1.36	11.85
		V_{1400}	1.41	1.83	18.88
		$Slip_{1100}$	1.44	1.14	5.89
		$Slip_{1400}$	0.54	0.76	10.75
Testing set	Output variable	SO	12.53	20.00	103.77
		FST	27.23	39.80	122.47
		V_{1100}	6.22	11.77	98.12
		V_{1200}	6.94	11.37	89.05
		V_{1300}	4.79	9.93	101.12
		V_{1400}	5.61	10.04	78.20
		$Slip_{1100}$	1.93	2.43	38.12
		$Slip_{1400}$	3.78	13.83	108.24

Table 6.22 - First set of results - case B3

Class	Error limits (%)		Case B1		Case B2		Case B3							
	Min.	Max.	SO	FST	SO	FST	SO	FST	V_{1100}	V_{1200}	V_{1300}	V_{1400}	S_{1100}	S_{1400}
1	0	2	47.03	44.92	2.97	0.42	51.46	46.46	47.29	40.21	60.21	56.88	69.58	88.33
2	2	4	7.63	5.93	48.94	1.06	7.71	6.25	24.38	22.71	16.67	15.00	20.42	3.13
3	4	6	4.87	2.75	3.81	2.33	2.92	3.33	7.50	12.08	5.63	6.46	6.46	1.46
4	6	8	2.54	1.06	1.91	1.48	3.54	3.13	2.71	2.71	3.75	2.92	2.08	0.63
5	8	10	2.54	3.18	2.12	1.06	1.46	3.33	2.08	3.33	1.46	2.29	0.21	0.42
6	10	12	2.97	2.12	4.45	1.27	1.88	3.33	1.46	1.25	0.63	2.08	0.42	0.42
7	12	14	1.06	0.64	6.14	1.06	1.25	1.04	1.67	2.71	0.63	1.04	0.63	0.21
8	14	16	2.97	1.06	1.91	1.06	2.92	0.83	1.25	2.71	1.46	1.67	0.00	0.21
9	16	18	3.60	1.06	1.06	2.12	5.42	0.63	1.88	0.83	1.46	1.67	0.00	0.42

Table 6.23 - Distribution of error in cases B1, B2 and B3 (testing sets only) (% of total number of testing vectors)

Class	Error limits (%)		Case B1		Case B2		Case B3							
	Min.	Max.	SO	FST	SO	FST	SO	FST	V_{1100}	V_{1200}	V_{1300}	V_{1400}	S_{1100}	S_{1400}
10	18	20	0.85	0.42	2.12	1.27	0.63	0.63	1.46	1.04	1.04	1.04	0.00	0.00
11	20	22	2.75	0.64	1.69	0.21	0.63	0.63	0.42	1.25	1.25	0.83	0.00	0.21
12	22	24	0.42	0.85	1.27	40.04	1.25	0.63	0.63	1.04	0.83	1.67	0.00	0.00
13	24	26	1.06	0.85	0.42	8.47	1.04	0.00	0.83	1.88	0.83	0.42	0.00	0.00
14	26	28	1.06	0.64	0.42	2.12	0.63	0.00	0.42	0.63	1.46	2.08	0.00	0.00
15	28	30+	18.65	33.88	20.77	36.03	17.26	29.78	6.02	5.62	2.69	3.95	0.20	4.56
Total			100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 6.23 - Distribution of error in cases B1, B2 and B3 (testing sets only) (% of total number of testing vectors)

The results obtained in series B showed that when the neural network learns the knowledge contained in the training set, this knowledge could not be used for estimating the value of the output variables in the testing set. This problem can be caused in a number of ways, namely:

(*i*) too large a neural network (with too many hidden-layer neurons);

(*ii*) improper mapping of the dynamic simulation onto the neural network problem language;

(*iii*) inadequate size of training/testing sets, with not enough vectors so as to properly represent the electrical problem.

In case B2 an attempt was made so as to correct the problem in view of cause (*i*) above (MLP too large). The evaluation error using the testing set did not significantly change with respect to case B1, while the evaluation error using the training set did increase. This points to the fact that cause (*i*) may not be the main reason for this learning problem.

In order to deal with cause (*ii*) (improper mapping), a new methodology for constructing training/testing sets was developed. This methodology seeks to transfer more information on the early stages of the dynamic simulation to the MLP. This approach was implemented as a new program (INTF3) and it will be presented in the following section, together with the analysis of cause (*iii*) above.

6.5 - Program INTF3

6.5.1 - Introduction

In this section, program INTF3 will be described. It implements the second methodology for mapping the results from dynamic simulations onto the MLP domain. The most important aspects of this methodology will be presented in some detail. The application of this methodology in the prediction of the future behaviour of the system will be described through two series of study cases. The results obtained in these cases will then be discussed.

6.5.2 - Methodology

Program INTF3 is, in many respects, similar to program INTF2. The main difference lies on the type of variables that control the dynamic simulation and make up the training vectors. Table 6.24 shows the types of variables defined in program INTF3.

Type of variable	Purpose
Control	To control the dynamic simulation
Input	To form the first part of a training vector
Output	To form the second part of a training vector

Table 6.24 - Types of variables in program INTF3

Control variables allow the user to specify what kind of events will appear in the operations list of the dynamic simulation. They are similar to the input variables, but because program INTF3 allows the definition of many new input variables, which are not directly related to the specification of the operations list, a clear distinction between control and input variables was needed. Table 6.25 shows the control variables recognised by program INTF3.

Code	Description
0	Short-circuit at a bus (occurrence/clearing)
1	Branch status (opening/closing)
2	Global load (multiplication factors)

Table 6.25 - Control variables in program INTF3

Code	Description
3	Global generation (multiplication factors)
4	Bus load (new values)
5	Bus generation (new values)
6	Bus reactor (new values)

Table 6.25 - Control variables in program INTF3

Input variables are associated with the new idea of *sampling* relevant information from a dynamic simulation scenario. A number of *sampling times* is specified in advance, and this number is kept fixed throughout the creation of the training set. Figure 6.29 shows an example illustrating the concept of sampling input variables. In this case, the number of sampling times is 4, the sampling interval is 5s, and the simulation is 60-second long.

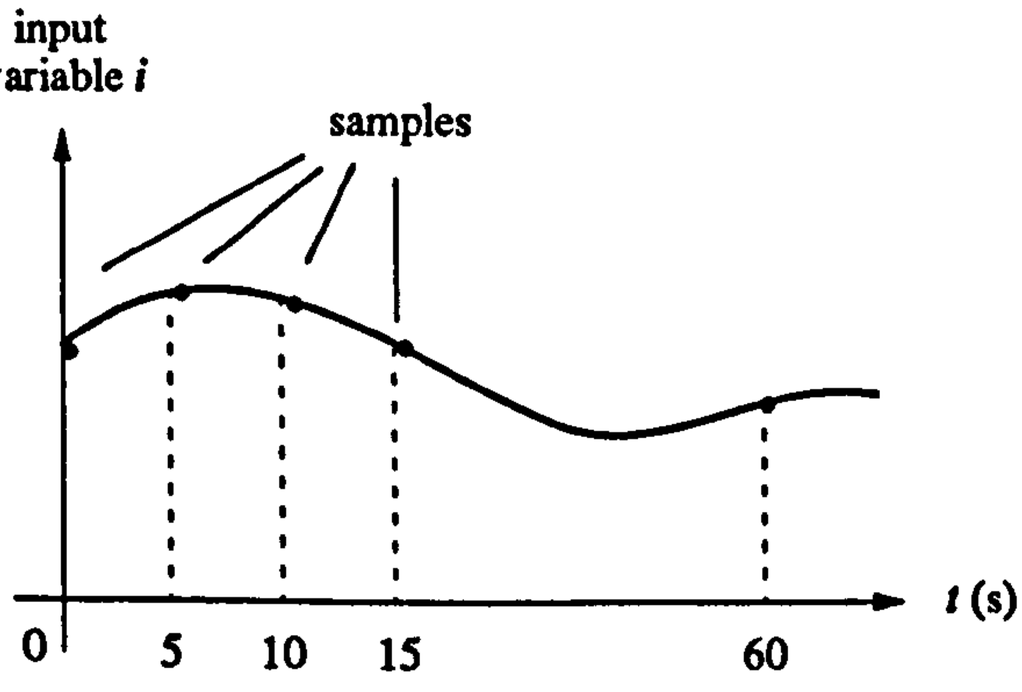


Figure 6.29 - Sampling input variables in program INTF3

All sampled values of any input variable (at times 0, 5, 10 and 15s) will be transferred to the input vector being generated if the user selects that input variable. Thus, the total number of input variables for the neural network is the product of the number of different input variables selected and the number of sampling times. In contrast to the methodology adopted in program INTF2, the main purpose of sampling input variables is to provide the neural network with more information on the early stages of the dynamic simulation.

Table 6.26 shows the complete set of input variables recognised by program INTF3. It can

Code	Description
0	Short-circuit at a bus (cv)
1	Branch status (cv)
2	Global load (cv)
3	Global generation (cv)
4	Bus load (cv)
5	Bus generation (cv)
6	Bus reactor (cv)
7	Minimum singular value of G_s
8	Minimum singular value of J
4	Angular speed of generator
5	Angle of generator
6	Angular difference between generators
7	Mechanical power of generator
8	Bus voltage
9	Bus angle
10	Bus active load
11	Bus reactive load
12	Bus active generation
13	Bus reactive generation
14	Bus reactor
15	Motor slip
16	Motor active power
17	Motor reactive power
18	OLTC
19	Dynamic load (active power)
20	Dynamic load (reactive power)

Table 6.26 - Input variables in program INTF3 ("cv" means "control variable")

be seen that the first seven input variables are the control variables themselves.

Figure 6.30 shows the structure of the input vector, where it is assumed that 3 different input variables and 4 sampling times were defined by the user.

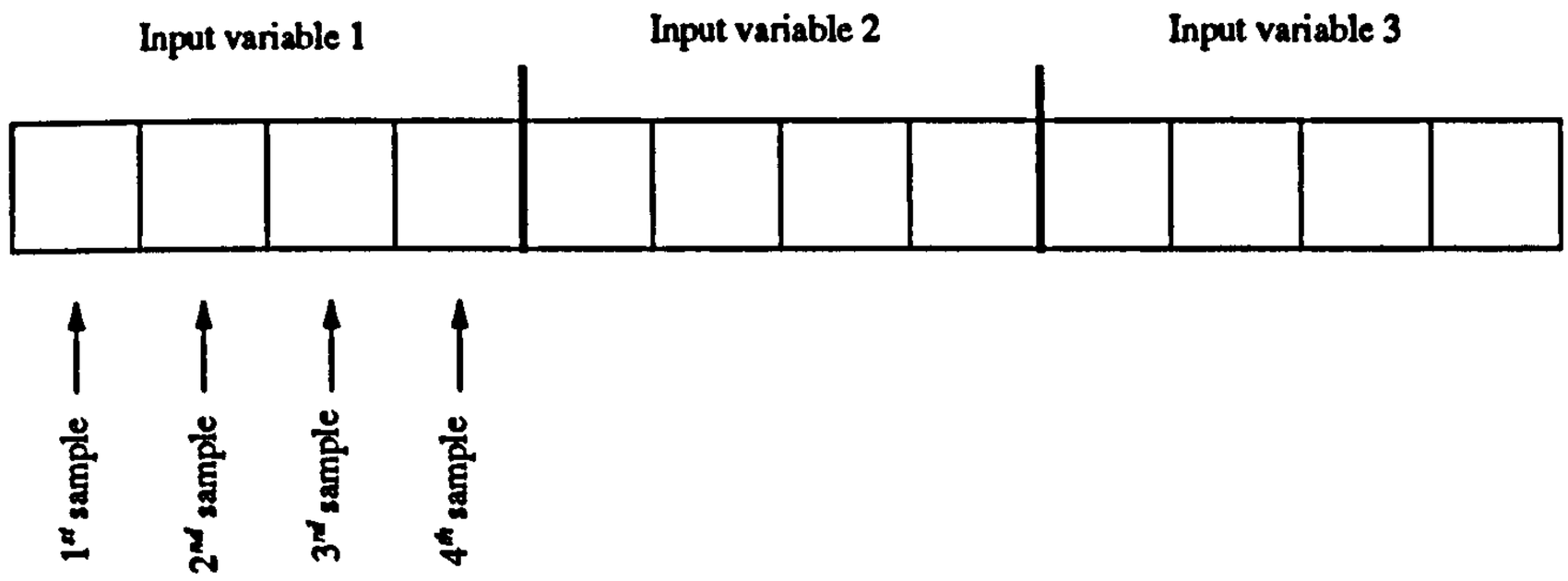


Figure 6.30 - Structure of the input vector in program INTF3

It should be noted that in automatic generation of the input vector, if a short-circuit at a bus were selected as input variable, for every vector being generated the program executes a random extraction so as to decide whether or not a short-circuit will occur during the current simulation run. In the case that a short-circuit has been selected, its starting and clearing times are decided upon further random extractions. The same applies to the case of branch operations (opening and closing). These random extractions are controlled by specific parameters given by the user. With respect to load, generation and reactor levels, the program updates their values at each sampling time, also utilising specific parameters set by the user.

Output variables are similar to their counterparts in program INTF2, and hence Table 6.15 also applies to program INTF3. The final simulation time (variable code 1) is now the *simulation extra time* (SET), which is the time elapsed between the last sampling time and the final simulation time (60 - 15 = 45s in the example of Figure 6.29). It should be noted that the codes for the simulation outcome are also the same as in program INTF2 (see Table 6.16).

In this methodology, the neural network is taught with a comprehensive training set which contains associations between data from the early stages of dynamic simulations and simulation outcomes. The present methodology provides the neural network with more information than the elementary transitions approach (cf. program INTF2). When in processing mode the neural network will be able to read data sampled from the real-time operation of the power system. It is expected that the neural network then be capable of recognising in advance both stable and potentially dangerous situations.

6.5.3 - Results

6.5.3.1 - Introduction

In this sub-section, two series of study cases (series C and D) will be described in order to illustrate the application of the computational system for analysing the VSP using the methodology implemented in program INTF3 (i.e., sampling of input variables). The main output variables to be considered here are the simulation extra time (SET) and the final value of the minimum singular value of matrix G_s (σ_G). In both series C and D the electrical system of Figure 6.10 will be utilised (see sub-section 6.3.3). It should be pointed out that only the most relevant data regarding training/testing sets and MLP architectures will be presented here; the complete set of data is contained in the Appendix.

6.5.3.2 - Series C

This series consists of three different cases, C1 through to C3, as described in Table 6.27. In this table, training/testing set files and MLP architectures are identified by codes which are explained in Tables 6.28 and 6.29.

Case	Code for training/ testing set files	Code for MLP architecture
C1	TTC1	NNC1
C2	TTC1	NNC2
C3	TTC3	NNC2

Table 6.27 - Specification of training/testing set files and MLP architecture in series C

Table 6.30 presents the average value, standard deviation and maximum value for the MLP evaluation error in case C1. In this case, the results obtained with both training and testing sets are very bad, showing that the training set, being too small, does not represent the physical problem properly, and also that the neural network may be too small (with not enough neurons).

In order to analyse the impact of the MLP size, in case C2 only the neural network was changed with respect to case C1 (training/testing sets in case C2 are the same as in case C1). Table

Code	N° of training/ testing vectors	Input variables	Output variables	Max. duration of simulation (s)	Sampling	
					N°	Δt (s)
TTC1	345/330	- State of branch 1000 1100 #1 (†) - State of branch 1100 1200 #1 (†) - Global load (†) - Load at bus 1400 (†) - Reactor at bus 1200 (†) - Reactor at bus 1300 (†) - Reactor at bus 1400 (†) - σ_G	- SET - σ_G	60.	4	5.
TTC3	1043/1165					

Table 6.28 - Main data for training/testing set files in series C († indicates variables also defined as control variables)

Code	MLP structure (N° of neurons in each layer)	Training sessions (N° of iterations/learning rate)				
		1	2	3	4	5
NNC1	32/25/2	1000/1.0	1000/0.9	1000/0.8	1000/0.7	1000/0.6
NNC2	32/25/20/2					

Table 6.29 - Structure and training data for MLPs in series C

Parameter	Training set		Testing set	
	Output variable		Output variable	
	SET	σ_G	SET	σ_G
Average error (%)	8.07	7.29	32.30	28.96
Std. dev. of error (%)	7.21	6.29	25.49	21.39
Maximum error (%)	35.55	40.45	154.98	103.79

Table 6.30 - First set of results - case C1

6.31 shows the first set of results for case C2.

Once again the evaluation error using the training set was very low, but for the testing set this error was too large. Concerning case C1, the hypothesis of training/testing sets too small still remains to be tested. For this reason, in case C3 only the training/testing sets were changed so as to include more vectors (the MLP structure is the same as in case C2). Table 6.32 shows the first set

Parameter	Training set		Testing set	
	Output variable		Output variable	
	SET	σ_G	SET	σ_G
Average error (%)	0.71	0.45	7.82	15.85
Std. dev. of error (%)	0.43	0.34	16.17	14.04
Maximum error (%)	1.76	1.39	104.68	71.58

Table 6.31 - First set of results - case C2

of results for case C3.

Parameter	Training set		Testing set	
	Output variable		Output variable	
	SET	σ_G	SET	σ_G
Average error (%)	1.22	6.32	7.32	18.85
Std. dev. of error (%)	0.90	4.58	18.74	17.84
Maximum error (%)	7.56	29.00	115.19	105.31

Table 6.32 - First set of results - case C3

Finally, Table 6.33 shows the distribution of error in cases C1, C2 and C3. It should be noted that this distribution was computed using the corresponding testing sets only.

Class	Error limits (%)		Case C1		Case C2		Case C3	
	Min.	Max.	SET	σ_G	SET	σ_G	SET	σ_G
1	0	2	2.33	4.32	45.18	10.63	74.64	8.88
2	2	4	4.32	5.32	20.60	8.97	6.70	7.70
3	4	6	3.99	6.31	9.63	7.97	2.26	6.61
4	6	8	3.65	3.65	5.32	9.63	1.90	9.78

Table 6.33 - Distribution of error in cases C1, C2 and C3 (testing sets only) (% of total number of testing vectors)

Class	Error limits (%)		Case C1		Case C2		Case C3	
	Min.	Max.	SET	σ_G	SET	σ_G	SET	σ_G
5	8	10	2.66	3.65	2.66	5.98	1.72	7.07
6	10	12	2.66	2.66	1.99	7.31	0.72	6.97
7	12	14	4.65	4.65	0.66	5.98	0.54	5.25
8	14	16	4.98	3.65	1.00	3.65	0.72	5.16
9	16	18	2.99	2.66	2.66	5.65	0.63	4.53
10	18	20	2.66	3.65	1.33	4.98	0.72	4.71
11	20	22	3.65	4.98	0.33	2.99	0.18	2.90
12	22	24	3.99	1.99	0.00	4.32	0.36	3.53
13	24	26	3.65	3.65	0.33	2.99	0.36	2.99
14	26	28	4.32	2.66	1.00	2.33	0.36	1.99
15	28	30+	49.50	46.20	7.31	16.62	8.19	21.93
Total			100.00	100.00	100.00	100.00	100.00	100.00

Table 6.33 - Distribution of error in cases C1, C2 and C3 (testing sets only) (% of total number of testing vectors)

6.5.3.3 - Series D

This series represents further attempts to improve the results previously obtained. The main differences with respect to series C lie in the duration of the dynamic simulations (now reduced to 30s) and the exclusion of the status of branches from the set of input variables. These variables introduce high non-linearities in the input set, due to their logical nature (they can only be equal to 0 or 1).

The series consists of three different cases, D1 through to D3, as described in Table 6.34. In this table, training/testing sets and MLP architectures are identified by codes which are explained in Tables 6.35 and 6.36.

Case	Code for training/ testing set files	Code for MLP architecture
D1	TTD1	NND1

Table 6.34 - Specification of training/testing set files and MLP architecture in series D

Case	Code for training/ testing set files	Code for MLP architecture
D2	TTD2	NND2
D3	TTD3	NND3

Table 6.34 - Specification of training/testing set files and MLP architecture in series D

Code	N° of training/ testing vectors	Input variables	Output variables	Max. duration of simulation (s)	Sampling	
					N°	Δt (s)
TTD1	1000/1000	- Global load (†) - Load at bus 1400 (†) - Reactor at bus 1200 (†) - Reactor at bus 1300 (†) - Reactor at bus 1400 (†) - σ_G	- SET - σ_G	30.	3	5.
TTD2	4891/4205					
TTD3	980/848					

Table 6.35 - Main data for training/testing set files in series D († indicates variables also defined as control variables)

Code	MLP structure (N° of neurons in each layer)	Training sessions (N° of iterations/learning rate)				
		1	2	3	4	5
NND1	18/15/10/2	1000/1.0	1000/0.9	1000/0.8	1000/0.7	1000/0.6
NND2	18/25/15/2					
NND3	18/20/15/2					

Table 6.36 - Structure and training data for MLPs in series D

Table 6.37 presents the average value, standard deviation and maximum value for the evaluation error in case D1.

In case D2, the number of training and testing vectors were greatly increased so as to provide the MLP with a more representative data set. At the same time, more hidden-layer neurons were defined in order to allow the MLP to learn the training set more easily. Table 6.38 shows the

Parameter	Training set		Testing set	
	Output variable		Output variable	
	SET	σ_G	SET	σ_G
Average error (%)	11.94	6.32	23.82	16.85
Std. dev. of error (%)	5.22	7.05	28.36	23.32
Maximum error (%)	68.37	70.17	155.46	122.65

Table 6.37 - First set of results - case D1

results obtained in this case. It can be seen that the MLP evaluation error is excessively high.

Parameter	Training set		Testing set	
	Output variable		Output variable	
	SET	σ_G	SET	σ_G
Average error (%)	11.64	7.60	16.01	11.90
Std. dev. of error (%)	17.15	10.31	24.24	17.08
Maximum error (%)	121.65	80.46	137.09	121.36

Table 6.38 - First set of results - case D2

One important concern in cases D1 and D2 was that the distribution of training and testing vectors was very unbalanced, with far more vectors having SET = (30 - 2*5) = 20s. This means that the large majority of vectors represent normal situations, with the dynamic simulation ending without any instability or collapse. Table 6.39 illustrates this point through the distribution of vectors according to the value of the output variables. In this table, the range between the minimum and maximum values of each output variable was divided in ten equal intervals and the vectors were classified accordingly.

Case D3 was then created from case D2 but excluding all those vectors which represented successful simulations. Table 6.40 shows the results obtained in this case.

Finally, Table 6.41 shows the distribution of error in cases D1, D2 and D3. It should be noted that this distribution was computed using the corresponding testing sets only. Although the

Set/output variable		Min. val.	Max. val.	Distribution										
				1	2	3	4	5	6	7	8	9	10	Tot.
Training	SET	0.	20.000	899	31	15	23	44	20	20	7	3	3829	4891
	σ_G	0.00033	2.59622	181	904	53	14	15	64	1952	1254	413	41	4891
Testing	SET	0.	20.000	822	19	15	20	30	12	18	2	1	3266	4205
	σ_G	0.00240	2.50549	187	782	28	10	10	32	888	1702	476	90	4205

Table 6.39 - Distribution of vectors in training/testing sets TTD2 according to the output variables (number of vectors in each class)

Parameter	Training set		Testing set	
	Output variable		Output variable	
	SET	σ_G	SET	σ_G
Average error (%)	0.65	1.22	7.05	6.14
Std. dev. of error (%)	0.60	1.03	18.88	15.84
Maximum error (%)	8.02	6.82	151.37	100.12

Table 6.40 - First set of results - case D3

Class	Error limits (%)		Case D1		Case D2		Case D3	
	Min.	Max.	SET	σ_G	SET	σ_G	SET	σ_G
1	0	2	0.61	16.79	53.93	23.54	77.02	65.89
2	2	4	2.25	17.71	1.65	18.55	3.79	16.14
3	4	6	3.89	16.48	1.34	11.12	2.44	3.30
4	6	8	2.25	9.62	1.03	12.77	1.22	2.81
5	8	10	3.58	6.14	1.86	5.49	1.22	1.47
6	10	12	12.59	3.28	2.89	3.46	0.49	1.10
7	12	14	47.08	2.15	2.65	2.60	0.86	0.61
8	14	16	2.76	1.94	1.53	1.65	0.61	0.73
9	16	18	2.05	1.23	1.74	1.98	0.12	0.24
10	18	20	1.54	1.23	0.64	1.48	1.83	0.12

Table 6.41 - Distribution of error in cases D1, D2 and D3 (testing sets only) (% of total number of testing vectors)

Class	Error limits (%)		Case D1		Case D2		Case D3	
	Min.	Max.	SET	σ_G	SET	σ_G	SET	σ_G
11	20	22	0.82	1.02	0.91	1.31	0.73	0.49
12	22	24	0.82	1.23	0.69	1.74	0.37	0.24
13	24	26	0.51	0.51	0.76	1.05	0.37	0.00
14	26	28	0.31	0.51	11.70	0.76	0.61	0.12
15	28	30+	18.94	20.16	16.68	12.50	8.32	6.74
Total			100.00	100.00	100.00	100.00	100.00	100.00

Table 6.41 - Distribution of error in cases D1, D2 and D3 (testing sets only) (% of total number of testing vectors)

evaluation error in case D2 is in average higher than in case D1, it can be seen from the histogram that the error distribution in case D2 is better than in case D1, with a larger proportion of testing vectors in the first error class (error of up to 2%). Case D3 shows the best distribution of all cases, but still presents an unacceptable number of vectors with evaluation error above 10%.

6.5.4 - Discussion of Series C and D cases

The methodology implemented in program INTF3 sought to correct the learning problem that had appeared in the previous methodology (INTF2).

The analysis of the distribution of the evaluation error across the testing sets in series B, C and D (histograms of Tables 6.23, 6.33 and 6.41 respectively) shows that there is no clear advantage of the methodology based upon sampling of input variables with respect to the methodology based upon elementary transitions. In all cases the proportion of testing vectors for which the evaluation error is large (error greater or equal 10%, for instance) became excessively large.

Case D3, where the successful dynamic simulations were not included, produced the best results. This confirmed the idea that a large number of training vectors with the same SET do make the training process more difficult. This would suggest the use of a pre-processor step for classifying the training vectors prior to the start of the training.

A number of hypothesis were formulated so as to explain the learning problems that occurred with the two methodologies. They will be discussed in detail in the summary section, later in this chapter.

6.6 - Processing time

Table 6.42 presents a summary of processing times for all programs used in this chapter. It is included with the single purpose of providing the order of magnitude of the computational cost. All times in Table 6.42 were obtained with the personal computer described in sub-section 6.3.2.9.

Program	Case	CPU time	
FLOW4	Base case of Figure 6.10 (sub-section 6.3.3.1) with convergence to a maximum absolute bus mismatch of 10^{-10} pu in 4 iterations	0.16 s	
DSIM3	Dynamic simulation case in sub-section 6.3.3.2 (total simulated time: 21.0 s)	20.59 s	
INTF2	Generation of training set in case TTB1 (sub-section 6.4.3)	19h 30min	
INTF3	Generation of training set in case TTD3 (sub-section 6.5.3.3)	13h 16min	
STATISTIC1	Case NND3 (sub-section 6.5.3.3)	Training mode	4h 46min
		Processing mode	- 848 testing vectors: 1.25 s - 1 testing vector: 0.00147 s

Table 6.42 - Summary of processing times

6.7 - Summary

This chapter has proposed an approach for detecting situations in which a power system might experience dynamic voltage collapses. The approach is based on dynamic simulations of the power system in the time domain, which allow for the dynamic aspects of the VSP.

The proposed approach was implemented as a self-contained computational system. Initially, the dynamic simulation program -one of the most important components within the system- was described in detail. Two examples of dynamic simulation, which illustrated some important mechanisms associated with voltage instabilities, were discussed in sub-sections 6.3.3.2 and 6.3.3.3.

In a second stage, in sections 6.4 and 6.5, a neural network-based prediction tool was developed. Its purpose was to learn about the VSP from a large number of dynamic simulation scenarios, and to use this knowledge in the prediction of the future behaviour of the power system given actual data from the system operation. The main indicator considered here was the simulation extra time, which is the time elapsed between the last instant when a piece of information was passed on to the neural network and the end of the simulation.

In sub-sections 6.4.3 and 6.5.3, the proposed approach was used in three different series of study cases. The results cannot be considered as definitive due to unacceptably high evaluation errors that were encountered. The following hypotheses to explain these high errors were formulated:

(1) too large MLPs, so that the training sets would be just memorised without any useful feature extraction;

(2) improper mapping between the dynamic simulation and the MLP;

(3) inadequate size of training/testing sets which would not properly represent the physical problem (too small sets);

(4) inconsistent training/testing sets, containing vectors with relatively similar values for the input variables but substantially different values for the output variables.

Hypotheses (1) and (3) were studied in detail in sub-sections 6.4.3 and 6.5.3 through the adjustment of relevant parameters. They could be discarded since no significant improvements were achieved with the associated adjustments.

A first attempt to deal with hypothesis (2) is represented by the methodology implemented in program INTF3 (sampling input variables, sub-section 6.5.2), which is an upgrade of the original methodology implemented in program INTF2. Hypotheses (2) and (4) remain to be further investigated in the future.

Despite the negative results obtained in this chapter, the main characteristics of the proposed approach should be underlined. Firstly, the earlier results for static voltage stability obtained with the computational system of Chapter 5 were very encouraging. Good accuracy and extremely low processing times were achieved with the MLP architecture. Secondly, it was seen that the dynamic simulation technique does take into account aspects that cannot be considered using a static load-flow program alone. Furthermore, conventional dynamic simulation is intrinsically very time consuming, which calls for an auxiliary tool, such as the MLP, to circumvent this major drawback.

Finally, from what has been presented in this chapter, it is clear that further investigation is

required in order to better assess whether or not the proposed methodology can ultimately be validated. This is further discussed in Chapter 7.

CHAPTER 7

DISCUSSION AND CONCLUSIONS

7.1 - General summary

This thesis dealt with the voltage stability problem in electric power systems within the framework of artificial neural networks. A number of computational models were developed and implemented in order to analyse the models described in the literature and also to validate the ideas proposed in this work.

In Chapter 3 an extensive literature review provided a comprehensive overview of the state-of-the-art in both the VSP and the applications of artificial neural networks in power systems.

The review of the VSP clearly indicated that this problem is still in an early stage of maturity. It is difficult even to establish a rigorous formulation of the problem. The relatively large number of approaches to the problem and the controversial discussions found in many papers confirm this statement. Within the spectrum of existing methodologies, it is possible to find methods ranging from well-known linear techniques to sophisticated approaches based upon advanced non-linear mathematics. The former disregard some important aspects of the problem but can be applied to large power systems, whereas the complexity of the latter prevents its application in cases with more than just a few buses. In between, the dynamic simulation technique offers excellent modelling capabilities, which ultimately allow the study of a large number of operational scenarios. This is most valuable for improving the understanding of the problem, with respect to its causes and also to the establishment of preventive controls for avoiding voltage collapses.

The survey of the application of ANNs in power systems showed that the MLP architecture is by far the most popular ANN model used, as in many other research fields. The most successful applications in power systems can be found in security analysis and load forecasting. In security analysis the most valuable feature of the MLP is perhaps its extremely fast computation, which makes it a serious candidate for on-line applications. In the load forecasting area, it was possible to

find applications where the predicting errors were very low. Also, the MLP network showed its ability to overcome the main drawbacks of the conventional methodologies traditionally used in load forecasting, such as the difficulty of dealing with weather information.

The literature survey also suggested and reinforced the main idea of the present work, which is to bring together the VSP and the ANN frameworks. No previous research in this area could be found.

Chapter 4 constitutes a complementary literature review of the vast field of ANN. The aim here was to study the main ANN models developed so far. The computational implementation of these models provided an adequate level of understanding about the models and their applicability in practical problems.

In connection with the backpropagation algorithm, a modification to the basic procedure was devised and implemented. This modification -an internal iteration loop- allows every training vector to be operated upon more than once in a given external iteration. This simple modification improved the overall training. Care must be taken, however, when choosing the maximum number of internal iterations. If too large, this parameter will slow down the whole training process. Further research is needed to produce some guidance for choosing this new parameter.

The aim of any ANN is to mimic the functioning of the human brain and this is still a distant goal to be achieved, simply because the human brain still remains mostly undiscovered. Therefore, it is never redundant to emphasise the little equivalence between the human brain and its models. Nevertheless, these models can capture one or more functional aspects of the human brain, such as associative memory, learning, interpolation, etc. The whole field of ANN, and especially the newer architectures such as the recurrent networks and the ART paradigm, is developing very fast. In just a few years' time the field will certainly look very different from what it does today.

In Chapter 5, the first results of the proposed work were produced. A static voltage stability index -the minimum singular value of a Jacobian-related matrix- was adopted and a MLP network was trained so as to compute this index from the knowledge of some input variables. It was shown that the MLP is capable of computing the voltage stability index nearly 1000 times faster than the load-flow program and without loss of accuracy, a compromise that frequently arises in computational applications. The main drawbacks of this approach are perhaps the lengthy training phase of the MLP (when using the Backpropagation algorithm) and the scale-up problem. Long training times do not usually pose a serious problem, since the training can be executed off-line (as is often the case). The major problem is the scale-up of the ANN model, which means implementing a neural network with as many input variables as necessary in order to represent a real system. This could mean 500, 1000 or even more input variables, and this is not possible -or, at least, not worthwhile- with today's computers. This will be further discussed later in this chapter.

Chapter 6 represents the natural extension of the work previously developed. Of the many improvements that could have made to the static approach of Chapter 5, one of them was the modelling of the system and its associated dynamic phenomena, and this was the chosen path.

A dynamic simulation program was developed in order to create a large number of scenarios from which to extract knowledge about the VSP. This simulation program is rather simple and it did not consider many dynamic models of the power system, especially those related to generators. In any case, this can be easily improved either through a more sophisticated simulation program or by the use of real system data extracted from practical measurements.

Two major reasons for choosing the simulation technique were its excellent modelling capabilities and the fact that it is an expensive technique in computational terms. If an MLP could be taught with data from dynamic simulations, then later on, in the processing mode, it would provide the same answers as the simulation program but in a much shorter time. The speed-up factor could easily be more than 1000, the value obtained in the load-flow case, because a dynamic simulation usually takes much longer than a load-flow to be executed.

A new voltage stability index -the simulation extra time- was proposed and MLP networks were trained with data from the dynamic simulation in order to teach them how to give some indication about the possibility of occurrence of voltage collapse. The results from this line of research were not as good as those obtained in Chapter 5, despite the development and implementation of two different techniques for mapping the results from the dynamic simulation onto the MLP problem language. Although it was possible to compute the majority of the testing vectors with a low evaluation error, it was not possible to eliminate a fraction of these vectors for which the error was excessively large. This may have happened due to a number of reasons, among which there are (*i*) an improper mapping of the dynamic simulation and (*ii*) incompatible training vectors coexisting in the same training file (i.e., vectors with similar input values but substantially different output values). Despite the considerable efforts spent on the study of this problem, no conclusive evidence could be found.

Finally, it should be noted that an important characteristic of the proposed approaches is their modularity. Whatever improvements are introduced in the future, as to either faster training algorithms and/or better analytical tools for the electrical problem, they will be easily accommodated in the proposed framework without implying major structural changes.

7.2 - Original contributions

7.2.1 - VSP in conjunction with ANN

The first major contribution of this work is the development of ANN-based techniques for studying the VSP. No similar approach could be found in the technical literature.

ANNs constitute a vast set of mathematical techniques that can perform some human-like tasks. In the present work, the emphasis was put on the practical aspects of the application of ANN, namely feature extraction and computational speed. Feature extraction allows an ANN to learn useful information about a specific problem just through the analysis of adequate examples. This is most valuable when the explicit specification of the problem is difficult or even impossible. The issue of computational speed is obviously crucial to the development of on-line tools such as the ones considered in this work.

7.2.2 - Modification to the basic backpropagation procedure

The careful study of the basic backpropagation procedure, together with the execution of a large number of study cases and a comprehensive sensitivity analysis with respect to some basic parameters, provided an adequate level of understanding of the backpropagation algorithm. A consequence of this study was the implementation of a simple modification which improved the overall training process.

7.2.3 - Fast computation of a static voltage stability index

The first important result of the coupling between the VSP and the ANN frameworks was the development of a MLP network capable of computing the static voltage stability index with a considerable speed-up factor with respect to the conventional load-flow technique. Such an MLP could then be thought of as a serious candidate for on-line applications in control centres of power systems. This constitutes a major contribution of this work.

7.2.4 - Use of MLP as a predicting tool for dynamic voltage stability

Another important, original idea of the present work was the attempt to train a neural network with data from the temporal evolution of a power system and require, from the trained network, a prediction about the future development of the system, given specific information on the recent past and present behaviour.

Although the results in this particular area did not prove as satisfactory as expected, it is the

opinion of the author that this basic idea is still feasible (see next section).

7.3 - Topics for further development

One of the most important topics for further development is the investigation of the learning problem which was encountered when training MLPs with data from the dynamic simulation (cf. Chapter 6). If the hypothesis of an improper mapping between the dynamic simulation and the ANN proves correct, then alternative schemes for this mapping should be investigated. In particular, new techniques for studying temporal series and the possible ways to process the information contained therein must be studied carefully.

It is the opinion of the author that MLP-based tools work best if designed as smaller subsystems, which would then constitute specialised parts of more global systems. This is important in cases when the data represent highly non-linear functions; it is then convenient to "break down" these non-linearities and to distribute smaller tasks among the subsystems. Using a unique MLP for executing a large task -such as evaluating a global voltage stability index of a large power system- can prove very demanding, especially with respect to the design of the training set and the training phase as well. Therefore, a new, modular design for the ANN section should be considered as an alternative to the global technique developed in this work.

Another topic to be considered for further development is the extension of the scope of the present work. As stated in Chapter 2, the main concern here was the issue of proximity to voltage collapse and the development of on-line tools for supporting the operation of a power system. Such tools are more useful if used in conjunction with other methodologies for suggesting adequate corrective actions once a stability problem has been detected. Therefore, the development of such new methodologies should be considered as a natural extension to the present work. From the literature review that was presented in Chapter 3, the first alternative to be considered would be the modal analysis of the Jacobian-related matrices. Although this is a linear approach, it was seen that useful information can be extracted from eigenvalues and associated eigenvectors (regarding critical areas within the system and the most effective buses for the application of corrective actions).

The application of the methodology developed in this thesis to large power systems should be considered as an important extension. The scope of the work, and especially the learning problem found during the implementation of the dynamic approach, prevented the application of the methodology in power systems other than the one described in Chapter 6.

Finally, the use of other ANN architectures should also be considered. It was seen in Chapter 4 that the various ANN models available are capable of representing specific characteristics of

the human brain, but none of the models encompasses all of the desired characteristics. Ways of exploring the complementary features of the models should be investigated. As a simple example, an ART model could be used to pre-process the training vectors to be used in MLP training. In this way, the vectors would be grouped and an average exemplar could substitute the whole group. The size of the training set would be thus reduced, and consequently the training times might be reduced as well.

REFERENCES

This section lists the publications that were directly referred to in the thesis; they appear in the same order as they were cited in the text. Headers are provided in order to facilitate the search for specific publications.

Publications that are not directly referred to in the thesis, but which may be useful for extending the concepts developed in this work, are listed in the following section, named Bibliography.

1. Application of Artificial Intelligence in Electric Power Systems

[1] T. S. Dillon and M. A. Laughton (editors): "Expert systems applications in power systems", Prentice-Hall International Series in Power Systems Computation, 1990.

2. Voltage Stability Problem in Electric Power Systems (I)

[2] C. Barbier and J.-P. Barret: "An analysis of phenomena of voltage collapse on a transmission system", Revue Générale d'Electricité, Tome 89, Nº 10, pp 672-690, October 1980.

[3] F. Bourgin, G. Testud, B. Heilbronn and J. Verseille: "Present practices and trends on the French power system to prevent voltage collapse", IEEE Transactions on Power Systems, Vol. 8, Nº 3, pp 778-788, August 1993.

[4] K. Walve: "Modelling of power system components at severe disturbances", CIGRE Report 38-18, 1986.

[5] K. Takahashi and Y. Nomura: "The power system failure on July 23, 1987, in Tokyo", CIGRE SC 37 on Power System Planning and Development, Montreal, 22-25 September, 1987.

[6] G. Brownell and H. Clark: "Analysis and solutions for bulk system voltage instability", IEEE Computer Applications in Power (Magazine), Vol. 2, Nº 3, pp 31-35, July 1989.

[7] M. K. Pal: "Voltage stability conditions considering load characteristics", IEEE Transactions on Power Systems, Vol. 7, Nº 1, pp 243-249, February 1992.

[8] V. A. Venikov, V. A. Stroev, V. I. Idelchik and V. I. Tarasov: "Estimation of electrical power system steady-state stability in load flow calculations", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-94, Nº 3, pp 1034-1041, May/June 1975.

[9] A. Tiranuchit, L. M. Ewerbring, R. A. Duryea, R. J. Thomas and F. T. Luk: "Towards a computationally feasible on-line voltage stability index", IEEE Transactions on Power Systems, Vol. 3, Nº 2, pp 669-675, May 1988.

[10] A. Tiranuchit and R. J. Thomas: "A posturing strategy against voltage instabilities in electric power systems", IEEE Transactions on Power Systems, Vol. 3, Nº 1, pp 87-93, February 1988.

3. Numerical Analysis and Matrix Computations (I)

[11] C. L. Lawson and R. J. Hanson: "Solving least squares problems", Prentice-Hall, Englewood Cliffs, NJ, 1974.

[12] H. R. Schwarz: "Numerical analysis: a comprehensive introduction", J. Wiley & Sons, 1989.

[13] G. H. Golub and C. F. Van Loan: "Matrix Computations", North Oxford Academic, Oxford, 1983.

4. Voltage Stability Problem in Electric Power Systems (II)

[14] P-A Löf, T. Smed, G. Andersson and D. J. Hill: "Fast calculation of a voltage stability index", IEEE Transactions on Power Systems, Vol. 7, Nº 1, pp 54-64, February 1992.

[15] P. Kessel and H. Glavitsch: "Estimating the voltage stability of a power system", IEEE Transactions on Power Delivery, Vol. PWRD-1, Nº 3, pp 346-354, July 1986.

[16] B. Gao, G. K. Morison and P. Kundur: "Voltage stability evaluation using modal analysis", IEEE Transactions on Power Systems, Vol. 7, Nº 4, pp 1529-1542, November 1992.

[17] P. W. Sauer and M. A. Pai: "Power system steady-state stability and the load-flow Jacobian", IEEE Transactions on Power Systems, Vol. 5, Nº 4, pp 1374-1383, November 1990.

[18] V. Ajjarapu and C. Christy: "The continuation power flow: a tool for steady state voltage stability analysis", IEEE Transactions on Power Systems, Vol. 7, Nº 1, pp 416-423, February 1992.

[19] K. Iba, H. Suzuki, M. Egawa and T. Watanabe: "Calculation of critical loading condition with nose curve using homotopy continuation method", IEEE Transactions on Power Systems, Vol. 6, Nº 2, pp 584-593, May 1991.

[20] A. Semlyen, B. Gao and W. Janischewskyj: "Calculation of the extreme loading condition of a power system for the assessment of voltage stability", IEEE Transactions on Power Systems, Vol. 6, Nº 1, pp 307-315, February 1991.

[21] Y. Kataoka: "An approach for the regularization of a power flow solution around the maximum loading point", IEEE Transactions on Power Systems, Vol. 7, Nº 3, pp 1068-1077, August 1992.

[22] Y. Tamura, H Mori and S. Iwamoto: "Relationship between voltage instability and multiple load flow solutions in electric power systems", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, Nº 5, pp 1115-1125, May 1983.

[23] M. Dehnel and H. W. Dommel: "A method for identifying weak nodes in nonconvergent load flows", IEEE Transactions on Power Systems, Vol. 4, Nº 2, pp 801-807, May 1989.

[24] F. D. Galiana and Z. C. Zeng: "Analysis of the load flow behaviour near a Jacobian singularity", IEEE Transactions on Power Systems, Vol. 7, Nº 3, pp 1362-1369, August 1992.

[25] Z. C. Zeng, F. D. Galiana, B. T. Ooi and N. Yorino: "A simplified approach to estimate maximum loading conditions", IEEE Transactions on Power Systems, Vol. 8, Nº 2, pp 646-654, May 1993.

[26] A. Berizzi, A. Silvestri, D. Zaninelli and R. Marconato: "Static voltage collapse when increasing the load: an algorithm of fast convergence recognition at each bus", Proceedings of the XI Power Systems Computation Conference, Avignon, France, 30 August - 3 September 1993, pp 1123-1131.

[27] T. V. Cutsem: "A method to compute reactive power margins with respect to voltage collapse", IEEE Transactions on Power Systems, Vol. 6, Nº 1, pp 145-156, February 1991.

[28] T. H. Jung, K. J. Kim and F. L. Alvarado: "A marginal analysis of the voltage stability with load variations", Proceedings of the X Power Systems Computation Conference, Graz, Austria, August 1990, pp 1196-1201.

[29] N. Flatabø, R. Ognedal and T. Carlsen: "Voltage stability condition in a power transmission system calculated by sensitivity methods", IEEE Transactions on Power Systems, Vol. 5, Nº 4, pp 1286-1293, November 1990.

[30] N. Flatabø, O. B. Fosso, R. Ognedal, T. Carlsen and K. R. Heggland: "A method for calculation of margins to voltage instability applied on the Norwegian system for maintaining required security level", IEEE Transactions on Power Systems, Vol. 8, Nº 3, pp 920-928, August 1993.

[31] M. M. Begovic and A. G. Phadke: "Control of voltage stability using sensitivity analysis", IEEE Transactions on Power Systems, Vol. 7, Nº 1, pp 114-123, February 1992.

[32] S. Abe, Y. Fukunaga, A. Isono and B. Kondo: "Power system voltage stability", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, Nº 10, pp 3830-3840, October 1982.

[33] H. Ohtsuki, A. Yokoyama and Y. Sekine: "Transient P-V curves for analysis of transient voltage stability", Proceedings of the X Power Systems Computation Conference, Graz, Austria, August 1990, pp 1202-1209.

[34] I. A. Hiskens and D. J. Hill: "Energy functions, transient stability and voltage behaviour in power systems with nonlinear loads", IEEE Transactions on Power Systems, Vol. 4, Nº 4, pp 1525-1533, October 1989.

[35] C. L. DeMarco and T. J. Overbye: "An energy based security measure for assessing vulnerability to voltage collapse", IEEE Transactions on Power Systems, Vol. 5, Nº 2, pp 419-427, May 1990.

[36] T. J. Overbye and C. L. DeMarco: "Improved techniques for power system voltage stability assessment using energy methods", IEEE Transactions on Power Systems, Vol. 6, Nº 4, pp 1446-1452, November 1991 and IEEE Transactions on Power Systems, Vol. 7, Nº 1, pp 344-345, February1992.

[37] T. J. Overbye: "Use of energy methods for on-line assessment of power system voltage security", IEEE Transactions on Power Systems, Vol. 8, Nº 2, pp 452-458, May 1993.

[38] H. G. Kwatny, A. K. Pasrija and L. Y. Bahar: "Static bifurcations in electric power networks: loss of steady-state stability and voltage collapse", IEEE Transactions on Circuits and Systems, Vol. CAS-33, Nº 10, pp 981-991, October 1986.

[39] C. A. Cañizares, F. L. Alvarado, C. L. DeMarco, I. Dobson and W. F. Long: "Point of col-

lapse methods applied to AC/DC power systems", IEEE Transactions on Power Systems, Vol. 7, Nº 2, pp 673-683, May 1992.

[40] V. Ajjarapu and B. Lee: "Bifurcation theory and its application to nonlinear dynamical phenomena in an electrical power system", IEEE Transactions on Power Systems, Vol. 7, Nº 1, pp 424-431, February 1992.

[41] H. D. Chiang, W. Ma, R. J. Thomas and J. S. Thorp: "A tool for analyzing voltage collapse in electric power systems", Proceedings of the X Power Systems Computation Conference, Graz, Austria, August 1990, pp 1210-1217.

[42] R. J.-Jumeau and H. D. Chiang: "Parameterizations of the load-flow equations for eliminating ill-conditioning load flow solutions", IEEE Transactions on Power Systems, Vol. 8, Nº 3, pp 1004-1012, August 1993.

[43] I. Dobson and L. Lu: "New methods for computing a closest saddle-node bifurcation and worst case load power margin for voltage collapse", IEEE Transactions on Power Systems, Vol. 8, Nº 3, pp 905-913, August 1993.

[44] N. Yorino, H. Sasaki, Y. Masuda, Y. Tamura, M. Kitagawa and A. Oshimo: "An investigation of voltage stability problems", IEEE Transactions on Power Systems, Vol. 7, Nº 2, pp 600-611, May 1992.

[45] C. O. Nwankpa and S. M. Sahidehpour: "A new approach to the indication of voltage collapse in electric power systems", Proceedings of the X Power Systems Computation Conference, Graz, Austria, August 1990, pp 1181-1188.

[46] M. M. Begovic and A. G. Phadke: "Dynamic simulation of voltage collapse", IEEE Transactions on Power Systems, Vol. 5, Nº 4, pp 1529-1534, November 1990.

[47] M. M. Begovic and A. G. Phadke: "Voltage stability assessment through measurement of a reduced state vector", IEEE Transactions on Power Systems, Vol. 5, Nº 1, pp 198- 203, February 1990

[48] W. R. Lachs and D. Sutanto: "Voltage instability in interconnected power systems: a simulation approach", IEEE Transactions on Power Systems, Vol. 7, Nº 2, pp 753-761, May 1992.

[49] T. Van Cutsem: "Analysis of emergency voltage situations", Proceedings of the XI Power Systems Computation Conference, Avignon, France, 30 August - 3 September 1993, pp 323-330.

[50] G. K. Morison, B. Gao and P. Kundur: "Voltage stability analysis using static and dynamic approaches", IEEE Transactions on Power Systems, Vol. 8, Nº 3, pp 1159-1171, August 1993.

[51] P. Kundur, G. J. Rogers, D. Y. Wong and M. G. Lauby: "A comprehensive power system stability analysis computer programs package", Proceedings of the Power Plant and Power System Training, Modelling and Simulation Conference, Miami Beach, Florida, 17-19 April 1991.

[52] J. Medanic, M. I. Spong and J. Christensen: "Discrete models of slow voltage dynamics for under load-tap changing transformer coordination", IEEE Transactions on Power Systems, Vol. PWRS-2, Nº 4, pp 873-882, November 1987.

[53] C. C. Liu and K. T. Vu: "Analysis of tap-changer dynamics and construction of voltage stability regions", IEEE Transactions on Circuits and Systems, Vol. 36, Nº 4, pp 575-590, April 1989.

[54] H. Ohtsuki, A. Yokoyama and Y. Sekine: "Reverse action of on-load tap changer in association with voltage collapse", IEEE Transactions on Power Systems, Vol. 6, Nº 1, pp 300-306, February 1991.

[55] I. A. Hiskens and C. B. McLean: "SVC behaviour under voltage collapse conditions", IEEE Transactions on Power Systems, Vol. 7, Nº 3, pp 1078-1087, August 1992.

5. Application of Artificial Neural Networks in Electric Power Systems

[56] M. A. El-Sharkawi, R. J. Marks, M. E. Aggoune, D. C. Park, M. J. Damborg and L. E. Atlas: "Dynamic security assessment of power systems using back error propagation artificial neural networks", Second Symposium on Expert Systems Application to Power Systems, Seattle, Washington, 17-20 July 1989, pp 366-370.

[57] M. A. El-Sharkawi, R. J. Marks, M. J. Damborg, L. E. Atlas, D. A. Cohn and M. Aggoune: "Artificial neural networks as operator aid for on-line static security assessment of power systems", Proceedings of the X Power Systems Computation Conference, Graz, Austria, pp 895-901, August 1990.

[58] M. Aggoune, M. A. El-Sharkawi, D. C. Park, M. J. Damborg and R. J. Marks II: "Preliminary results on using artificial neural networks for security assessment", IEEE Transactions on Power Systems, Vol. 6, Nº 2, pp 890-896, May 1991.

[59] D. J. Sobajic and Y.-H. Pao: "Artificial neural-net based dynamic security assessment for electric power systems", IEEE Transactions on Power Systems, Vol. 4, Nº 1, pp 220- 228, February 1989.

[60] Y.-H. Pao and D. J. Sobajic: "Combined use of unsupervised and supervised learning for dynamic security assessment", IEEE Transactions on Power Systems, Vol. 7, Nº 2, pp

878-884, May 1992.

[61] D. Niebur and A. J. Germond: "Power system static security assessment using the Kohonen neural network classifier", IEEE Transactions on Power Systems, Vol. 7, Nº 2, pp 865- 872, May 1992.

[62] H. Mori, Y. Tamaru and S. Tsuzuki: "An artificial neural-net based technique for power system dynamic stability with the Kohonen model", IEEE Transactions on Power Systems, Vol. 7, Nº 2, pp 856-864, May 1992.

[63] J. N. Fidalgo, J. A. Peças Lopes, V. Miranda and L. B. Almeida: "Fast assessment of transient stability margins by a neural network approach", Proceedings of the XI Power Systems Computation Conference, Avignon, France, 30 August - 3 September 1993, pp 81-87.

[64] A. A. Fouad, Q. Zhou and J. Davidson: "Security/vulnerability assessment of a stability-limited power system using artificial neural networks", Proceedings of the XI Power Systems Computation Conference, Avignon, France, 30 August - 3 September 1993, pp 487-493.

[65] T. S. Dillon, K. Morsztyn and K. Phua: "Short term load forecasting using adaptive pattern recognition and self organising techniques", Proceedings of the V Power Systems Computation Conference, Cambridge, UK, paper 2.4/3, pp 1-16, 1975.

[66] D. C. Park, M. A. El-Sharkawi, R. J. Marks II, L. E. Atlas and M. J. Damborg: "Electric load forecasting using an artificial neural network", IEEE Transactions on Power Systems, Vol. 6, Nº 2, pp 442-449, May 1991.

[67] K. Y. Lee, Y. T. Cha and J. H. Park: "Short term load forecasting using an artificial neural network", IEEE Transactions on Power Systems, Vol. 7, Nº 1, pp 124-132, February 1992.

[68] K. L. Ho, Y. Y. Hsu and C. C. Yang: "Short term load forecasting using a multilayer neural network with an adaptive learning algorithm", IEEE Transactions on Power Systems, Vol. 7, Nº 1, pp 141-149, February 1992.

[69] T. M. Peng, N. F. Hubele and G. G. Karady: "Advancement in the application of neural networks for short-term load forecasting", IEEE Transactions on Power Systems, Vol. 7, Nº 1, pp 250-257, February 1992.

[70] S. T. Chen, D. C. Yu and A. R. Moghaddamjo: "Weather sensitive short-term load forecasting using nonfully connected artificial neural network", IEEE Transactions on Power Systems, Vol. 7, Nº 3, pp 1098-1105, August 1992.

[71] T. M. Peng, N. F. Hubele and G. G. Karady: "An adaptive neural network approach to

one-week ahead load forecasting", IEEE Transactions on Power Systems, Vol. 8, Nº 3, pp 1195-1203, August 1993.

[72] C. N. Lu, H. T. Wu and S. Vemuri: "Neural network based short term load forecasting", IEEE Transactions on Power Systems, Vol. 8, N. 1, pp 336-342, February 1993.

[73] T. Baumann, H. Strasser and H. Landrichter: "Short-term load forecasting methods in comparison: Kohonen learning, backpropagation learning, multiple regression analysis and Kalman filters", Proceedings of the XI Power Systems Computation Conference, Avignon, France, 30 August - 3 September 1993, pp 445-451.

[74] E. H. P. Chan: "Application of neural-network computing in intelligent alarm processing", Power Industry Computer Application Conference, Seattle, Washington, 1-5 May 1989, pp 246-251.

[75] D. J. Sobajic, Y. H. Pao and J. Dolce: "On-line monitoring and diagnosis of power system operating conditions using artificial neural networks", IEEE International Symposium on Circuits and Systems, Portland, Oregon, 8-11 May 1989, pp 2243-2246.

[76] H. Tanaka, S. Matsuda, H. Ogi, Y. Izui, H. Taoka and T. Sakaguchi: "Design and evaluation of neural network for fault diagnosis", Second Symposium on Expert Systems Application to Power Systems, Seattle, Washington, 17-20 July 1989, pp 378-384.

[77] N. Kandil, V. K. Sood, K. Khorasani and R. V. Patel: "Fault identification in an AC-DC transmission system using neural networks", IEEE Transactions on Power Systems, Vol. 7, Nº 2, pp 812-819, May 1992.

[78] H. Ogi, H. Tanaka, Y. Akimoto and Y. Izui: "Abnormality detection for gas insulated switchgear using self-organizing neural networks", Proceedings of the XI Power Systems Computation Conference, Avignon, France, 30 August - 3 September 1993, pp 1171-1177.

[79] R. Fischl, M. Kam, J. C. Chow and S. Ricciardi: "Screening power system contingencies using a backpropagation trained multiperceptron", IEEE International Symposium on Circuits and Systems, Portland, Oregon, 8-11 May 1989, pp 486-494.

[80] R. Fischl, M. Kam, J-C Chow and H. H. Yan: "On the design of neural networks for detecting the limiting contingencies in power system operation", Proceedings of the X Power Systems Computation Conference, Graz, Austria, pp 887-894, August 1990.

[81] R. K. Hartana and G. G. Richards: "Harmonic source monitoring and identification using neural networks", IEEE Transactions on Power Systems, Vol. 5, Nº 4, pp 1098-1104, November 1990.

[82] H. Mori, K. Itou, H. Uematsu and S. Tsuzuki: "An artificial neural-net based method for

predicting power system voltage harmonics", IEEE Transactions on Power Delivery, Vol. 7, Nº 1, pp 402-409, January 1992.

[83] Z. Ouyang and S. M. Shahidehpour: "A hybrid artificial neural network-dynamic programming approach to unit commitment", IEEE Transactions on Power Systems, Vol. 7, Nº 1, pp 236-242, February 1992.

[84] H. Sasaki, M. Watanabe, J. Kubokawa, N. Yorino and R. Yokoyama: "A solution method of unit commitment by artificial neural networks", IEEE Transactions on Power Systems, Vol. 7, Nº 3, pp 974-981, August 1992.

[85] A. P. Alves da Silva, A. M. Leite da Silva, J. C. S. de Souza and M. B. do Coutto Filho: "State forecasting based on artificial neural networks", Proceedings of the XI Power Systems Computation Conference, Avignon, France, 30 August - 3 September 1993, pp 461-467.

[86] H. Mori and S. Tsuzuki: "Power system topological observability analysis using a neural network model", Second Symposium on Expert Systems Application to Power Systems, Seattle, Washington, 17-20 July 1989, pp 385-391.

[87] M. Chow and R. J. Thomas: "Neural network synchronous machine modelling", IEEE International Symposium on Circuits and Systems, Portland, Oregon, 8-11 May 1989, pp 495-498.

[88] S. Matsuda and Y. Akimoto: "The representation of large numbers in neural networks and its applications to economical load dispatching of electric power", IEEE INNS International Joint Conference on Neural Networks, Washington, DC, 18-22 June 1989, pp I.587-I.592.

[89] J. H. Park, Y. S. Kim, I. K. Eom and K. Y. Lee: "Economic load dispatch for piecewise quadratic cost function using Hopfield neural network", IEEE Transactions on Power Systems, Vol. 8, Nº 3, pp 1030-1038, August 1993.

[90] N. Iwan Santoso and O. T. Tan: "Neural-net based real-time control of capacitors installed on Distribution systems", IEEE Transactions on Power Delivery, Vol. 5, Nº 1, pp 266- 272, January 1990.

[91] H. Kim, Y. Ko and K. Y. Jung: "Artificial neural-network based feeder reconfiguration for loss reduction in distributions systems", IEEE Transactions on Power Delivery, Vol. 8, N. 3, pp 1356-1366, July 1993.

[92] S. Ebron, D. L. Lubkeman and M. White: "A neural network approach to the detection of incipient faults on power distribution feeders", IEEE Transactions on Power Delivery, Vol. 5, Nº 2, pp 905-914, April 1990.

[93] A. F. Sultan, G. W. Swift and D. J. Fedirchuk: "Detection of high impedance arcing faults using a multi-layer perceptron", IEEE Transactions on Power Delivery, Vol. 7, Nº 4, pp 1871-1877, October 1992.

[94] M. Y. Chow, S. O. Yee and L. S. Taylor: "Recognizing animal-caused faults in power distribution systems using artificial neural networks", IEEE Transactions on Power Delivery, Vol. 8, N. 3, pp 1268-1273, July 1993.

[95] Y. J. Feria, J. D. McPherson and D. J. Rolling: "Cellular neural networks for eddy current problems", IEEE Transactions on Power Delivery, Vol. 6, Nº 1, pp 187-195, January 1991.

6. Artificial Neural Networks

[96] P. D. Wasserman: "Neural computing - theory and practice", Van Nostrand Reinhold, New York, 1989.

[97] R. Beale and T. Jackson: "Neural computing: an introduction", Adam Hilger, Bristol, 1990.

[98] T. J. Sejnowsky and C. R. Rosenberg: "Parallel networks that learn to pronounce English text", Complex Systems, Nº 1, pp 145-168, 1987.

[99] J. D. Burr: "Experiments with a connectionist text reader, Proceedings of the First International Conference on Neural Networks, San Diego, CA, Vol. 4, pp 717-724, 1987.

[100] G. W. Cottrell, P. Munro and D. Zipser: "Image compression by backpropagation: an example of extensional programming", Advances in Cognitive Science, Vol. 3, NJ, 1987.

[101] W. W. McCulloch and W. Pitts: "A logical calculus of the ideas imminent in nervous activity", Bulletin of Mathematical Biophysics, Nº 5, pp 115-133, 1943.

[102] W. Pitts and W. W. McCulloch: "How we know universals", Bulletin of Mathematical Biophysics, Nº 9, pp 127-147, 1947.

[103] F. Rosenblatt: "Principles of neurodynamics", Spartan Books, New York, 1962.

[104] M. L. Minsky and S. Papert: "Perceptrons", MIT Press, Cambridge, MA, 1969.

[105] R. O. Windner: "Single-state logic", AIEE Fall General Meeting, 1960.

[106] D. O. Hebb: "Organization of Behaviour", Science Editions, New York, 1961.

[107] D. E. Rumelhart, G. E. Hinton and R. J. Williams: "Learning internal representations by

error propagation", In Parallel Distributed Processing, Vol. 1, pp 318-362, MIT Press, Cambridge, MA, 1986.

[108] D. B. Parker: "Learning Logic", Invention Report S81-64, File 1, Office of Technology Licensing, Stanford University, Stanford, CA, 1982.

[109] P. J. Werbos: "Beyond regression: new tools for prediction and analysis in the behavioral sciences", Masters thesis, Harvard University, 1974.

[110] P. D. Wasserman: "Experiments in translating Chinese characters using backpropagation", Proceedings of the Thirty-Third IEEE Computer Society International Conference, Washington, DC, 1988.

[111] G. E. Hinton and T. J. Sejnowsky: "Learning and relearning in Boltzmann machines", In Parallel Distributed Processing, Vol. 1, pp 282-317, MIT Press, Cambridge, MA, 1986.

[112] H. Szu and R. Hartley: "Fast simulated annealing", Physics Letters, 1222(3,4), pp 157-162, 1987.

[113] P. D. Wasserman: "Combined backpropagation/Cauchy machine", Neural Networks: Abstracts of the First INNS Meeting, Boston, Vol. 1, p 556, 1988.

[114] R. Hecht-Nielsen: "Counterpropagation networks", Proceedings of the IEEE First International Conference on Neural Networks, Vol. 2, pp 19-32, San Diego, CA, 1987.

[115] R. Hecht-Nielsen: "Counterpropagation networks", Applied Optics, Vol. 26, Nº 23, pp 4979-4984, December 1987.

[116] R. Hecht-Nielsen: "Applications of counterpropagation networks", Neural Networks, Vol. 1, pp 131-139, 1988.

[117] T. Kohonen: "Self-organization and associative memory", 2nd edition, Springer-Verlag, New York, 1988.

[118] S. Grossberg: "Some networks that can learn, remember, and reproduce any number of complicated space-time patterns", Journal of Mathematics and Mechanics, Vol. 19, pp 53-91, 1969.

[119] S. Grossberg: "Embedding fields: underlying philosophy, mathematics, and applications of psychology, physiology, and anatomy", Journal of Cybernetics, Vol. 1, pp 28-50, 1971.

[120] S. Grossberg: "Studies of mind and brain", Reidel, Boston, 1982.

[121] M. A. Cohen and S. G. Grossberg: "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks", IEEE Transactions on Systems, Man and Cybernetics, Vol. 13, pp 815-826, 1983.

[122] J. J. Hopfield: "Neurons with graded response have collective computational properties like those of two-state neurons", Proceedings of the National Academy of Science, Vol. 81, pp 3088-3092, 1984.

[123] J. J. Hopfield: "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Science, Vol. 79, pp 2554-2558, 1982.

[124] Y. S. Abu-Mostafa and J. St. Jacques: "Information capacity of the Hopfield model, IEEE Transactions on Information Theory, Vol. 31, Nº 4, pp 461-464, 1985.

[125] B. Kosko: "Bi-directional associative memories", IEEE Transactions on Systems, Man and Cybernetics, Vol. 18, Nº 1, pp 49-60, 1987.

[126] G. A. Carpenter and S. Grossberg: "A massive parallel architecture for a self-organizing neural pattern recognition machine", Computer, Vision, Graphics and Image Processing, Vol. 37, Nº 1, pp 54-115, January 1987.

[127] G. A. Carpenter and S. Grossberg: "ART-2: self-organization of stable category recognition codes for analog input patterns", Applied Optics, Vol. 26, Nº 23, pp 4919- 4930, December 1987.

[128] G. A. Carpenter and S. Grossberg: "ART-3: hierarchical search using chemical transmitters in self-organizing pattern recognition architectures", Neural Networks, Vol. 3, pp 129-152, 1990.

[129] G. A. Carpenter, S. Grossberg and J. H. Reynolds: "ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network", Neural Networks, Vol. 4, pp 565-588, 1991.

[130] G. A. Carpenter, S. Grossberg and D. B. Rosen: "Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system", Neural Networks, Vol. 4, pp 759-771, 1991.

7. Voltage Stability Problem in Electric Power Systems (III)

[131] H. Prieto Schmidt and R. N. Adams: "Assessment of static voltage stability using artificial neural networks", Proceedings of the XI Power Systems Computation Conference, Avignon, France, 30 August - 3 September 1993.

8. Matrix Computations (II)

[132] F. R. Gantmacher: "The theory of matrices", Chelsea Publishing Company, New York, 1977.

9. Sample Power Systems

[133] J. B. Ward and H. W. Hale: "Digital computer solution of power-flow problems", AIEE Transactions - Part III - Power Apparatus and Systems, Vol. 75, pp 398-404, June 1956.

[134] L. L. Freris and A. M. Sasson: "Investigation of the load-flow problem", Proceedings IEE, Vol. 115, Nº 10, pp 1459-1470, October 1968.

10. Power System Stability

[135] P. M. Anderson and A. A. Fouad: "Power system control and stability". The Iowa State University Press, Ames, Iowa, 1977.

11. Voltage Stability Problem in Electric Power Systems (IV)

[136] Y. Sekine and H. Ohtsuki: "Cascaded voltage collapse", IEEE Transactions on Power Systems, Vol. 5, Nº 1, pp 250-256, February 1990.

BIBLIOGRAPHY

1. Voltage Stability Problem in Electric Power Systems

1. L. H. Fink (ed.): "Proceedings: bulk power system voltage phenomena, voltage stability and security", EPRI Report EL-6183, Potosi, Missouri, January 1989.

2. Y. Mansour (ed.): "Voltage stability of power systems: concepts, analytical tools, and industry experiences", IEEE Task Force Report, Publication 90TH0358-2-PWR.

3. N. W. Miller, R. D'Aquila, K. M. Jimma, M. T. Sheehan and G. L. Comegys: "Voltage stability of the Puget Sound system under abnormally cold weather conditions", IEEE Transactions on Power Systems, Vol. 8, N. 3, pp 1133-1142, August 1993.

4. C. Counan, M. Trotignon, E. Corradi, G. Bortoni, M.Stubbe and J. Deuse: "Major incidents on the French electric system: potentiality and curative measures studies", IEEE Transactions on Power Systems, Vol. 8, N. 3, pp 879-886, August 1993.

5. P.-A. Löf, G. Andersson and D. J. Hill: "Generator modelling for static voltage stability studies", Proceedings of the XI Power Systems Computation Conference, Avignon, France, 30 August -3 September 1993, pp 923-929.

6. P.-A. Löf, G. Andersson and D. J. Hill: "Voltage stability indices for stressed power systems", IEEE Transactions on Power Systems, Vol. 8, N. 1, pp 326-335, February 1993.

7. A. Kurita, H. Okubo, K. Oki, S. Agematsu, D. B. Klapper, N. W. Miller, W. W. Price, J. J. Sanchez Gasca, K. A. Wirgau and T. D. Younkins: "Multiple time-scale power system dynamic simulation", IEEE Transactions on Power Systems, Vol. 8, N. 1, pp 216-223, February 1993.

8. J. Deuse and M. Stubbe: "Dynamic simulation of voltage collapses", IEEE Transactions on Power Systems, Vol. 8, N. 3, pp 894-904, August 1993.

9. J. D. McCalley, J. F. Dorsey, J. F. Luini, R. Peter Mackin and G. H. Molina: "Subtransmission reduction for voltage instability analysis", IEEE Transactions on Power Systems, Vol. 8, N. 1, pp 349-356, February 1993.

10. B. H. Lee and K. Y. Lee: "Dynamic and static voltage stability enhancement of power systems", IEEE Transactions on Power Systems, Vol. 8, N. 1, pp 231-238, February 1993.

11. C. A. Cañizares and F. L. Alvarado: "Point of collapse and continuation methods for large AC/DC systems", IEEE Transactions on Power Systems, Vol. 8, N. 1, pp 1-8, February 1993.

2. Load Modelling

12. IEEE Task Force on Load Representation for Dynamic Performance: "Load representation for dynamic performance analysis", IEEE Transactions on Power Systems, Vol. 8, N. 2, pp 472-482, May 1993.

13. D. J. Hill: "Nonlinear dynamic load models with recovery for voltage stability studies", IEEE Transactions on Power Systems, Vol. 8, N. 1, pp 166-176, February 1993.

3. Application of Artificial Neural Networks in Electric Power Systems

14. R. Khosla and T. Dillon: "Combined symbolic-artificial neural net alarm processing system", Proceedings of the XI Power Systems Computation Conference, Avignon, France, 30 August -3 September 1993, pp 259-266.

APPENDIX

A.1 - Introduction

This Appendix contains the complete data for all study cases presented in Chapter 6. These data will be presented in the following order: load-flow data, dynamic simulation data, and interface/MLP data.

A.2 - Data for load-flow study

All load-flow base cases refer to the electrical network of Figure 6.10. Tables A.1, A.2 and A.3 present general data, bus data, and branch data respectively. It should be pointed out that the data for generator internal buses in Figure 6.10 will be presented in the dynamic simulation section.

Parameter	Value
Number of *PQ* buses	11
Number of *PQV* buses	0
Number of *PV* buses	4
Number of branches	18
Max. number of iterations (Newton-Raphson method)	15
Bus power mismatch tolerance	10^{-10} pu
MVA base	100 MVA

Table A.1 - General data for load-flow base case

Bus n°	Type	Rating (kV)	Volt. (pu)	Angle (deg.)	Load			Generation				Capacitor (MVAr)
					P (MW)	*Q* (MVAr)	Exponent	*P* (MW)	*Q* (MVAr)	Q_{min} (MVAr)	Q_{max} (MVAr)	
100	Swing	16.50	1.040	0.000	0.000	0.000	0.	99.724	31.849	-40.000	40.000	
200	*PV*	18.00	1.025	8.398	0.000	0.000	0.	163.000	7.344	-15.000 (*b*) -30.000	15.000 (*b*) 30.000	
300	*PV*	13.80	1.025	3.779	0.000	0.000	0.	85.000	-10.183	-15.000 (*b*) -30.000	15.000 (*b*) 30.000	
400	*PQ*	230.00	1.024	-3.092	0.000	0.000	0.	0.000	0.000			
500	*PQ*	230.00	0.994	-4.874	125.000	50.000	0.	0.000	0.000			
600	*PQ*	230.00	1.011	-4.572	90.000	30.000	0.	0.000	0.000			
700	*PQ*	230.00	1.025	2.835	0.000	0.000	0.	0.000	0.000			
800	*PQ*	230.00	1.015	-0.160	100.000	35.000	0.	0.000	0.000			
900	*PQ*	230.00	1.032	1.079	0.000	0.000	0.	0.000	0.000			
1000	*PQ*	230.00	1.021	-3.346	0.000	0.000	0.	0.000	0.000			
1100	*PQ*	230.00	0.977	-6.543	70.000	45.000	0.	0.000	0.000			
1200	*PQ*	230.00	0.969	-7.668	0.000	0.000	0.	0.000	0.000			4.696
1300	*PQ*	13.80	0.961	-8.898	20.000	15.000	0.	0.000	0.000			7.388
1400	*PQ*	18.00	0.959	-10.259	35.000	15.000	0.	0.000	0.000			7.351
1500	*PV*	13.80	1.040	-0.649	0.000	0.000	0.	100.000	40.925	-45.000 (*b*) -60.000	45.000 (*b*) 60.000	
Total					440.000	190.000	-	447.724	69.934	-	-	19.435

Table A.2 - Bus data for load-flow base case (solved case) (*b* for series B only)

Branch ID			Type	Tap (pu)	Rating (MVA)	Electrical parameters (pu) (†)		
From bus	To bus	Circuit #				*r*	*x*	*c*
100	400	1	Trans.	1.000	100	0.0000	0.0576	
200	700	1	Trans.	1.000	100	0.0000	0.0625	
300	900	1	Trans.	1.000	100	0.0000	0.0586	
400	500	1	Line		80	0.0100	0.0850	0.0880
400	600	1	Line		80	0.0170	0.0920	0.0790
400	1000	1	Line		25	0.0060	0.0350	0.0290
400	1000	2	Line		25	0.0060	0.0350	0.0290
500	700	1	Line		80	0.032	0.1610	0.1530
600	900	1	Line		80	0.0390	0.1700	0.1790
700	800	1	Line		80	0.0085	0.0720	0.0745
800	900	1	Line		80	0.0119	0.1008	0.1045
1000	1100	1	Line		60	0.0300	0.1000	0.0250
1000	1100	2	Line		60	0.0300	0.1000	0.0250
1000	1500	1	Transf.	1.000	100	0.0000	0.0500	
1100	1200	1	Line		25	0.0150	0.0700	0.0200
1100	*1200*	*2*	Line		25	0.0150	0.0700	0.0200
1200	1300	1	Transf.	1.000	30	*0.0000*	*0.1000*	
1200	1400	1	Transf.	1.000	25	0.0000	0.1200	

Table A.3 - Branch data for load-flow base case († 100 MVA base)

A.3 - Data for dynamic simulation study

In this section, data regarding equipment that present dynamic behaviour will be presented. Bus numbering in this case also refers to the electrical network of Figure 6.10. With the introduction of induction motors and dynamic loads, some bus loads were decreased from their original value in order to compensate for the introduction of these new equipment. Table A.4 shows the bus data that were modified with respect to the original data in Table A.2. Tables A.5 through A.9

show the data for the dynamic simulations.

Bus n°	Load			Capacitor (MVAr)
	P (MW)	*Q* (MVAr)	Exponent	
1100	40.000 (*b*) 0.000 (†)	30.000 (*b*) 0.000 (†)	0.	0.000
1300	0.000 (†)	0.000 (†)	0. (†)	8.000 (†)
1400	15.000	5.000	0.	5.000

Table A.4 - New bus data for dynamic simulation (*b* for series B only; † for series C and D only)

Parameter			Value
Modified Euler Method	Maximum number of iterations		20
	Tolerance (pu)		10^{-6}
	Integration step (s)	Series B	0.002
		Series C & D	0.005
Newton-Raph-son Method	Maximum number of iterations		20
	Tolerance (pu)		10^{-6}

Table A.5 - General data for dynamic simulation

Internal bus	External bus	Rating (MVA)	Governor model (see Table 6.2)	*r* (pu) (†)	*x* (pu) (†)	*H* (s) (†)	*D*
101	100	248	2	0.0000	0.1508	9.55152	0.
201	200	192	2	0.0000	0.2300	3.33333	0.
301	300	128	2	0.0000	0.2321	2.35156	0.
1501	1500	100	2	0.0000	0.0600	9.0000	0.

Table A.6 - Generator data († rated MVA base)

Bus	Rating (MVA)	H (s) (†)	Initial slip (pu)	r (pu) (†)	x (pu) (†)
1100	35	25.00000 10.00000 (*d*)	0.03000	0.0300	0.5000
1400	25	25.00000 10.00000 (*d*)	0.03000	0.0300	0.5000

Table A.7 - Induction motor data († rated MVA base; *d* for series D only)

Bus	Reference power (set point) (MW)	cos φ	K_l (see Eq. (6.6))
1100	30.000	0.80	0.100 (*c*) 0.400 (*d*)
1300	20.000	0.80	0.100 (*c*) 0.400 (*d*)

Table A.8 - Dynamic load data (*c* for series C only; *d* for series D only)

Branch ID			N° of taps		Tap step (pu)	Ini-tial tap	Cycle (s)	Con-trolled bus	Reference voltage (set-point) (pu)	Voltage toler-ance (pu)
From bus	To bus	Cir-cuit #	Above	Below						
1200	1300	1	8 (*b*) 5 (*c*,*d*)	8 (*b*) 5 (*c*,*d*)	0.00625 (*b*) 0.01000 (*c*,*d*)	0	20.00 (*b*) 10.00 (*c*) 4.50 (*d*)	1300	0.962 (*b*) 0.971 (*c*,*d*)	0.003125 (*b*) 0.005000 (*c*,*d*)
1200	1400	1	8 (*b*) 5 (*c*,*d*)	8 (*b*) 5 (*c*,*d*)	0.00625 (*b*) 0.01000 (*c*,*d*)	0	20.00 (*b*) 10.00 (*c*) 4.50 (*d*)	1400	0.957 (*b*) 0.966 (*c*,*d*)	0.003125 (*b*) 0.005000 (*c*,*d*)

Table A.9 - OLTC data (*b* for series B, ...)

A.4 - Data for interface/MLP - Series B

Tables A.10, A.11, A.12 and A.13 present data for the interface program, input variables, output variables and data for MLP networks respectively for series B.

Parameter		Value
Type of training/testing vector generation		Automatic
Distribution of random load values		Uniform
N° of training/testing vectors		500/500
N° of input variables		9 x 2 = 18
N° of output variables	Case TTB1	2
	Case TTB3	8
Minimum value for neural network input		-10.
Maximum value for neural network input		10.
Minimum value for neural network output		0.3
Maximum value for neural network output		0.7
Initial simulation time (s)		0.
Maximum duration of simulation (s)		300.
Probability of a branch to be open before transition		0.5
Probability of a branch to be open after transition		0.5
Load range (% of rated load)		50. ~ 150.
Reactor range (% of rated MVAr)		0. ~ 100.

Table A.10 - Data for interface program INTF2 - series B

State of branch 1000 1100 #1
State of branch 1100 1200 #1
Global load
Load at bus 1100
Load at bus 1300
Load at bus 1400
Reactor at bus 1200
Reactor at bus 1300
Reactor at bus 1400

Table A.11 - Input variables in series B

Simulation outcome
Final simulation time
Voltage at bus 1100 (†)
Voltage at bus 1200 (†)
Voltage at bus 1300 (†)
Voltage at bus 1400 (†)
Slip of motor at bus 1100 (†)
Slip of motor at bus 1400 (†)

Table A.12 - Output variables in series B († refers to case TTB3 only)

Parameter		Value
N° of layers		3
Weight initialisation range		-0.01 ~ 0.01
N° of neurons in each layer	case NNB1	18 20 10 2
	case NNB2	18 10 5 2
	case NNB3	18 30 20 8
Bias neuron?		Yes
N° of weights in each layer	case NNB1	380 + 210 + 22 = 612
	case NNB2	190 + 55 + 12 = 257
	case NNB3	570 + 620 + 168 = 1358
Training mode		Pure backpropagation
Training algorithm		Exponential smoothing
N° of external iterations in each training session		1000
Maximum number of internal iterations		3
Tolerance for internal loop		0.005
λ (see Eq. (4.5))		1.0
Exponential smoothing coeff. (α, Eq. (4.9))		0.2
Learning rate in each training session		1.0 0.9 0.8 0.7 0.6

Table A.13 - MLP data in series B

A.5 - Data for interface/MLP - Series C

Tables A.14, A.15, A.16 and A.17 present data for the interface program, input variables, output variables and data for MLP networks respectively for series C.

Parameter		Value
Type of training/testing vector generation		Automatic
Distribution of random load values		Uniform
N° of training/testing vectors	Case TTC1	345/330
	Case TTC3	1043/1165
N° of sampling times		4
N° of input variables		8 x 4 = 32
N° of output variables		2
Minimum value for neural network input		-10.
Maximum value for neural network input		10.
Minimum value for neural network output		0.3
Maximum value for neural network output		0.7
Initial simulation time (s)		0.
Maximum duration of simulation (s)		60.
Sampling interval (s)		5.
Probability of a branch to be open before simulation		0.5
Probability of a branch to be opened during simulation		0.5
Probability of a branch to be reclosed during simulation		0.5
Branch opening time (t_{open}, s)		0. ~ 5.
Branch closing time (s)		t_{open} ~ 10.
Load range (% of rated load)		0. ~ 200.
Reactor range (% of rated MVAr)		0. ~ 100.

Table A.14 - Data for interface program INTF3 - series C

State of branch 1000 1100 #1 (†)
State of branch 1100 1200 #1 (†)

Table A.15 - Input variables in series C († indicates variables also defined as control variables)

Global load (†)
Load at bus 1400 (†)
Reactor at bus 1200 (†)
Reactor at bus 1300 (†)
Reactor at bus 1400 (†)
Minimum Singular Value of matrix G_s

Table A.15 - Input variables in series C († indicates variables also defined as control variables)

Simulation extra time
Minimum Singular Value of matrix G_s

Table A.16 - Output variables in series C

Parameter		Value
N° of layers		3
Weight initialisation range		-0.01 ~ 0.01
N° of neurons in each layer	case NNC1	32 25 2
	case NNC2	32 25 20 2
Bias neuron?		Yes
N° of weights in each layer	case NNC1	825 + 52 = 877
	case NNC2	825 + 520 + 42 = 1387
Training mode		Pure backpropagation
Training algorithm		Exponential smoothing
N° of external iterations in each training session		1000
Maximum number of internal iterations		5
Tolerance for internal loop		0.005
λ (see Eq. (4.5))		1.0
Exponential smoothing coeff. (α, Eq. (4.9))		0.2
Learning rate in each training session		1.0 0.9 0.8 0.7 0.6

Table A.17 - MLP data in series C

A.6 - Data for interface/MLP - Series D

Tables A.18, A.19, A.20 and A.21 present data for the interface program, input variables, output variables and data for MLP networks respectively for series D.

Parameter		Value
Type of training/testing vector generation		Automatic
Distribution of random load values		Uniform
N° of training/testing vectors	Case TTD1	1000/1000
	Case TTD2	4891/4205
	Case TTD3	980/848
N° of sampling times		3
N° of input variables		6 x 3 = 18
N° of output variables		2
Minimum value for neural network input		-10.
Maximum value for neural network input		10.
Minimum value for neural network output		0.3
Maximum value for neural network output		0.7
Initial simulation time (s)		0.
Maximum duration of simulation (s)		30.
Sampling interval (s)		5.
Load range (% of rated load)		0. ~ 200.
Reactor range (% of rated MVAr)		0. ~ 100.

Table A.18 - Data for interface program INTF3 - series D

Global load (†)
Load at bus 1400 (†)
Reactor at bus 1200 (†)
Reactor at bus 1300 (†)

Table A.19 - Input variables in series D († indicates variables also defined as control variables)

Reactor at bus 1400 (†)
Minimum Singular Value of matrix G_s

Table A.19 - Input variables in series D († indicates variables also defined as control variables)

Simulation extra time
Minimum Singular Value of matrix G_s

Table A.20 - Output variables in series D

Parameter		Value
N° of layers		3
Weight initialisation range		-0.01 ~ 0.01
N° of neurons in each layer	case NND1	18 15 10 2
	case NND2	18 25 15 2
	case NND3	18 20 15 2
Bias neuron?		Yes
N° of weights in each layer	case NND1	285 + 160 + 22 = 467
	case NND2	475 + 390 + 32 = 897
	case NND3	380 + 315 + 32 = 727
Training mode		Pure backpropagation
Training algorithm		Exponential smoothing
N° of external iterations in each training session		1000
Maximum number of internal iterations		5
Tolerance for internal loop		0.005
λ (see Eq. (4.5))		1.0
Exponential smoothing coeff. (α, Eq. (4.9))		0.2
Learning rate in each training session		1.0 0.9 0.8 0.7 0.6

Table A.21 - MLP data in series D